Scalable Semi-Supervised Grammar Induction using Cross-Linguistically Parameterized Syntactic Prototypes

Prachya Boonkwan

Doctor of Philosophy
Institute for Language, Cognition and Computation
School of Informatics
University of Edinburgh
2014

Abstract

This thesis is about the task of unsupervised parser induction: automatically learning grammars and parsing models from raw text. We endeavor to induce such parsers by observing sequences of terminal symbols. We focus on overcoming the problem of *frequent collocation* that is a major source of error in grammar induction. For example, since a verb and a determiner tend to co-occur in a verb phrase, the probability of attaching the determiner to the verb is sometimes higher than that of attaching the core noun to the verb, resulting in erroneous attachment *((Verb Det) Noun) instead of (Verb (Det Noun)). Although frequent collocation is the heart of grammar induction, it is precariously capable of distorting the grammar distribution. Natural language grammars follow a Zipfian (power law) distribution, where the frequency of any grammar rule is inversely proportional to its rank in the frequency table. We believe that covering the most frequent grammar rules in grammar induction will have a strong impact on accuracy.

We propose an efficient approach to grammar induction guided by cross-linguistic language parameters. Our language parameters consist of 33 parameters of frequent basic word orders, which are easy to be elicited from grammar compendiums or short interviews with naïve language informants. These parameters are designed to capture frequent word orders in the Zipfian distribution of natural language grammars, while the rest of the grammar including exceptions can be automatically induced from unlabeled data. The language parameters shrink the search space of the grammar induction problem by exploiting both word order information and predefined attachment directions.

The contribution of this thesis is three-fold. (1) We show that the language parameters are adequately generalizable cross-linguistically, as our grammar induction experiments will be carried out on 14 languages on top of a simple unsupervised grammar induction system. (2) Our specification of language parameters improves the accuracy of unsupervised parsing even when the parser is exposed to much less frequent linguistic phenomena in longer sentences when the accuracy decreases within 10%. (3) We investigate the prevalent factors of errors in grammar induction which will provide room for accuracy improvement.

The proposed language parameters efficiently cope with the most frequent grammar rules in natural languages. With only 10 man-hours for preparing syntactic prototypes, it improves the accuracy of directed dependency recovery over the state-of-the-art Gillenwater et al.'s (2010) completely unsupervised parser in: (1) Chinese by 30.32% (2) Swedish by 28.96% (3) Portuguese by 37.64% (4) Dutch by 15.17% (5) German by 14.21% (6) Spanish by 13.53% (7) Japanese by 13.13% (8) English by 12.41% (9) Czech by 9.16% (10) Slovene by 7.24% (11) Turkish by 6.72% and (12) Bulgarian by 5.96%. It is noted that although the directed dependency accuracies of some languages are below 60%, their TEDEVAL scores are still satisfactory (approximately 80%). This suggests us that our parsed trees are, in fact, closely related to the gold-standard trees despite the discrepancy of annotation schemes.

We perform an error analysis of over- and under-generation analysis. We found three prevalent problems that cause errors in the experiments: (1) PP attachment (2) discrepancies of dependency annotation schemes and (3) rich morphology.

The methods presented in this thesis were originally presented in Boonkwan and Steedman (2011). The thesis presents a great deal more detail in the design of cross-linguistic language parameters, the algorithm of lexicon inventory construction, experiment results, and error analysis.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Prachya Boonkwan*)

Acknowledgements

I am tremendously indebted to many people whose guidance, help, and support make the preparation and completion of this study possible. A million thanks of mine are due here.

First and foremost, to **Mark Steedman** (University of Edinburgh), who is an exceptional, devoting, and knowledgeable advisor to all of his students, and the inventor of Combinatory Categorial Grammar (CCG). During the four years of my studentship, he showed me how to thrive in the academic world. He shaped me how to think like a computational linguist, critically contemplate research questions, meticulously design and conduct experiments, and write academically decent articles. His book, *The Syntactic Process* (Steedman, 2000), has always been motivating my ambition in an attempt to understand the behaviors of natural language grammars beyond the expressive power of context-free grammars. He became my ultimate academic role model at least a half of whose success I wish to attain one day.

Second, to **Philipp Koehn** (University of Edinburgh), who is an excellent teacher, a wonderful friend, and a renowned godfather of statistical phrase-based translation, to **Bonnie Webber** (University of Edinburgh), who is a caring and comforting teacher and whose contribution in discourse structure and question answering research is tremondous, to **Sharon Goldwater** (University of Edinburgh; her signature is $\mathrm{AuH_2O}$), whose easy-to-understand lectures, tutorials, and papers help me get a grip on mastering the Markov Chain Monte Carlo methods during the first two years of my study, to **Stephen Clark** (University of Cambridge), **Michael Collins** (Columbia University), **Luke Zettlemoyer** (University of Washington), **Adam Lopez** (Johns Hopkins University), **Noah Smith** (Carnegie Mellon University), and **Chris Dyer** (Carnegie Mellon University), who gave me a lot of suggestions in unsupervised grammar induction and research directions for my thesis during our invaluable discussion while they were spending their wonderful time in Edinburgh, to **Tejaswani Deoskar** (University of Edinbugh) for giving me insightful advice in linguistics when Mark was away, and finally to **Tom Kwiatkowski** (University of Edinburgh), a nice super-friendly *genius* and also my office mate who spent a lot of his time coaching me to deal with machine learning and its underlying mathematics, and discussed our common problems relevant to our work during his study in Edinburgh.

Third, to all my course mates including, but not limited to: **Tom Kwiatkowski**, **Aciel Eshky**, **Mark Granroth-Wilding** (Little Mark), **Christos Christodoulopoulos**

(ΤΟΣ), **Greg Coppola**, **Mike Lewis**, **Michael Auli**, **Srinivasan Janarthanam**, **David Matthews**, **Desmond Elliott**, **Saša Petrovic**, **Stella Frank**, **Benjamin Allison**, **John Pate**, **Philipp Petrenz**, **Joel Lang**, **Niyue Tan**, **Benjamin Rosman**, and so on. These people have made my time in School of Informatics, University of Edinburgh one of the most wonderful time of my life. 4PM biscuits then Pear Tree, anyone?

Fourth, a million thanks to all my dear close friends, particularly: **Chidchanok Thepsoonthorn** (Cheeze), **Wasin Sakulkoo** (Gough), **Sittikorn Assavanives** (Pepe), **Wanaruk Chaimayo** (Bong), **Thipnatee Sansawatt** (Punim), **Łukasz Kopeć**, **Darryl Alexander Geegan**, **Michael MacKenzie**, **Chatwalai Phairojwithayaporn** (Toon), **Chonchanok Theethakaew** (Mod), **Tanatip Uan-On** (Pook), and **Miroslava Hulvanova** (Mirka), for making my extra-academic life amazing, colorful, and exhilarating, and giving me a lot of help, mental support, and comforts whenever I was in trouble and needed a shoulder. Who said life-long friendship is hard to sustain?

Fifth, I am grateful for **National Science and Technology Development Agency** (Thailand) for offering me a full-time PhD scholarship in 2008. This opportunity has tremendously widened my academic perspectives towards natural language processing technology and scientific research which have strong academic and social impacts for my beloved motherland Thailand. I truly appreciate all help, mental support, and document work from my bosses: **Thepchai Supnithi**, Director of Language and Semantic Technology Lab, and **Chai Wutiwiwatchai**, Director of Speech Technology Lab, as well as all my colleagues in both labs.

Finally and most importantly, I am truly indebted to my parents, **Parinya Boonkwan** and **Laddawan Boonkwan**, as well as my sister **Prachda Boonkwan**, for their unconditional love, devotion, and mental support. I love you guys with all my heart.

My time in Edinburgh was full of happiness and challenges. I have so many fond memories with this charming Scottish city. From time to time, it is the place where I shed tears for several reasons: excitement, joys, and a bit of sadness. Edinburgh will always be my second home for the rest of my life.

To my parents who never give up supporting their son,
and my sister who has always been my life-long companion.

Contents

List of Figures

List of Tables

List of Algorithms

Part I

Preliminary

Chapter 1

Introduction

1.1 Goals and Approach

Syntactic parsing plays an important role in natural language processing. A syntactic parser analyzes an input sentence according to a set of grammar rules into a syntactic structure that carries grammatical information hidden under the surface form. This information has been shown beneficial in various NLP tasks, such as machine translation, information extraction, and document summarization.

There are generally two approaches for constructing a syntactic parser: rule-based approach and statistical approach. In the *rule-based approach*, the set of grammar rules must be handcrafted by experienced linguistic experts and they are required to cover as much of linguistic phenomena in the language as possible. However, handcrafting the rules from scratch is laborious and rule conflicts may occur due to syntactic ambiguity.

In the *statistical approach*, the parser is automatically learned from a large amount of data manually annotated with syntactic structures. Statistical parsing has shown reasonable accuracy for a few languages such as English. The best-performing parsers acquire grammars and statistical parsing models from sentences explicitly annotated with syntactic structures (Black et al., 1992).

For the training process, available supervised parsers demand large amounts of hand-labeled data, such as Penn Treebank (Marcus et al., 1993) or CCGbank (Hockenmaier and Steedman, 2007). Producing such rich linguistic resources is labor- and time-intensive, requiring well trained linguists to define and annotate syntactic categories and resolve inconsistent annotations. This procedure becomes impractical for a less-privileged language whose linguists are scarce. Consequently, there are few treebanks available for training supervised parsers.

Grammar induction was introduced to remedy this issue by automatic learning of linguistic structures from raw text. It has gained general interest for several decades, offering a possibility of building practical syntactic parsers by reducing the labor of constructing a treebank from scratch. Pioneering work in this area includes the Constituent-Context Model (CCM) (Klein and Manning, 2001b; Klein and Manning, 2002), the Dependency Model with Valence (DMV), and the mixture CCM+DMV (Klein and Manning, 2004; Klein, 2005). Several structure search techniques have been added to DMV, modeled in terms of PCFGs, in an attempt to overcome the issues of local optima and data sparsity (Smith, 2006; Cohen et al., 2008; Headden III et al., 2009). Dependency structure can also be indirectly induced from grammar formalisms other than PCFG such as Tree Substitution Grammar (Cohn et al., 2010).

Problem statement: These unsupervised techniques are prone to dependency attachment errors. Boonkwan and Steedman (2011) point out that frequent collocation can sometimes cause unexpected mistakes in parsing. For example, a verb (VBZ) and a determiner (DT) tend to co-occur in a verb phrase, resulting in bracketing ((VBZ DT) NN) instead of (VBZ (DT NN)). The distribution over the resulting grammar rules does not reflect the true distribution of natural language grammars. Natural language grammars follow a Zipfian (power law) distribution, where the frequency of any grammar rule is inversely proportional to its rank in the frequency table.

In this thesis, we attempt to solve this issue by integrating Chomsky's (1965) renowned *Principles and Parameters* Theory (P&P) and the empirical approach for induction of hierarchical linguistic structures. The P&P approach suggests that children have their innate Universal Grammar that puts strong constraints on possible grammars the children can learn from the noisy linguistic inputs. It is however daunting to prescribe such Universal Grammar that copes with both (1) a small amount of frequent grammar rules and (2) a large amount of much less frequent ones called *exceptions*. On the other hand, the empirical approach suggests that children have general-purpose learning mechanism that allows them to draw complex inferences of linguistic structures from noisy linguistic inputs (Elman et al., 1996; Goldwater, 2007). While frequent grammar rules can be prescribed in Universal Grammar, the exceptions that are not prescribed can be empirically captured if they occur frequently in the linguistic examples, allowing us to attain better accuracy.

We propose an effective approach to prototype-driven grammar induction using cross-linguistic language parameters. Our language parameters consist of a set of 33 parameters of frequent basic word orders, which are easy to be elicited from non-linguist informants. These parameters are designed to capture frequent grammar rules in the Zipfian distribution of natural languages, while the rest of the grammar can be automatically induced from unlabeled data. An initial grammar generated from such information is represented in terms of a lexicalized categorial grammar. We believe that covering the most frequent grammar rules via the syntactic prototype will have a strong impact on accuracy. Our language parameters shrink the search space of the grammar induction problem by exploiting both word order and predefined attachment directions.

We also devise an interview dialog to facilitate the process of language parameter elicitation. Although our language parameters seem to need slightly more labor to elicit than Naseem et al.'s (2010) head-dependent pairs do, it becomes easier for us to acquire the language parameters with this dialog. The dialog can be used in two ways: (1) conducting personal interviews with naïve language informants with it, and (2) indirectly observing how machine translation performs a translation task based on the dialog. Intending to shape the interview process and result interpretation, it is designed based on our 33 word-order parameters, where both correspond to each other, question against question. In each question, a linguistic unit (either an example sentence or phrase) is given alongside its pattern to observe. An informant and/or machine translation is asked to translate such linguistic unit into his own language and provide word alignment between the source and target languages.

The contribution of this thesis is three-fold. (1) We show that the language parameters are adequately generalizable cross-linguistically, as our grammar induction experiments will be carried out on 14 languages on top of a simple unsupervised grammar induction system. (2) Our specification of language parameters improves the accuracy of unsupervised parsing even when the parser is exposed to much less frequent linguistic phenomena in longer sentences when the accuracy decreases within 10%. (3) We investigate the predominant sources of errors in grammar induction which will provide room for accuracy improvement.

The use of language parameters in this thesis poses a new set of research questions about structure induction of natural language grammars, including:

- **Research Question 1:** How can we design the set of language parameters so that they can capture frequent word orders?

- **Research Question 2:** How much human labor do we have to spend in order to elicit the language parameters for a previously unseen language, compared with the less-supervised techniques?

- **Research Question 3:** How much improvement in accuracy over unsupervised grammar induction systems can we achieve by incorporating language parameters?

- **Research Question 4:** What word orders can the language parameters capture and how much does the accuracy improve according to them?

- **Research Question 5:** How well can the language parameters cope with less frequent word orders and long-tail dependencies?

- **Research Question 6:** What are the main sources of error in the prototype-driven parser?

We therefore arrange the content of the thesis as follows.

1.2 Thesis Outline

Chapter 2 This chapter discusses Chomsky's (1965) *Principles and Parameters* Theory, describes the task of grammar induction, and reviews its state-of-the-art techniques. Three approaches are explained: active learning approach, unsupervised approach, and prototype-driven approach. We develop the idea of computational preliminaries and mathematics that underlie this research. First we explain statistical techniques for grammar induction by means of the idea of statistical modeling. We then describe some successful models for grammar induction based on these statistical techniques, and explain evaluation metrics for accuracy assessment of grammar induction. Finally we describe the characteristics of the datasets used for evaluation and how we prepare them for evaluation.

Chapter 3 This chapter explains the main contribution of this thesis: our language parameterization, which are used throughout the thesis to improve the accuracy of unsupervised grammar induction. Based on *Principles and Parameters* Theory and the empirical approach, we first start by developing the motivation of this research, particularly in the benefits of incorporating syntactic prototypes (prior syntactic knowledge of the language of interest) into unsupervised models. Then we review the attempt to encode such prior knowledge. We explain our alternative language parameterization method which is linguistically motivated and easy to elicit by direct consultation with grammar compendiums, by interview with naïve informants, and by indirect observation from machine translation. Since our language parameters are designed to capture frequent word orders, each language parameter is sorted with respect to their frequency in natural language grammars. We explain the dialog we use to elicit such prior knowledge by interview with naïve informants. According to their frequencies, the hypothetical trend of the accuracy is: the accuracy rises rapidly and starts to saturate as we increase the number of language parameters in the syntactic prototype. Finally, we explain how we encode the acquired prior knowledge into the syntactic prototype. *This chapter answers Research Questions 1 and 2.*

Chapter 4 This chapter explains our method of grammar induction using prior syntactic knowledge parameterized by word orders. We explain the system overview and the algorithms we use in each step. We then describe our parsing models which are used and assessed in the experiments.

Chapter 5 This chapter presents our experiment settings and the comparison of experiment results with the state-of-the-art techniques. To make our experiments replicable, we explain how to initialize the model parameters and describe our controlled variables. The results are presented in three aspects: (1) multilingual experiments in which we compare our results with the related work and with PASCAL Challenge, (2) long-tail dependencies, and (3) scalability of language parameters. *This chapter answers Research Question 3.*

Chapter 6 This chapter presents the performance evaluation of our method when applied to each language in four ways. First we compare the directed dependency accuracies and TEDEVAL scores of each model when evaluated on the corpora of length 10. Then we analyze the best model that yields the most directed dependency accuracy in terms of performance improvement when incorporating our language parameters. We further analyze the model's capability of coping with long-tail dependencies in longer sentence lengths. Finally, we discuss the errors produced by the parser in each language in terms of over- and undergeneration. *This chapter answers Research Questions 4 and 5.*

Chapter 7 This chapter discusses three predominant problems that cause errors in the experiments: (1) PP attachment, (2) discrepancy of dependency annotation scheme, and (3) rich morphology. When we incorporate our language parameters, the accuracy of the unsupervised parser starts to approach that of the supervised parser. As a result, PP attachment becomes more obvious, which is similar to supervised parsing. Dependency annotation schemes used in the corpora are analyzed by thorough observation and categorized for ease of analysis. We also explain the criteria of languages whose directed dependency accuracy exceeds 70% tend to comply with. We also discuss how much our approximation of inflection system can improve the accuracy of grammar induction in morphologically rich languages. Finally we discuss the effects of model's expressiveness with respect to the accuracy. *This chapter answers Research Question 6.*

Chapter 8 This chapter summarizes the entire thesis together with its achievement and contributions. We review our goals and our method proposed in the thesis — the use of cross-linguistically parameterized syntactic prototypes in grammar induction. We summarize the experimental results from Chapter 5 and the error analysis from Chapter 6. We then recap the predominant problems that underlie the errors from Chapter 7. We finally present future work and conclude the thesis.

Chapter 2

Background

Outline

This chapter discusses *Principles and Parameters* Theory, describes the task of grammar induction, and reviews its state-of-the-art techniques. Three approaches are explained: supervised approach, unsupervised approach, and prototype-driven approach. We then explain mathematical background of statistical modeling and statistical techniques for grammar induction. We describe some successful models for grammar induction based on these statistical techniques. Finally, we describe the characteristics of the datasets used for evaluation and how we prepare them for evaluation.

2.1 *Principles and Parameters* Theory

The *Principles and Parameters* Theory (P&P) (Chomsky, 1965) is a nativist theory of language acquisition of children. To explain how children learn their first languages, Chomsky proposes the *argument from the poverty of the stimulus* as follows:

> It seems clear that many children acquire first or second languages quite successfully even though no special care is taken to teach them and no special attention is given to their progress. It also seems apparent that much of the actual speech observed consists of fragments and deviant expressions of a variety of sorts. Thus it seems that a child must have the ability to "invent" a generative grammar that defines well-formedness and assigns interpretations to sentences even though the primary linguistic data that he uses as a basis for this act of theory construction may, from the point of view of the theory he constructs, be deficient in various respects.

It suggests that children have their innate Universal Grammar that puts strong constraints on possible grammars the children can learn from the noisy linguistic inputs.

The Universal Grammar consists of *principles* (or rules) that determine the set of all possible grammars the children can learn, and *parameters* (or values) that the children have to learn from the linguistic inputs. In other words, it is believed that all human languages are generated from the common Universal Grammar while their cross-linguistic parameters differentiate them from each other.

Chomsky (1965) further distinguishes two aspects of any human language user: (1) linguistic *performance*, the set of all utterances the user produces in all occasions, and (2) linguistic *competence*, the set of all strings the user intuitively knows are grammatical in his/her language. These sets are partially disjoint; i.e. there are some ungrammatical utterances which the user produces in actuality. Following his suggestion, most linguistic theories focus on the linguistic competence in which grammatical sentences are represented by a kind of hierarchical syntactic representation. In terms of linguistic competence, each grammar falls into either of these two categories (Chomsky, 1964): (1) *observationally adequate* competence in which the entire set of observed grammatical sentences can be generated by the grammar, and (2) *descriptively adequate* competence in which the grammar generates the set of observed grammatical sentences and also considers other relations on those sentences with respect to the native speakers' linguistic intuition.

P&P Theory is considered controversial in computational language acquisition. Some linguists and psychologists such as (Yang, 2008; Clark and Lappin, 2010; Clark and Lappin, 2011) propose that the language acquisition task should be both linguistic and empirical; i.e. at initial, children partially acquire their first or second languages from primary linguistic data (PLD) with some sort of linguistic insights. Once they have acquired enough linguistic insights, they then move on to more noisy linguistic data to improve their linguistic performance and competence.

A significant issue of P&P Theory is that: it is daunting to manually prescribe Universal Grammar that is able to cope with (1) a small amount of frequent grammar rules and (2) a large amount of much less frequent ones called *exceptions*. Alternative empirical methods are therefore proposed to capture natural language grammars with statistical inference. The empirical approach suggests that children have general-purpose learning mechanism that allows them to draw complex inferences of linguistic structures from noisy linguistic inputs (Elman et al., 1996; Goldwater, 2007). This approach underlies the task of grammar induction which will be discussed in the next section.

2.2 Task of Grammar Induction

Grammar induction is a task of automatic learning of linguistic structures from raw text. Precisely speaking, we want to induce hierarchical syntactic structures, such as brackets, trees, and dependency structures, by observing the terminal symbols of such structures. For example, we attempt to recover tree structures from sequences of terminal symbols (i.e. words) or preterminal symbols (i.e. part of speech tags or POS tags). Let us observe the following set of example sentences.

1. the boy brought a rose to the girl
2. the girl blushed and thanked him
3. the boy was so happy

We can notice that the phrases ‘the boy’ and ‘the girl’ occur frequently and should therefore be grouped as units. At this point, we have statistically induced two phrases from the observation set. However in practice, conducting this task by observing example sequences of terminal symbols alone is daunting especially in morphologically rich languages in which words change their forms to express different grammatical functions. The overwhelming amount of word forms deteriorate the capability of automatic learning of syntactic structures because of the *data sparsity* issue, i.e. the lack of adequate data that represents the true probability distribution of our event of interest.

Nowadays it is a common practice to induce syntactic structures from the preterminal symbols instead to avoid such issue. For example, if we annotate the above example sentences with the Penn Treebank’s tagset[1], we obtain:

1. the/DT boy/NN brought/VBD a/DT rose/NN to/TO the/DT girl/NN
2. the/DT girl/NN blushed/VBD and/CC thanked/VBD him/PRP
3. the/DT boy/NN was/VBD so/RB happy/JJ

Considering only the preterminal symbols (POS tags) is much more convenient in grammar induction. We can now see that the unit [DT NN] occurs frequently throughout the example set.

[1] DT is article, NN is noun, VBD is past-form verb, TO is a specialized tag to the word ‘to’, CC is conjunction, and RB is adverb.

In this section, we review state-of-the-art techniques for grammar induction. From broad perspective, there are three approaches for automatic learning of syntactic structures. As to be described in Section 2.2.1, early attempts to imitate human's language acquisition are based on active learning, which involves time-consuming and laborious human supervision in syntax learning. Section 2.2.2 describes another approach, called unsupervised approach, which completely eradicates human supervision from grammar induction to ultimately speed up the learning process. Finally, Section 2.2.3 explains the hybrid of both aforementioned approaches to boost the accuracy and minimize human labor and supervision.

2.2.1 Active Learning Approach

The grammar induction task was first attempted in 1990s following the active learning paradigm; i.e. it requires human supervision in the grammar construction process. There are several interactive categorial grammar learning systems including EMILE (Adriaans, 1992; Adriaans, 1999), CLL (Categorial Lexicon Learner) (Watkinson and Manadhar, 2001), and GraSp (Grammar for Speech) (Henrichsen, 2002). All of them attempt to learn a categorial grammar by means of active learning.

The first system, EMILE (Adriaans, 1992; Adriaans, 1999), is an interactive categorial grammar learner based on teacher-child paradigm. Given an unannotated corpus, the system extracts subexpressions in each sentence and constructs phrase structures from them based on categorial grammar's operators (i.e. forward and backward applications), resulting in a lot of syntactic categories produced. These syntactic categories are then inspected and selected by users. EMILE is a slow learner — it requires a large amount of evidence to construct a practical grammar (5 million sentences for 50,000-word grammar in Adriaan's calculation).

CLL (Watkinson and Manadhar, 2001) is in turn an improved syntactic category guesser. Given a set of initial lexicons and an input sentence with unknown words, it makes use of an n-best probabilistic CKY parser to find the most likely set of derivation trees. It then selects the best parse that maximizes the compression rate of the lexicons; i.e. it minimizes the number of syntactic categories per lexicon. The drawback of this method is that the entire corpus has to be reparsed every time to determine the effects of newly added lexicons, ensuring the optimal compression of the resultant grammar. The best accuracy of grammar induction achieved by this approach is 51.89% on Penn Treebank II Corpus given 384 initial lexicons.

Finally, GraSp (Henrichsen, 2002) is a more interactive structure inference system that allows users to manipulate the induced syntactic categories. First, atomic categories are assigned to each lexicon. Then it applies to each sentence the category manipulation such as adding and removing argument slashes. The induction process is controlled by the *disorder score* calculated from the number of uninterpretable categories in the Gentzen-Lambek categorial grammar derivation. The induction algorithm iteratively performs until the disorder score converges. To the best of our knowledge, no qualitative evaluations on GraSp have been published yet.

These systems undergo the same issue: laborious and time-consuming human supervision. In EMILE the user needs to select correct syntactic categories out of a large amount of those produced by the system. Though CLL proposes a more convenient system for grammar construction, it still requires a set of initial lexicons. The interactive GraSp system requires human supervision while building a grammar.

2.2.2 Unsupervised Approach

To eradicate the issue of time-consuming human supervision in language structure inference, research gradually shifted to the unsupervised approach that relies on machine learning techniques. The problem of grammar induction is modeled as a search problem with statistical estimation. There are three mainstream approaches so far in learning syntax from unannotated corpora: context-distributional clustering, phrase-structural clustering, and Markov Chain Monte Carlo methods (MCMC).

2.2.2.1 Context-Distributional Clustering

One can intuitively model the notion of *constituent*, a unit of meaning, as a probability distribution of its linear contexts, such as frequently-occurring phrases that are surrounded by a variety of words. The basic idea of this approach is enumerate all possible syntactic structures in the search space (e.g. pairs of text sequences and phrase symbols, words-tags pairs) and cluster them by distinguishing more frequent constituents from the others.

Grammar induction using minimum length description (MDL) (Stolcke and Omohundro, 1994; Chen, 1995) clusters sets of sequences that can be derived into single nonterminals (phrase symbols). That is, grammar induction is perceived as a compression problem where frequent sequences of text can be replaced by nonterminal symbols, minimizing the length of resultant sentences and the number of nonterminals.

An optimization algorithm, such as the EM Algorithm is then applied to optimize the compression rate resulting in an optimal grammar. MDL can also be used to optimize the number of syntactic categories per lexicon deduced from the corpus (Osborne and Briscoe, 1997). However, these techniques suffer from the issues of large-scale processing and the tendency to chunk common functional units that are obviously not syntactic constituents; e.g. preposition + determiner such as *'in the'* in the phrase *((in the) locomotive).

Schütze (1995) reported a successful method of POS tagging using context distribution. Words are clustered to a syntactic category based on its surrounding words, or *context vectors*, and singular value decomposition (SVD) was made to reduce dimensionality of context vectors avoiding sparseness issues. He also noticed that punctuation marks are a source of errors as they are not informative enough for POS prediction.

Klein and Manning (2001a) proposed two clustering algorithms that hierarchically construct phrase structure rules from clustered sequences of POS tags. The GREEDY-MERGE Algorithm merges the most likely pair consecutive words/phrases at a time based on their divergence score calculated from the cooccurence frequency of their context vectors. The other algorithm, CONSTITUENCY-PARSER, maximizes the probability distribution of a text sequence and its context vectors by the EM Algorithm. Both positive and negative examples are taken into account to prevent biased judgement with only frequent collocation.

This work was later extended to the Constituent-Context Model (CCM) (Klein and Manning, 2001b; Klein and Manning, 2002). In this approach, the probability of a sequence of POS tags being a constituent is a joint probability of such sequence cooccuring in the dataset and its linear context. The probability of each constituent is proportional to the correction score which is an odds ratio of the probability of generating a constituent as opposed to as a distituent.

Clark (2001) introduced a combination of MDL and context-distributional clustering to avoid the issues of ambiguity and data sparseness. Alignment-based learning (ABL) (van Zaanen, 2002) also searches for the best-fit structure where constituents are disambiguated by the probability of the overlapped word and its type. The accuracy of this system is recorded at 62% when tested on the OVIS Corpus.

Categorial grammar can also be learned in an unsupervised fashion (Osborne and Briscoe, 1997; Villavicencio, 2002). The system has access to all constituent structures of each input sentence. On each node of those constituent structures, its syntactic category and the headedness of its daughter nodes are induced from the parsing model.

The headedness determines the direction of the argument slash. For example, if the syntactic category for the node is X, the left daughter is headed, and the syntactic category of the right daughter is Y, then the left daughter is assigned syntactic category X/Y. The parameters of the model are iteratively estimated with a variation of EM Algorithms. In Osborne and Briscoe's (1997) system, they induce a categorial grammar from a set of input sentences, each of which being a POS tag sequence to minimize the data sparsity issue. They experimented two parsing models based on maximum likelihood estimation (MLE) and minimum description length (MDL), and they found that MDL marginally outperforms MLE. Villacencio's (2002) system is motivated by Osborne and Briscoe's method. She induced a categorial grammar from a set of input sentences paired with corresponding logical forms. Incorporating logical forms in grammar induction helps the system propose more meaningful syntactic categories, such as logical forms help distinguish an argument from an adverbial.

2.2.2.2 Phrase-Structural Clustering

In the last approach, the notion of constituent is modeled based on the probability distribution of its context. On the other hand, in this approach, it is modeled as a probability distribution of structural derivation. The parameters of the distribution are then estimated by the Inside/Outside Algorithm (Baker, 1979), a kind of EM Algorithm.

The Inside/Outside Algorithm was first introduced by (Baker, 1979) for estimating phonological rules handcrafted in context-free grammars. In this algorithm, the system has access to i.e. all possible tree structures for each sentence which are usually packed for space compactness (Boyer and Moore, 1972; Moore, 1973; Tomita, 1987). That is, the algorithm tries to group a word/nonterminal symbol subsequence into a nonterminal symbol, forming a *syntactic derivation rule*. Once the probability of each CFG rule is assigned at random, the algorithm iteratively modifies these rule probabilities (or *parameters*) with phrase-level inside and outside scores until the expectation of the model converges. The algorithm maximizes the expectation of the probability distribution over the given corpus.

In the late 1980s, the estimation of language models for English CFG (Jelinek, 1985) and English spelling rules (Dodd, 1988) were conducted. Lari and Young (1990) reported the parameter estimation of artificial grammars and the fragility of the algorithm in dealing with local optima. As an optimization problem, finding the global optimum is affected by the starting points, making the algorithm stuck in a local optimum. Carroll and Charniak (1992) attempted to learn syntax automatically from random starting points from an unannotated corpus but they suggested that the outcome was of discouragingly poor quality. Schabes et al. (1993) reported semi-supervised phrase bracketing on partially bracketed short sentences (of length up to 15 words) drawn from the Wall Street Journal Corpus.

More recent advances in this approach are Constituent-Context Model (CCM) (Klein and Manning, 2001b; Klein and Manning, 2002) and Dependency Model with Valence (DMV) (Klein and Manning, 2004; Klein, 2005), which is the first method that achieves the accuracy higher than that of the right-branching baseline (Headden III et al., 2009). The CCM is a generative phrase bracketing model that combines the robustness of distributional clustering with the parameter search. From their experiments on the Wall Street Journal Corpus, CCM yields quite promising results: 71.1% F_1 accuracy on POS-tagged input and 63.2% F_1 accuracy on unannotated input. The DMV is a generative head-outward projective dependency model (Collins, 1999) and was introduced for recovering syntactic dependencies modeled as a set of CFG production rules (Klein and Manning, 2004; Klein, 2005). DMV alone does not perform well on the WSJ10 corpus (55.7% F_1 accuracy) because the order of dependent generation is arbitrary, resulting in structural ambiguity of dependency trees. However, the combination of DMV and CCM, abbreviated as DMV+CCM, performs best on the Wall Street Journal Corpus: 77.6% F_1 accuracy of recovering constituency and 64.5% F_1 accuracy of recovering dependency in WSJ10. The improvement in accuracy caused by the search space of dependent generation is constrained by the recovered constituency.

Structure estimation techniques, smoothing techniques, and various training strategies were introduced to deal with local optima and data sparsity in DMV. Smith (2006) used various interpolation smoothing techniques to improve the accuracy. As for the use of valency, Cohen et al. (2008) utilized an extended valency grammar which is a modification of DMV's grammar with valency information attached to each head. The accuracy of the DMV alone is significantly improved to 66.8% accuracy evaluated on the WSJ10 corpus. Spitkovsky et al. (2010) proposed a training strategy where the model which is fully trained on shorter sentences and trained on longer sentences tends to outperform the model fully trained on the entire dataset. Gillenwater et al. (2010) proposed the use of posterior regularization in the EM Algorithm in which the posterior distribution of parent-child POS tags are penalized by another joint probability distribution, accelerating the convergence of the model. Spitkovsky and Alshawi (2011a) proposed a method to overcome local optima in the EM Algorithm-based grammar induction by switching between two objective functions when reaching a local optimum until the model converges. Moreover Spitkovsky and Alshawi (2011b) also suggested a strategy to use punctuation to help with grammar induction. They imposed hard constraints of constituent forming in the search space enumeration with respect to the use of punctuation marks in English, and they regained some accuracy.

Seginer (2007) introduced an incremental dependency recovery algorithm alongside DMV and CCM. For each sentence, an input word is analyzed for possible dependency links to each word in its prefix. Each link is assigned with a non-negative weight calculated from its depth in the existing bracketing. It adds the link with the maximum weight to the dependency structure and recalculates the weights for the remainder links. The process repeats until all link weights become zero, then it reads a new word. The whole process iterates until all sentences are read.

Dependency link reranking has also been of interest where the quality of grammar induction is concerned. Reichart and Rappoport's (2009) POS-based unsupervised parse assessment (PUPA) score is used for reranking proposed constituents. The PUPA score for a constituent is computed from the frequency of its yield and the nearest surrounding POS tags (or *contexts*). This score performs well on constituency parsing. In dependency parsing, Dell'Orletta et al. (2011) devised a dependency parse score called ULISSE computed from linguistic features such as parse tree depths, depths of complement chains, and arity of verbs.

Some claim that gold-standard POS tags are unnecessary for grammar induction. In his two-staged framework, Søgaard (2011) alternatively clustered all words in an unannotated corpus into distributional clusters and ranked dependent links with the PAGERANK Algorithm (Page and Brin, 1998). Spitkovsky et al. (2011) made use of Clark's (2000) word distribution clustering algorithm before applying the DMV. They suggested that the assumption of polysemous word clustering yields slightly better results than the monosemy assumption.

However, there are still several limitations in this approach. First, the system has to have access to all possible syntactic structures in each iteration in order to estimate the probability distribution of the CFG rules. This step is time-consuming and space-inefficient in grammar induction whose search space is usually very large. Second, the approach is limited by the complexity of the models; i.e. the models used in the EM Algorithm have to be able to find its optima. Third and last, the EM Algorithm only guarantees local optima — the resulting model may not be the globally best model for a given data set. These limitations prohibit us from making use of complex models in grammar induction.

2.2.2.3 Markov Chain Monte Carlo Methods

To overcome the issues of search space enumeration, model's complexity, and local optima, research also started to focus on the Markov Chain Monte Carlo methods (MCMC). We do not have to enumerate the search space in these models. Instead we place a probability distribution over the search space. We then select from the distribution, or *sample*, a parameter value at a time with respect to such distribution and then update the probability distribution with it. We iterate this process until the distribution converges. There are three approaches of MCMC-based structure induction: CFG, Tree-Substitution Grammar (TSG), and Adaptor Grammar.

In the first approach, Johnson et al. (2007b) proposed Gibbs and Metropolis-Hastings sampling algorithms, differing from each other by their methods of random walk. In Gibbs sampling (Geman and Geman, 1984), we sample a parameter value from an easy-to-find conditional probability of each variable. Metropolis-Hastings sampling (Metropolis et al., 1953; Hastings, 1970), on the other hand, allows us to sample from any probability distributions without finding such conditional probability. This can be done by controlling the walk with a *proposal distribution*. To efficiently represent the complexity of a natural language grammar, Liang et al. (2007) modeled the probabilistic CFGs as hierarchical Dirichlet Processes. In this framework, the parameters of each node, such as rule types, terminal symbols, and productions, are sampled from their distributions whose parameters are previously sampled from another set of controlling distributions.

We can also see grammar induction as Tree-Substitution Grammar recovery (Cohn et al., 2009; Cohn et al., 2010). Despite being theoretically equivalent to CFG in terms of expressive power, we perceive a syntactic tree in TSG being composed of elementary trees, where each of which can be of more than one node height, representing replacable phrase and idiomatic structures. Each elementary tree is sampled from a training tree with any sampling algorithms.

Adaptor Grammar (Goldwater, 2007; Johnson et al., 2007a) is the other approach of MCMC-based structure induction. Adaptor Grammar is a modification of probabilistic CFG. A distribution over each nonterminal, called *adaptor distribution*, is also added to make it possible to make an independent assumption for the choice of generating a syntactic tree. Adaptor Grammars can be trained by either MCMC or variational Bayesian methods (Cohen et al., 2010). Adaptor Grammars are often used for modeling word segmentation and morphological analysis (Johnson, 2008; Johnson and Goldwater, 2009; Johnson and Demuth, 2010).

MCMC grammar induction may also utilize other clues to improve the accuracy. Snyder et al. (2009) proposed the use of word alignment from bilingual corpora to help eliminating the unexpected frequent collocation. Their assumption was based on the fact that some ambiguous syntactic structures in one language correspond to less ambiguous ones in other languages. To avoid excessively random walks, Naseem et al. (2010) uses a small amount of (universal) linguistic knowledge as soft contrains to control their MCMC-based dependency generation algorithm following the DMV.

2.2.3 Prototype-Driven Approach

Despite many techniques proposed for grammar induction, the performance of completely unsupervised systems is still inadequate for practical use in natural language parsing, because these models can still be misled by *frequent collocation*.

Frequent collocation makes the unsupervised models too ambiguous to represent the probability distribution of linguistic phenomena. A common example of it is the collocation of a determiner (DT) and an adjective (JJ) in a noun phrase. This collocation misleads the models to form an incorrect noun phrase structure by grouping a frequent couple of an article and an adjective and then grouping them with a noun (NN), resulting in an incorrect structure such as *((DT JJ) NN). This adversely affects the accuracy of the unsupervised grammar induction. Therefore, the distribution over the resulting grammar does not truly reflect the probability distribution of natural language grammars.

Natural language grammars follow a Zipfian (power law) distribution, where the frequency of any grammar rule is inversely proportional to its rank in the frequency table. For example, plotting the frequency of each grammar rule used in the Penn Treebank 30 (sentences of length up to 30 words) obtains the distribution in Figure 2.1.

To avoid this problem, the use of a syntactic prototype was proposed. The notion of *syntactic prototype* is a small amount of fundamental linguistic knowledge in any forms that can be used to guide unsupervised grammar induction, such as basic word orders and phrase structure rules. It considerably improves the accuracy of structure recovery by eliminating phrase construction from ungrammatical frequent collocation, thus greatly constraining the search space. A syntactic prototype can be either *universal* (i.e. it is built once and then used for all languages) or *ad hoc* (i.e. it can be rapidly built for one language).

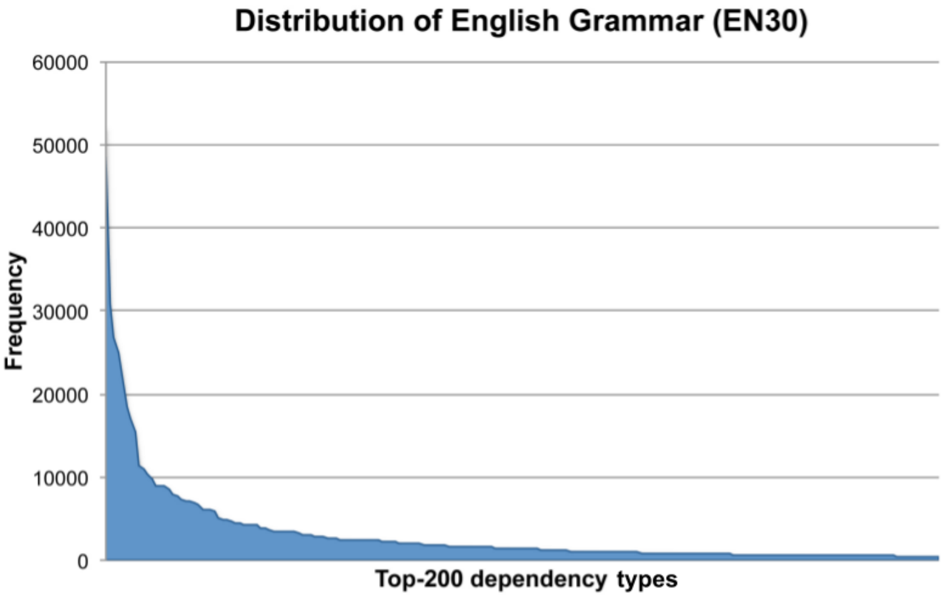


Figure 2.1: Distribution of English grammar of Penn Treebank 30

In the recent literature, syntactic prototypes are used as either soft or hard constraints for various machine learning techniques. Haghighi and Klein (2006) proposed the use of bracketing rules extracted from WSJ10 in CCM. They initialized bracketing parameters in CCM by biasing them with the postulated bracketing rules so that they could improve the accuracy. Druck et al. (2009) proposed the use of dependency formation heuristics encoded as feature functions for non-projective dependency tree CRFs. The best accuracy of theirs is around 71% when they incorporated all 60 linguistic constraints to the system. They also discovered that the accuracy starts to saturate when incorporating more linguistic constraints. Snyder et al. (2009) proposed a semi-supervised grammar induction from bilingual text with the help of a supervised parser on one side and statistical word alignment. Naseem et al. (2010) proposed the use of *universal* linguistic knowledge represented as a set of allowable head-dependent pairs. Boonkwan and Steedman (2011) suggested that parameterized *ad hoc* syntactic prototypes, when used as hard constraints for grammar induction, can considerably improve the accuracy of dependency recovery in some languages such as Chinese and Japanese. Bisk and Hockenmaier (2012a; 2012b) induced an inventory of language-specific types from unlabeled training data by the use of more general linguistic knowledge, such as that sentences are headed by verbs, which take arguments headed by nouns. All of these techniques show significant improvement in accuracy of grammar induction on corpora of short sentences.

Phrase	Prototypes
NP	DT NN
	JJ NNS
	NNP NNP
S	PRP VBD DT NN
	DT NN VBD IN DT NN
PP	IN NN
	TO CD CD
	IN PRP
ADVP	RB RB
	RB CD
	RB CC RB

Phrase	Prototypes
VP	VBN IN NN
	VBD DT NN
	MD VB CD
QP	CD CD
	RB CD
	DT CD CD
ADJP	RB JJ
	JJ
	JJ CC JJ

Figure 2.2: Haghighi and Klein's (2006) English phrase type prototype list

There are four syntactic prototypes related to the design of our language parameters: phrase structure prototypes (Haghighi and Klein, 2006), dependency grammar prototype (Druck et al., 2009), universal linguistic knowledge (Naseem et al., 2010), and category induction scheme (Bisk and Hockenmaier, 2012a; Bisk and Hockenmaier, 2012b).

2.2.3.1 Phrase Structure Prototypes

Haghighi and Klein (2006) employed the *phrase structure prototype* to boost the accuracy of bracket recovery based on CCM (Klein and Manning, 2004) and PCFG.

As shown in Figure 2.2, a phrase structure prototype consists of a list of phrase labels and their possible daughters. Since the experiment was on English, this prototype is manually handcrafted from a list of POS tag sequences of the seven most frequent phrase categories in the Penn Treebank. Each entry can be seen as a set of flattened phrases where each of their preterminal tags are concatenated.

This kind of syntactic prototype is used as soft constraints for unsupervised learning. Since the syntactic prototype merely resembles a phrase structure grammar, it is used as a local factor for each bracket. A bracket is preferred, or biased, if its yield is in the syntactic prototype.

Constraint	Expectation
MD → VB	1.00
POS ← NN	0.75
JJ ← NNS	0.75
NNP ← POS	0.75
ROOT → MD	0.75
ROOT → VBD	1.00
ROOT → VBP	0.75
ROOT → VBZ	0.75
TO → VB	1.00
VBN → IN	0.75

Constraint	Expectation
NNS ← VBD	0.75
PRP ← VBD	0.75
VBD → TO	1.00
VBD → VBN	0.75
NNS ← VBP	0.75
PRP ← VBP	0.75
VBP → VBN	0.75
PRP ← VBZ	0.75
NN ← VBZ	0.75
VBZ → VBN	0.75

Figure 2.3: Druck et al's (2009) English dependency grammar prototype called *oracle constraints*

2.2.3.2 Dependency Grammar Prototype

Druck et al. (2009) employed the *dependency grammar prototype* (Druck et al., 2009) to improve the accuracy of unsupervised non-projective dependency parsing based on Conditional Random Field (CRF).

This kind of syntactic prototype consists of oracle constraints as in Figure 2.3. The oracle constraints contain a list of head and dependent tags and attachment directions, represented in the form $H \rightarrow D$ and $D \leftarrow H$, where H is a head tag and D is a dependent tag. Each entry of the oracle constraints is assigned an expected value. The oracle constraints are provided by automatically selecting each dependency attachment from the gold corpus based on statistics of edge probability.

The oracle constraints are used as soft constraints for the CRF-based parsing model; i.e. it is a feature function for each dependency attachment. Again, a dependent attachment is preferred if it is prescribed in the oracle constraints.

2.2.3.3 Universal Linguistic Knowledge

Naseem et al. (2010) proposed *universal linguistic knowledge* which generalizes the dependency grammar prototype by separating the prototype from corpus-specific tagsets.

As shown in Figure 2.4, their syntactic prototype consists of two kinds of knowledge: the universal linguistic knowledge and the language-specific dependency rules. The universal linguistic knowledge encodes cross-linguistic dependency rules represented in the same format as the oracle constraints's one. Each rule defines only the attachment relation between coarse tags but not the ordering information. The other part of the syntactic prototype is the language-specific dependency rules as shown in Figure 2.4(b).

Being considerably dependent on a language of interest and tentatively on a specific corpus, these rules deal with various depths of syntax ranging from determining the headword of the sentence to identifying noun phrase boundaries. Furthermore, the method assumes that the corpus have to be annotated with their coarse tagset, so a mapping table between the coarse tagset and the corpus-specific tagset must also be provided in corpus preparation.

This kind of syntactic prototype is used as hard constraints for a hierarchical Dirichlet process parsing model, minimizing the search space and greatly improving the accuracy.

2.2.3.4 CCG Induction

Finally, Bisk and Hockenmaier (2012b) proposed the use of *derivation rules in Combinatory Categorial Grammar* (CCG) in grammar induction. As illustrated in Figure 2.5, there are two kinds of knowledge: the initial tags for basic categories and the CCG derivation rules. The first part is a mapping table between basic POS tags and atomic categories. The other part is a schema of CCG derivations where two syntactic categories are combined to form another syntactic category.

For each sentence, the initial tags are first assigned to basic categories: namely, noun, verb, pronoun, numeral, determiner, and conjunction. Then they extend their lexical inventory by deducing the combinator categories (i.e. complex categories) from the already known arguments, which forms plausible syntactic structures.

The most likely syntactic structure is selected by the predicate-argument parsing model closely related to (Hockenmaier, 2003a; Hockenmaier, 2003b). For comparison, dependency structures are induced from the syntactic derivations by the model in which heads and dependents are identified statistically. We consider this kind of syntactic prototype as hard constraints for unsupervised learning because it explicitly eliminates implausible syntactic categories from the search space.

Root → Auxiliary	Noun → Adjective
Root → Verb	Noun → Article
Verb → Noun	Noun → Noun
Verb → Pronoun	Noun → Numeral
Verb → Adverb	Preposition → Noun
Verb → Verb	Adjective → Adverb
Auxiliary → Verb	

(a) Universal linguistic knowledge

1. Identify non-recursive NPs:
   (a) All nouns, pronouns, and possessive marker are part of an NP.
   (b) All adjectives, conjunctions, and determiners immediately preceding an NP are part of the NP.
2. The first verb or modal in the sentence is the headword.
3. All words in an NP are headed by the last word in the NP.
4. The last word in an NP is headed by the word immediately before the NP if it is a preposition, otherwise it is headed by the headword of the sentence if the NP is before the headword, else it is headed by the word preceding the NP.
5. For the first word set its head to be the headword of the sentence. For each other word set its headword to be the previous word.

(b) English-specific dependency rules

Figure 2.4: Naseem et al's (2010) syntactic prototype

CONJ	→	conj
DET, NOUN, NUM, PRONOUN	→	N
VERB	→	S

(a) Initial tags for basic categories

Functional application	X/Y	Y	$\Rightarrow$	X	$(\mathsf{B}^0_>)$
	Y	$X\backslash Y$	$\Rightarrow$	X	$(\mathsf{B}^0_<)$
Functional composition	X/Y	Y/Z	$\Rightarrow$	X/Z	$(\mathsf{B}^1_>)$
	$Y\backslash Z$	$X\backslash Y$	$\Rightarrow$	$X\backslash Z$	$(\mathsf{B}^1_<)$
Crossed composition	X/Y	$Y\backslash Z$	$\Rightarrow$	$X\backslash Z$	$(\mathsf{B}^1_{\times>})$
	Y/Z	$X\backslash Y$	$\Rightarrow$	X/Z	$(\mathsf{B}^1_{\times<})$

(b) CCG derivation rules

Figure 2.5: Bisk and Hockenmaier's (2012) category induction scheme

2.2.3.5 Discussion

In this thesis we focus on dependency structure recovery; therefore, we focus on the use of syntactic prototype in grammar induction in two aspects: improving the accuracy of grammar induction and facilitating the task of language encoding. In Haghighi and Klein's (2006) and Druck et al.'s (2009) syntactic prototypes, linguistic knowledge is automatically extracted from the corpora and are used as soft constraints. Naseem et al. (2010), on the other hand, handcrafted their universal linguistic knowledge and their detailed language-specific dependency rules.

Particularly, Naseem et al.'s (2010) English-specific dependency rules are painstakingly built based on insightful observation on Penn Treebank in order to implicitly capture the word order of English. They are however unsystematic and laborious to achieve when this method is applied on low-resource languages.

One way to facilitate the construction of syntactic prototypes is to explicitly incorporate word orders and dependency attachment to the syntactic prototype. For example, the word order of English can be easily encoded as in Figure 2.6. The word order information is systematically simpler and requires far less effort to observe than the linguistic behaviors in an entire corpus.

Word orders are more convenient to elicit according to the availability of syntax compendiums such as The *World Atlas of Language Structures* (Haspelmath et al., 2005), grammar textbooks, and various treebank annotation manuals. Alternatively, especially in less studied languages, they can be achieved either by direct interview

subject **verb** indirect-object object subject **modal**	**preposition** noun-phrase
	relative-pronoun verb-phrase
modal verb-phrase	adverb **verb-phrase** **verb-phrase** adverb adverb **adjective** adverb **adverb**
adjective **noun** determiner **noun** **noun** nominal-modifier	

Figure 2.6: Word order of English. Headwords are underlined and bolded and the word order matters.

with linguistics-literated informants, indirectly by translation interview with naïve informants, or, less preferably, by automatic machine translation. The design of our language parameters and how it enhances the accuracy of grammar induction will be elaborated in Chapter 3.

2.3 Computational Preliminaries

2.3.1 Statistical Modeling

In learning problems, we wish to accurately predict a stochastic event from our hypothesis based on previous observations. The model used in prediction is called a *statistical model*.

We can assess how well a particular event is represented by our model. Let us denote a set of observations, or a *dataset*, $\mathcal{D} = \{x_1, x_2, x_3, \ldots, x_n\}$, where each x_i is an observation, and Θ as a hypothesis. The more the probability of the hypothesis Θ given the dataset $\mathcal{D}$, the better the model. That is, we want to find the best hypothesis Θ^*, such that

$$\Theta^* \quad = \quad \arg\max_{\Theta} P(\Theta|\mathcal{D}) \tag{2.1}$$

The probability $P(\Theta|\mathcal{D})$ is called the *posterior probability* (or *a posteriori probability*). We rewrite Eq (2.1) as

$$\Theta^* \quad = \quad \arg\max_{\Theta} P(\Theta|x_1, x_2, x_3, \ldots, x_n) \tag{2.2}$$

Generally the probability in Eq (2.2) is hard to factorize. We therefore apply Bayes' rule to obtain:

$$\begin{aligned}\Theta^* &= \arg\max_{\Theta} \frac{P(\mathcal{D}|\Theta)P(\Theta)}{P(\mathcal{D})} \\ &= \arg\max_{\Theta} \frac{P(\mathcal{D}|\Theta)P(\Theta)}{\sum_{\Theta'} P(\mathcal{D}|\Theta')P(\Theta')} \\ &= \arg\max_{\Theta} P(\mathcal{D}|\Theta)P(\Theta) \qquad (2.3)\end{aligned}$$

in which $P(\mathcal{D}|\Theta) = P(x_1, x_2, x_3, \ldots, x_n|\Theta)$ can easily be factorized to $\prod_{i=1}^{n} P(x_i|\Theta)$, if we assume that each observation x_i is independent and identically distributed. We call $P(\mathcal{D}|\Theta)$ a *likelihood*, which evaluates how well the hypothesis Θ describes the dataset $\mathcal{D}$. The probability $P(\Theta)$ is the *prior probability* of the hypothesis Θ. $P(\mathcal{D})$, called *marginal likelihood*, is the probability of the dataset.

We have come to terms with the question of how to *estimate* these parameters so that the model can well represent the dataset. One way to achieve such goal is analytical parameter estimation. In this approach, we estimate the model parameters by treating the maximization of posterior probability as an optimization problem. That is, we directly find a set of parameters that make the posterior probability an optimal point. Popular methods in this approach include maximum likelihood estimation (MLE), expectation maximization algorithm (EM), maximum *a posteriori* estimation (MAP), and variational Bayesian EM (VBEM).

2.3.1.1 Maximum Likelihood Estimation (MLE)

In this method we assume that the prior probability of each Θ is uniform. We then estimate the parameters Θ with respect to a dataset $\mathcal{D}$, whose each observation is assumed to be independent and identically distributed. Recalling the posterior probability, we find Θ^* such that it maximizes the probability of the parameters given the dataset $\mathcal{D}$.

$$\Theta^* = \arg\max_{\Theta} P(\mathcal{D}|\Theta)P(\Theta) \qquad (2.4)$$

In MLE, we assume that $P(\Theta)$ is uniform. We have

$$\Theta^* = \arg\max_{\Theta} P(\mathcal{D}|\Theta) \qquad (2.5)$$

Since $\mathcal{D} = \{x_1, x_2, x_3, \ldots\}$ where each x_i is an observation, we rewrite:

$$\Theta^* = \arg\max_{\Theta} \prod_{i=1}^{|\mathcal{D}|} P(x_i|\Theta) \qquad (2.6)$$

To find Θ^*, we can treat Eq (2.6) as an optimization problem and find its analytical solution.

2.3.1.2 Expectation Maximization Algorithm (EM)

It is not always possible to find the analytical solution to MLE. In this case, Expectation Maximization Algorithm (Dempster et al., 1977) can be employed to perform MLE when not all variables are observed although the global optimum is not guaranteed. Given a dataset $\mathcal{D}$ and a set of observed parameters Θ, let us assume $\mathbf{Z}$ be the latent variables. We want to estimate the likelihood probability $P(\mathcal{D}|\Theta)$, where

$$P(\mathcal{D}|\Theta) \quad = \quad \sum_{\mathbf{Z}} P(\mathcal{D}, \mathbf{Z}|\Theta) \tag{2.7}$$

The EM algorithm seeks for the MLE of this equation by iterating the following steps.

1. **Expectation Step (E-Step):** Calculate the expectation (i.e. weighted average) of the log likelihood function; i.e.

   $$Q(\Theta^{(t+1)}|\Theta^{(t)}) \quad = \quad \mathbb{E}_{P(\mathbf{Z}|\mathcal{D},\Theta^{(t)})}\left[\log P(\mathcal{D}, \mathbf{Z}|\Theta^{(t)})\right] \tag{2.8}$$

   where $\mathbf{Z}|\mathcal{D}, \Theta^{(t)}$ are the latent variables $\mathbf{Z}$ given the dataset $\mathcal{D}$ and the current parameters $\Theta^{(t)}$.

2. **Maximization Step (M-Step):** Find the known parameters that maximize the expectation.

   $$\Theta^{(t+1)} \quad = \quad \arg\max_{\Theta} Q(\Theta|\Theta^{(t)}) \tag{2.9}$$

   This step is an optimization problem.

These steps are iterated until the likelihood probability in Eq (2.7) converges. In practice, the termination condition is usually set to the difference of the likelihood probabilities being less than a predefined *convergence threshold* $\epsilon > 0$ which is a small constant. The expectation maximization algorithm repeatedly iterates over these equations and estimates all parameters according to them until the likelihood probability converges.

Expectation Maximization has two disadvantages. First, it is guaranteed to converge to a local optimum of the likelihood probability, not always the global optimum. This is because the likelihood probability may have more than one optimum. Second, as stated in Goldwater's (2007) thesis, EM is suitable for a learning problem whose number of parameters is known. If the number of parameters is unknown, the algorithm prefers a more complex model (i.e. more number of latent variables) to better fit the data. This in turn leads to the model to over-fit the dataset. To overcome the data over-fitting issue, we should instead consider that each hypothesis should have a different prior probability. It is therefore necessary to take into account the a posteriori probability.

2.3.1.3 Maximum A Posteriori Estimation (MAP)

In this method, we assume that each hypothesis has a different prior probability. This method is suitable for a statistical model where we have some prior knowledge about the dataset. Unlike MLE, we seek to find Θ^* such that it maximizes the a posteriori probability; i.e.

$$\begin{aligned}\Theta^* &= \arg\max_{\Theta} P(\Theta|\mathcal{D}) \\ &= \arg\max_{\Theta} P(\mathcal{D}|\Theta)P(\Theta) \qquad (2.10)\end{aligned}$$

The prior probability $P(\Theta)$ is chosen based on our prior knowledge about $\mathcal{D}$.

Another way to put a bias on the hypothesis is to favor its simplicity by using the minimum description length principle (MDL) (Rissanen, 1978; Rissanen, 1989) in which each hypothesis is biased by its information-theoretic *code length*. The MDL principle is suitable for learning problems in which the number of parameters can change over time, such as word segmentation where different segmentation leads to different number of words. We assume that simpler hypotheses can encode the dataset more efficiently and are thus more preferable. This makes learning similar to a data compression problem. That is,

$$\begin{aligned}\Theta^* &= \arg\max_{\Theta} P(\mathcal{D}|\Theta)P(\Theta) \\ &= \arg\min_{\Theta} [\mathsf{length}(\mathsf{encoding}_{\Theta}(D)) + \mathsf{length}(\Theta)] \\ &= \arg\min_{\Theta} [-\log P(\mathcal{D}|\Theta) + \mathsf{length}(\Theta)] \qquad (2.11)\end{aligned}$$

There are several measures for the length of the hypothesis, most of which information-theoretically motivated, such as entropy, relative entropy, Kullback-Leibler divergence, and Fisher's information. Alternatively, Rissanen (1978) proposed the use of Kolgomorov's complexity that he further approximates with

$$\mathsf{length}(\Theta) = k \log \sqrt{n} \qquad (2.12)$$

where n is the size of the dataset $\mathcal{D}$ and k is the number of parameters in Θ.

2.3.1.4 Variational Bayesian Approximation (VB)

In variational Bayesian methods (Attias, 2000; Ghahramani and Beal, 2000; Beal, 2003), we approximate a posterior probability as well as its observed and latent variables with factorizable and analytically solvable distributions called *variational distributions*. The posterior probability can be achieved by computing:

$$\begin{aligned} P(\Theta, \mathbf{Z}|\mathcal{D}) &= \frac{P(\mathcal{D}, \mathbf{Z}|\Theta)P(\Theta)}{P(\mathcal{D})} \\ &= \frac{P(\mathcal{D}, \mathbf{Z}|\Theta)P(\Theta)}{\sum_{\Theta}\sum_{\mathbf{Z}} P(\mathcal{D}, \mathbf{Z}|\Theta)P(\Theta)} \end{aligned} \tag{2.13}$$

For a complex model, the posterior probability cannot be computed analytically because the integral over the model parameters Θ and the latent variables $\mathbf{Z}$ is intractable. We then approximate $P(\Theta, \mathbf{Z}|\mathcal{D})$ with mean field approximation:

$$\begin{aligned} P(\Theta, \mathbf{Z}|\mathcal{D}) &\approx Q(\Theta, \mathbf{Z}) \\ &= Q(\Theta)Q(\mathbf{Z}) \end{aligned} \tag{2.14}$$

where $Q(\Theta)$ and $Q(\mathbf{Z})$ are variational distributions. By Jensen's inequality we obtain the lower bound of $\log P(\mathcal{D})$ as follows:

$$\begin{aligned} \log P(\mathcal{D}) &\geq \sum_{\Theta}\left[\mathrm{KL}(Q(\Theta)||P(\Theta)) + \sum_{\mathbf{Z}} \mathrm{KL}(Q(\mathbf{Z})||P(D, \mathbf{Z}|\Theta))\right] \\ &= \mathcal{F}(Q(\Theta), Q(\mathbf{Z})) \end{aligned} \tag{2.15}$$

where $\mathrm{KL}(Q||P)$ is Kullback-Leibler divergence of the distribution Q from P. Quantity $\mathcal{F}(Q(\Theta), Q(\mathbf{Z}))$ denotes the lower bound of $\log P(\mathcal{D})$.

From Eq (2.15), an approximation of $\log P(\mathcal{D})$ can be achieved by maximizing $\mathcal{F}(Q(\Theta), Q(\mathbf{Z}))$, which is again an optimization problem. Attias (2000) introduced an iterative algorithm to maximize the free energy which is summarized as follows.

1. **VB Expectation Step (VBE Step):** We compute the variational distribution of the latent variables $Q(\mathbf{Z})$ by solving

$$\frac{\partial \mathcal{F}(Q(\Theta), Q(\mathbf{Z}))}{\partial Q(\mathbf{Z})} = 0 \tag{2.16}$$

   We then obtain that

$$Q(\mathbf{Z}) \propto \exp \mathbb{E}_{Q(\Theta)}\left[\log P(\mathcal{D}, \mathbf{Z}|\Theta)\right] \tag{2.17}$$

2. **VB Maximization Step (VBM Step):** We compute the variational distribution of the observed parameters $Q(\Theta)$ by solving

$$\frac{\partial\mathcal{F}(Q(\Theta)Q(\mathbf{Z}))}{\partial Q(\Theta)} = 0 \tag{2.18}$$

We then obtain that

$$Q(\Theta) \propto P(\Theta)\exp\mathbb{E}_{Q(\mathbf{Z})}\left[\log P(\mathcal{D},\mathbf{Z}|\Theta)\right] \tag{2.19}$$

The algorithm proceeds in the same fashion as the canonical EM algorithm. First the variational distribution of the latent variables $Q(\mathbf{Z})$ is computed. Then the variational distribution of the observed parameters is computed, thus adjusting the model parameters. The algorithm iterates these two steps until convergence. The difference between EM and VBEM is that VBEM places priors over the numerator and denominator by adding them with the hyperparameters and then scales them with function $\exp(\psi(\cdot))$ that is less sensitive to noise than arithmetic division.[2]

2.3.2 Methods for Grammar Induction

2.3.2.1 Grammar Induction as Statistical Inference

Grammar induction can be seen as a stochastic event where a syntactic structure is statistically predicted from an input symbolic sequence based on previous observations. The grammar induction problem can be formally defined as follows. Let us first define a probabilistic context-free grammar (PCFG)

$$G = (V_N, V_T, R, S, \Theta) \tag{2.20}$$

where

- V_N is a set of nonterminal symbols (i.e. node labels),
- V_T is a set of terminal symbols (i.e. words),
- $R \subseteq V_N \times (V_N \times V_N \cup V_T)$ is a set of grammar rules, each of which being either $A \rightarrow BC$ (the branching form) or $A \rightarrow w$ (the terminal form), where $A, B, C \in V_N$ and $w \in V_T$,
- $S \in V_N$ is the start symbol (i.e. the root node's symbol), and

[2] $\psi(\cdot)$ is the digamma function, where $\psi(x) = \frac{d}{dx}\log\Gamma(x)$, and $\Gamma(x)$ is the gamma function which is a generalization of $x!$ on real numbers.

- $\Theta = \{\pi_{A\to BC} | A \to BC \in R\} \cup \{\pi_{A\to w} | A \to w \in R\}$ is a set of parameters (i.e. probabilities of each rule), having an equality constraint for each $A \in V_N$

$$\sum_X \pi_{A\to X} \;=\; 1 \tag{2.21}$$

  where $A \to X \in R$ is either in the branching or terminal forms.

For simplicity, the grammar rules in R are in Chomsky Normal Form to which all context-free grammars can be converted. Each $A \to BC$ is said to be in the *branching* form and each $A \to w$ in the *terminal* form.

There are two steps in grammar induction: training and decoding. In the training step, we want to learn the parameters Θ automatically from the given dataset $\mathcal{D} = \{s_1, s_2, s_3, \ldots\}$, where each $s_i \in V_T^+$ is a sentence which is a non-empty string of V_T. Formally speaking, we seek to find the optimal parameters Θ^* such that

$$\Theta^* \;=\; \arg\max_\Theta P(\Theta|\mathcal{D}) \tag{2.22}$$

Since each $\pi \in \Theta$ is a rule probability, we have to enumerate all possible syntactic structures of each sentence, similar to treating latent variables; i.e.

$$\begin{aligned}\Theta^* &= \arg\max_\Theta P(\Theta|\mathcal{D}) \\ &= \arg\max_\Theta P(\mathcal{D}|\Theta)P(\Theta)\end{aligned} \tag{2.23}$$

We then enumerate all syntactic structures for each $s_i \in \mathcal{D}$, yielding

$$\begin{aligned}\Theta^* &= \arg\max_\Theta P(\mathcal{D}|\Theta)P(\Theta) \\ &= \arg\max_\Theta \prod_{i=1}^{|\mathcal{D}|} P(s_i|\Theta)P(\Theta) \\ &= \arg\max_\Theta \prod_{i=1}^{|\mathcal{D}|} \sum_{j=1}^{N_i} P(s_i, t_{ij}|\Theta)P(\Theta)\end{aligned} \tag{2.24}$$

where each latent variable t_{ij} is one of the possible syntactic structures of the sentence s_i. Assuming that each grammar rule in t_{ij} is independent and identically distributed, we elaborate Eq (2.24) with

$$\Theta^* \;=\; \arg\max_\Theta \prod_{i=1}^{|\mathcal{D}|} \sum_{j=1}^{N_i} P(\Theta) \prod_{r\in\mathcal{R}_{ij}} P(s_i, r|\Theta)^{c_{ij}^r} \tag{2.25}$$

where c_{ij}^r is the count of the rule $r \in \mathcal{R}_{ij}$ in the syntactic structure t_{ij}.

At first glance, it seems easy to apply the EM algorithm to the grammar induction problem. For example, we assume that the prior probability for each model is uniform and the following condition for each expected count $q_{ij}^{A \to X}$ holds:

$$\sum_i \sum_j \sum_X q_{ij}^{A \to X} = 1 \qquad \text{for each } A \in V_N \tag{2.26}$$

The EM algorithm for MLE for Eq (2.25) is as follows.

1. **E Step:** We compute each $q_{ij}^{A \to X}$ by

$$\widehat{q}_{ij}^{A \to X} = \frac{q_{ij}^{A \to X} (\pi_{A \to X})^{c_{ij}^{A \to X}}}{\sum_i \sum_j \sum_{A \to X'} q_{ij}^{A \to X'} (\pi_{A \to X'})^{c_{ij}^{A \to X'}}} \tag{2.27}$$

2. **M Step:** We estimate each $\pi_{A \to X} \in \Theta$ by

$$\widehat{\pi}_{A \to X} = \frac{\sum_i \sum_j c_{ij}^{A \to X} q_{ij}^{A \to X} \pi_{A \to X}}{\sum_i \sum_j \sum_{A \to X'} c_{ij}^{A \to X'} q_{ij}^{A \to X'} \pi_{A \to X'}} \tag{2.28}$$

However, computation of these equations is time-consuming and space-inefficient, because arbitrary tree enumeration requires a $O(2^n)$ search space that causes combinatory explosion. To overcome this hindrance, it is necessary to apply a dynamic programming algorithm similar to the standard Forward-Backward Algorithm (Baum et al., 1970) and Inside/Outside Algorithm (Baker, 1979; Lari and Young, 1990).

2.3.2.2 Variational Bayesian Inside-Outside Algorithm

Instead of the standard Inside-Outside algorithm, we exploit the variational Bayesian expectation maximization algorithm (VBEM) (Kurihara and Sato, 2006) to approximate the parameters of the parsing models, because it is shown to outperform the original algorithm.

Kurihara and Sato (2006) show that VBEM for PCFG is less data-overfitting than the Inside-Outside Algorithm. They theoretically claimed that this is due to the fact that VBEM uses the free energy as a criterion of model selection. They state that VBEM outperforms other model selection criteria, such as MDL and Bayesian Information Criterion (BIC), because these models do not fit with non-identifiable probabilistic models, such as HMM, maximum entropy, and PCFG, according to their singular Fisher information matrices. We are also aware of other structure search techniques such as deterministic annealing, skewed deterministic annealing, and structural annealing (Smith, 2006). One can improve the VBEM with these techniques for his own preference.

In practice, this algorithm estimates the parameters with a corpus and a set of prior hyperparameters. We assume that $P(\Theta)$ is a Dirichlet distribution as aforementioned and we denote $\mathbf{u}^{\text{prior}} = \{u_r^{\text{prior}} | r \in \mathcal{R}\}$ as the prior hyperparameters of each grammar rule r. The VBEM algorithm for PCFG can be summarized as follows.

1. **Initialization:** We initialize the posterior hyperparameters with the prior hyperparameters.

$$\mathbf{u}^{(0)} \quad = \quad \mathbf{u}^{\text{prior}} \tag{2.29}$$

where $\mathbf{u}^{(0)}$ is the initial posterior hyperparameters of each grammar rule. At the same time, the parameters of the model $\pi^{(0)}(r)$ is initialized in some way, such as randomization, uniform distribution, and biased preferences.

2. **VBE Step:** We precompute the inside and outside probabilities and then compute each latent variable by

$$q_i(A \to X_1 \ldots X_K) \quad = \quad \sum_{n_i^A \Rightarrow d^{X_1} \ldots d^{X_K}} \mathbf{f}(n_i^A) \prod_{k=1}^{K} \mathbf{e}(d^{X_k}) \tag{2.30}$$

where $\mathbf{e}(\cdot)$ and $\mathbf{f}(\cdot)$ are the inside and outside scores of a node, respectively. These scores can be computed by the standard Inside/Outside Algorithm.

3. **VBM Step:** We estimate the rule parameters by

$$\widehat{\pi}_{A \to X} \quad = \quad \frac{\exp \psi \left[u_{A \to X}^{\text{prior}} + \sum_{i=1}^{|\mathcal{D}|} \mathbb{E}_{q_i(A \to X)} \pi_{A \to X} \right]}{\exp \psi \left[\sum_{A \to X'} \left[u_{A \to X'}^{\text{prior}} + \sum_{i=1}^{|\mathcal{D}|} \mathbb{E}_{q_i(A \to X')} \pi_{A \to X'} \right] \right]} \tag{2.31}$$

where $\psi(\cdot)$ is the digamma function.

Steps 2 and 3 are iteratively applied until the posterior probability converges.

2.4 Evaluation Metrics

In the experiments, we show the capability of the system in recovering bracketings and dependency structures. We measured five aspects of accuracy: bracket recovery, crossing bracket rate, undirected dependency recovery, directed dependency recovery, and head recovery.

2.4.1 Bracket Recovery (BP, BR, and BF1)

Bracket recovery (Klein, 2005) is the measurement of phrase spans that can be recovered. We pay attention on this metric as it can show us the ability of the system in recovering brackets from scratch. Let us denote $span(\tau)$ the set of phrase spans of a tree τ, where:

$$span(\tau) \quad = \quad \{(i, j, w_{i,j}) | \text{each phrase in } \tau \text{ spans from } i \text{ to } j\} \tag{2.32}$$

and $w_{i,j}$ is a substring of the yields of τ from position i to position j. For example, let τ be the tree in Figure 4.3. We have that:

$$\begin{aligned} span(\tau) \quad = \quad & \{(1, 1, \mathsf{DT}), (2, 2, \mathsf{NN}), (3, 3, \mathsf{VBD}), (4, 4, \mathsf{JJ}), \\ & (5, 5, \mathsf{NNS}), (1, 2, \mathsf{DT\ NN}), (4, 5, \mathsf{JJ\ NNS}), \\ & (3, 5, \mathsf{VBD\ JJ\ NNS}), (1, 5, \mathsf{DT\ NN\ VBD\ JJ\ NNS})\} \end{aligned} \tag{2.33}$$

We measured this accuracy with three metrics: precision, recall, and F1 score. For a gold standard tree g and a test tree t, we calculate the three metrics with the following formulae.

$$\text{bracketing precision (BP)} \quad = \quad \frac{|span(g) \cap span(t)|}{|span(t)|} \tag{2.34}$$

$$\text{bracketing recall (BR)} \quad = \quad \frac{|span(g) \cap span(t)|}{|span(g)|} \tag{2.35}$$

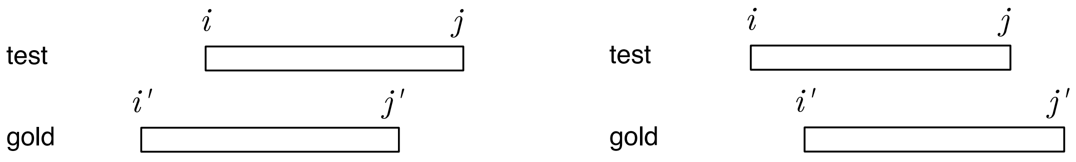
$$\text{bracketing F1 (BF1)} \quad = \quad \frac{2}{\text{BP}^{-1} + \text{BR}^{-1}} \tag{2.36}$$

Please note that these metrics are comparable to labeled bracketing accuracy (Klein, 2005).

2.4.2 Crossing Bracket Rate (CBR)

Crossing bracket rate (Klein, 2005) is the ratio of spans that cross any other spans in the gold standard tree. This metric is used as a supplement of the bracket recovery for showing that the recovered bracketings do not cross over any gold standard phrases. As illustrated in Figure 2.7, we say that a test span $(i, j, w_{i,j})$ crosses over a gold standard span $(i', j', w'_{i',j'})$ if either

1. $i' < i$, $j' < j$, and $i < j'$, or
2. $i < i'$, $j < j'$, and $i' < j$.



(a) Case 1

(b) Case 2

Figure 2.7: Crossing brackets

For a test tree t and a gold standard tree g, we can calculate the crossing bracket rate by the following formula.

$$\text{CBR} = \frac{|\{s \in span(t) | s \text{ crosses over at least one } s' \in span(g)\}|}{|span(t)|} \tag{2.37}$$

2.4.3 Undirected Dependency Recovery (UDP, UDR, and UDF1)

Undirected dependency recovery (Klein, 2005) is the measurement of dependency relations we can recover with the model, taking dependency direction out of consideration. We use this metric to roughly claim the ability of the system in recovering the majority of dependency relations from scratch. Let us denote $undep(\delta)$ the set of undirected dependency relations of a dependency structure δ, where:

$$undep(\delta) = \{\{(w_i, i), (w_j, j)\} | w_i \text{ has a dependency to } w_j \text{ in } \delta\} \tag{2.38}$$

and w_i and w_j are words at the positions i and j, respectively. It is noticeable that each element of the set is defined as a set of paired words, because the positions of words are not considered in this case. For example, let δ be the dependency structure in Figure 3.13(b). We obtain that:

$$\begin{aligned} undep(\delta) = \{&\{(\text{John}, 1), (\text{eats}, 2)\}, \{(\text{eats}, 2), (\text{sandwiches}, 4)\}, \\ &\{(\text{delicious}, 3), (\text{sandwiches}, 4)\}\} \end{aligned} \tag{2.39}$$

In our gold standard, we instead consider the dependency relations between parts of speech.

We measured this accuracy with three metrics: precision (UDP), recall (UDR), and F1 score (UDF1). For a gold standard dependency structure g and a test dependency structure t, we calculate the three metrics with the following formulae.

$$\begin{aligned}
\text{UDP} &= \frac{|undep(g) \cap undep(t)|}{|undep(t)|} && (2.40)\\
\text{UDR} &= \frac{|undep(g) \cap undep(t)|}{|undep(g)|} && (2.41)\\
\text{UDF1} &= \frac{2}{\text{UDP}^{-1} + \text{UDR}^{-1}} && (2.42)
\end{aligned}$$

2.4.4 Directed Dependency Recovery (DDP, DDR, and DDF1)

Directed dependency recovery (Klein, 2005) is similar to undirected dependency recovery, except that it also considers the dependency direction. We focus on this metric as it shows the capability of the system in recovering dependency relations from scratch. Let us denote $dep(\delta)$ the set of directed dependency relations of a dependency structure δ, where:

$$dep(\delta) = \{((w_i, i), (w_j, j)) | w_i \text{ has a dependency to } w_j \text{ in } \delta\} \quad (2.43)$$

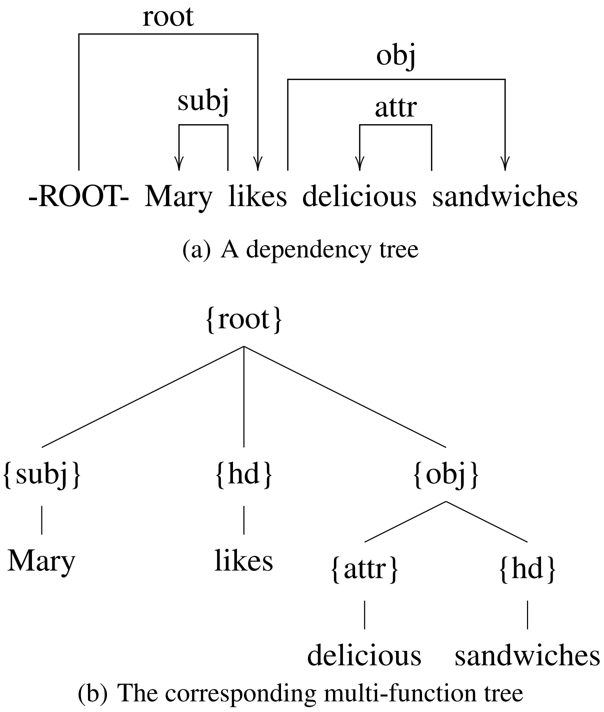
We say that w_i is a head that has a dependent w_j. We also say that w_j is a leftward dependent if $j < i$, and a rightward dependent if $i < j$. We can notice that each element is now defined as an ordered pair of a head and its dependent because the order of elements does matter in this case. For example, let δ be the dependency structure in Figure 3.13(b). We obtain that:

$$\begin{aligned}
dep(\delta) = \{&((\text{eats}, 2), (\text{John}, 1)), ((\text{eats}, 2), (\text{sandwiches}, 4)), && (2.44)\\
&((\text{sandwiches}, 4), (\text{delicious}, 3))\}
\end{aligned}$$

In our gold standard, we instead consider the dependency relations between PTB's parts of speech.

We measured this accuracy with three metrics: precision (DDP), recall (DDR), and F1 score (DDF1). For a gold standard dependency structure g and a test dependency structure t, we calculate the three metrics with the following formulae.

$$\begin{aligned}
\text{DDP} &= \frac{|dep(g) \cap dep(t)|}{|dep(t)|} && (2.45)\\
\text{DDR} &= \frac{|dep(g) \cap dep(t)|}{|dep(g)|} && (2.46)\\
\text{DDF1} &= \frac{2}{\text{DDP}^{-1} + \text{DDR}^{-1}} && (2.47)
\end{aligned}$$



(a) A dependency tree

(b) The corresponding multi-function tree

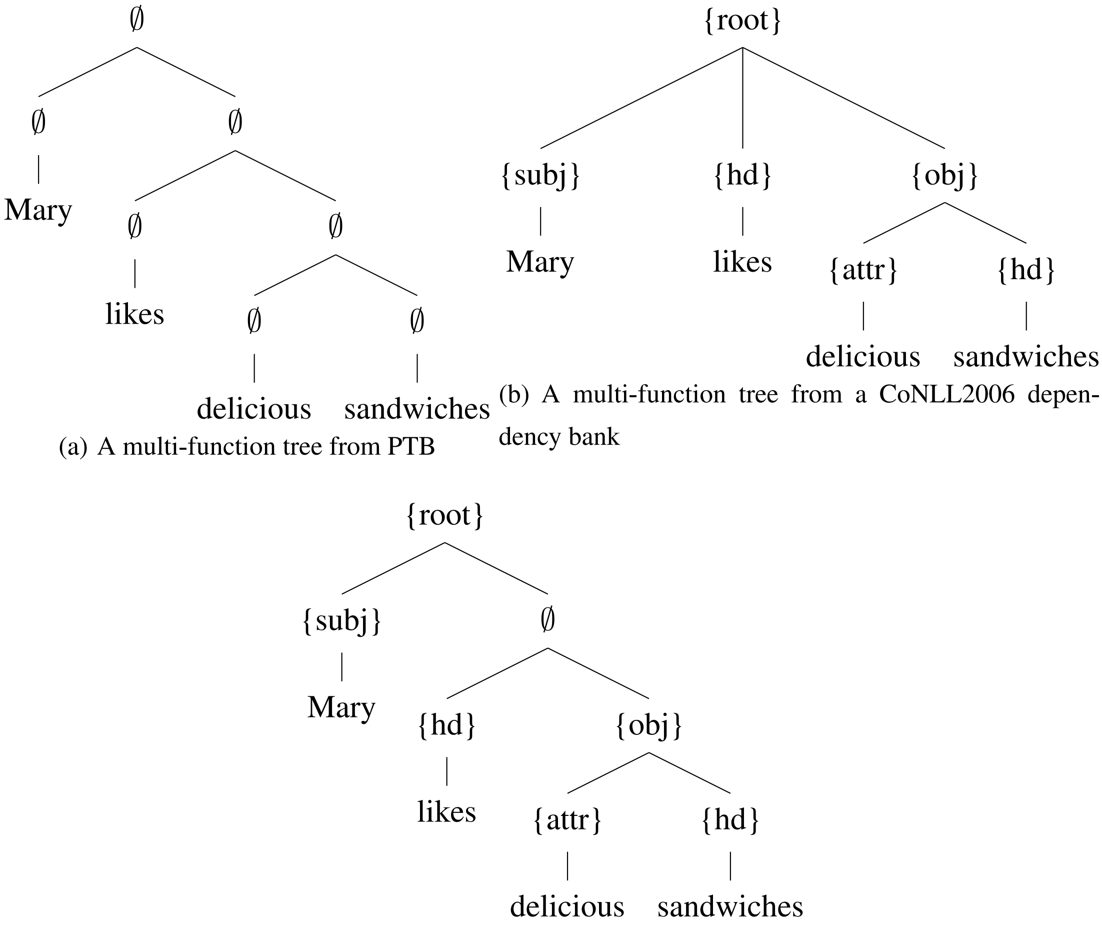
Figure 2.8: An example of conversion to a multi-function tree

2.4.5 Tree Edit Distance Evaluation Metric (TEDEVAL)

The aforementioned metrics are based on the assumption that all training and testing data must be annotated in the same scheme. It is, however, not the case in our experiments, where each dependency bank was annotated in a different annotation scheme. These discrepancies of dependency attachment conceal the true accuracies when measuring the recovery accuracy with the metrics.

One way to measure the accuracy of dependency recovery across different annotation theories is use the tree edit distance evaluation metric (TEDEVAL) (Tsarfaty et al., 2011; Tsarfaty et al., 2012a; Tsarfaty et al., 2012b). TEDEVAL has the following procedure.

Each input pair of parsed and gold trees are converted into multi-function trees which effectively handle annotation discrepancies. Each multi-function tree represents dependency labels by node labels. For each node in a multi-function tree, its label is denoted by a set of dependencies for the node, which can also be an empty set if the multi-function tree is converted from an input tree not annotated with dependency relations. For example, if an input tree is represented in the CoNLL2006's dependency form as illustrated in Figure 2.8(a), each dependency label is converted to a node label in the resulted function tree depicted in Figure 2.8(b).



(a) A multi-function tree from PTB

(b) A multi-function tree from a CoNLL2006 dependency bank

(c) The generalized multi-function tree

Figure 2.9: Generalization of multi-function trees

Each function tree pair are then generalized to find the most general tree that contains both of the input function trees. On each node of the generalized tree, there exists at least one node on both input trees that have the same spans as the node. For example, we can generalize the input trees in Figures 2.9(a) and 2.9(b) into the general multi-function tree in Figure 2.9(c).

The input trees are then compared with the generalized tree and edit distance is computed. There are two operations that are counted as edit: node addition and node deletion. An edit script $\mathrm{ES}(t_1, t_2)$ is an edit path for amending the tree t_1 to be the other one t_2. We seek to find the shortest edit script between t_1 and t_2; that is,

$$\mathrm{ES}^*(t_1, t_2) = \min_{\mathrm{ES}(t_1,t_2)} \sum_{e \in \mathrm{ES}(t_1,t_2)} \mathrm{cost}(e) \tag{2.48}$$

For example, as illustrated in Figure 2.9, $\mathrm{ES}^*(\text{PTB tree}, \text{generalized tree})$ is 0 (no edits), while $\mathrm{ES}^*(\text{CoNLL tree}, \text{generalized tree})$ is 1 (inserting a node for the verb phrase). The edit distance is measured by the function $\delta(t_{\text{parse}}, t_{\text{gold}}, t_{\text{gen}})$, where

$$\begin{aligned}\delta(t_{\text{parse}}, t_{\text{gold}}, t_{\text{gen}}) &= \text{cost}(\text{ES}^*(t_{\text{parse}}, t_{\text{gen}})) \\ &\quad -\text{cost}(\text{ES}^*(t_{\text{parse}}, t_{\text{gen}}) \cap \text{ES}^*(t_{\text{gold}}, t_{\text{gen}}))\end{aligned} \tag{2.49}$$

The accuracy is reported in terms of the average of the edit distances across the entire dataset, the value ranging from 0 (worst) to 1 (best).

$$\text{score}(\mathcal{D}_{\text{parse}}, \mathcal{D}_{\text{gold}}, \mathcal{D}_{\text{gen}}) = 1 - \frac{\sum_{i=1}^{|\mathcal{D}_{\text{parse}}|} \delta(t^{(i)}_{\text{parse}}, t^{(i)}_{\text{gold}}, t^{(i)}_{\text{gen}})}{\sum_{i=1}^{|\mathcal{D}_{\text{parse}}|} |t^{(i)}_{\text{parse}}| + |t^{(i)}_{\text{gen}}|} \tag{2.50}$$

where $|t|$ is the total number of nodes in the tree t.

2.5 Datasets

2.5.1 Dependency Banks

We compare our method with other state-of-the-art techniques, most of which are assessed using the Wall Street Journal part of Penn Treebank (Marcus et al., 1993). We use WSJ10, WSJ15, and WSJ20, the standard collection of trees whose sentence lengths do not exceed 10, 15, and 20 words, respectively, after eliminating punctuation marks and empty elements. We automatically convert PTB into dependency structures with the LTH Conversion Tool (Johansson and Nugues, 2007; Surdeanu et al., 2008).[3] The program is trained and tested using POS tag sequences from WSJ10, 15 and 20 as the terminal symbols (rather than strings of words) to minimize data sparsity.

For multilingual experiments, we use available dependency corpora from the CoNLL-X Shared Task 2006 (Buchholz and Marsi, 2006) including Danish [DA] (Kromann et al., 2003), Dutch [DU] (van der Beek et al., 2002), Portuguese [PO] (Afonso et al., 2002), and Swedish [SV] (Nilsson et al., 2005), all of which are Indo-European. To investigate grammar induction in other language families, we also evaluate our method against Arabic [AR] (Smrž et al., 2002), Bulgarian [BU] (Simov et al., 2001), Chinese [CH] (Keh-Liann and Hsieh, 2004), Czech [CZ] (Bohomovà et al., 2001), German [DE] (Brants et al., 2002), Japanese [JA] (Kawata and Bartels, 2000), Slovene [SL] (Džeroski et al., 2006), Spanish [ES] (Civit and Martí, 2004), and Turkish [TU] (Oflazer et al., 2003). We follow the same data preparation procedure as for WSJ, where punctuation marks are taken out and only POS tag sequences are used instead of word strings.

[3] Configuration: `-splitSlash=false -qmod=true -deepenQP=true -whAsHead=true`.

Table 2.1: Ten fields of CoNLL-X Shared Task 2006's data format

Number	Field	Description
1	ID	Token counter, starting at 1 for each sentence
2	FORM	Word form or punctuation symbol
3	LEMMA	Lemma or stem
4	CPOSTAG	Coarse-grain POS tag
5	POSTAG	Fine-grain POS tag
6	FEATS	Unordered set of syntactic/morphological features, separated by a vertical bar '|'
7	HEAD	ID of the head of the current token
8	DEPREL	Dependency relation to HEAD
9	PHEAD	ID of the projective head of the current token
10	PDEPREL	Dependency relation to PHEAD

2.5.2 Data Format

All datasets used in this thesis adhere to the CoNLL-X Shared Task 2006's rules. Each data file contains sentences separated by a blank line. Each sentence consists of one or more words represented as a tab-separated row of the ten fields described in Table 2.1, each of which strictly containing no space or blank characters. In the case where there is no information in a field, a placeholder '_' is put into the field to avoid mistaken field separation. The data files must be encoded in UTF-8 (Unicode).

2.5.3 Tagset Conversion for Dependency Banks

Since verb transitivity distinction in the tagset can improve the accuracy of parsing, it is useful to distinguish them in our corpora when possible. As to be described in Chapter 3, we reduce the ambiguity of the tagset by fusing CoNLL-X's CPOSTAG and POSTAG fields to become fine-grained tags and reclustering them. For all inflected languages (Arabic, Bulgarian, Czech, Slovene, and Turkish in this thesis), we recluster the tags with respect to the morphological attributes and assign new tags for the resulting groups. The procedure of assigning new tags to the corpora is summarized as follows.

Arabic Since noun declension plays an important role in Arabic, we have to distinguish the grammatical cases. We retag each noun of tags N and Z with respect to its grammatical case. If it is in case 1 (nominative), the tag becomes Nnom. If the case is 2 (genitive/accusative), the tag becomes Nga. If the case is 4 (genitive), the tag becomes Ngen. The same criteria apply to all pronouns of type S; resulting in the new tags Snom, Sga, and Sgen. The three tags of punctuation marks G, X, and – are also eliminated.

Bulgarian In Bulgarian, nouns are distinguished by their genders but this is rather harmful in parsing due to the problem of data sparsity. We neutralize the genders of noun (tag N), adjective (tag A), and hybrid (tag H); for example, we convert tags Nm, Nf, and Nn to a simpler tag N. We combine the adverbs Dm, Dt, Dl, Dq, and Dd as the tag D. We combine both types of numerals: cardinal Mc and ordinal Mo, as the tag M. Verbs of tags Vxi, Vyp, and Vii are regrouped as the tag COP (copula). The remaining verbs are clustered by their transitivity into tags intransitive Vi and transitive Vt.

Chinese The Chinese dependency bank makes use of a large fine-grained tagset. Due to the problem of data sparsity, we regroup them with the following criteria. All conjunctions of tags starting with C are combined into a simpler tag C. All adverbs of tags starting with Da are converted to QUANT. All adverbs of tags starting with Db are retagged as MODAL. All adverbs of tag Dc is renamed to NEG. All adverbs of tag Dk is retagged as SMOD. The remaining adverbs of tags starting with Dd, Df, Dg, Dh, Di, and Dj are simply retagged as ADV. The possessive marker of tag DE-Di is shortened to DE. All nouns of tags beginning with Na, Nb, Nc, and Nv are tagged as NN. All nouns of tags starting with Nd are retagged as TIME. All nouns of tags starting with Ne are retagged as determiner DET. All nouns of tags starting with Nf are retagged as classifier CL. All nouns beginning with Nh are retagged as pronoun PRO. All nouns of tag Ng is recategorized as a postposition POST. All prepositions of tags beginning with P are retagged as PREP. All types of particles annotated with tags starting with T are reduced to a simpler T. All verbs starting with VA, VB, VG, and VH are retagged as intransitive VI. All verbs starting with VC, VI, and VJ are tagged as transitive VT. All verbs of tags beginning with VD are retagged as ditransitive VD. All verbs of tags beginning with VF are reduced to VF. All verbs of tags starting with VE, VK, and VL are retagged as complex verb VCOMP. Uncategorized verbs of tags V_11, V_12, and V_2 are retagged as verb V.

Czech The Czech dependency bank makes use of a large tagset augmented with fine-grained morphological attributes. We eliminate the morphological attributes except the grammatical cases of nouns whereby we regroup nouns and adjectives according to them. Any noun (tag N) or adjective (tag A) tags are simply attached with a grammatical case. For example, if the case is 1 (nominative), the noun tag becomes N1. A wildcard grammatical case X is also attached to the tag if present, such as an adjective with a wildcard case is retagged as AX. Tags of unknown words X−x and X−@ are reduced to X.

Danish We keep the original tagset as is but we eliminate all punctuations of tag X.

Dutch We attach the morphological attribute of the conjunctions and prepositions to the original tags. For example, a conjunction (tag Conj) with a morphological attribute onder (subordinate) is retagged as Conj−onder. Each verb of tag V is also retagged according to the morphological attribute if it determines the transitivity or complexity (either intransitive intrans, transitive trans, or auxiliary/copulative hulpofkopp), e.g. a verb with intrans is retagged as intransitive V−intrans.

English The following punctuation mark tags are eliminated: ,, ., :, ″, “, −LRB−, −RRB−, $, #, and −NONE−.

German We rename the named entity tag from NNE to NE. The following punctuation mark tags are eliminated: $(, $,, and $..

Japanese We keep the tagset as is but we eliminate the only punctuation tag −−.

Portuguese We distinguish the syntactic categories of the independent pronoun (tag pron−indp) by its morphological attribute. We retag such pronoun with <quant> as adv−pron−indp, <rel> as pron−rel, and <dem> as pron−dem.

Slovene Generally, we abbreviate each tag name for easier reference. We also elaborate the pronouns with respect to their grammatical cases expressed in the morphological attribute. For example, if the case is genitive, the pronoun is retagged as ProGen. If the case is locative, the pronoun is retagged as ProLoc. Moreover, we retag every kind of verbs that are in the participial form (participle) with the gerund tag Ger.

Spanish We lowercase and reduce two tags: the abbreviations which start with Y and the numerals that begin with Z. We also eliminate all punctuation marks tagged with F.

Swedish We eliminate all punctuation marks annotated with the tags starting with `I`.

Turkish We retag all adjectives `Adj` that exhibit any adpositional relation (`With`, `Without`, `FitFor`, `InBetween`, `JustLike`, and `Rel`) as postposition `Postp`. We also retag all adverbs `adv` that exhibit any subordinating relation (`AsIf`, `AfterDoingSo`, `ByDoingSo`, `SinceDoingSo`, `Since`, `When`, `While`, and `WithoutHavingDoneSo`) as subordinate conjunction `AdvSubconj`. We rename the adverbs which exhibit the relation `Ly` with the tag `Advz`. Finally all personal pronouns `Pron` − `PersP` that show the genitive case (`Gen`) are retagged with `PronPersGen`.

By the above procedure, we obtain the corpora for the later experiments. The statistics for our corpora is listed in Table 2.2. It is worth remarking that this conversion may result in slight change in the accuracy when applied to the existing techniques.

2.6 Summary

We have explained the task of grammar induction which is the target of this thesis. We have reviewed the state of the art for grammar induction, and argued in favor of a new approach which (1) is prototype-driven and linguistically motivated, and (2) correctly captures frequent linguistic phenomena without the distortion of frequent collocation. We have explained how we prepare our datasets, i.e. dependency banks, for accuracy assessment of our technique.

We have developed the idea of computational preliminaries and mathematics that underly this research. We have introduced the theoretical concept of statistical modeling and explained the statistical techniques for grammar induction by means of this concept. We also described some of the successful models for grammar induction such as CCM, DMV, and DMV+CCM. We have also explained the evaluation metrics for accuracy assessment of grammar induction. In the next part, we describe our methodology in this research.

Table 2.2: Statistics of the corpora

Language	Length 10		Length 15		Length 20		Tags
	Sents	Words	Sents	Words	Sents	Words	
Arabic	241	1,635	354	3,457	478	6,290	22
Bulgarian	6,097	41,146	9,216	82,416	11,298	120,555	33
Chinese	52,424	295,821	56,985	356,952	57,647	369,573	30
Czech	27,375	157,946	43,039	361,830	55,855	590,918	73
Danish	1,995	12,256	3,136	26,977	4,060	43,543	19
Dutch	6,844	44,162	9,527	78,740	11,339	111,121	27
English	7,422	52,248	15,922	163,715	25,523	336,556	36
German	13,473	78,506	22,039	189,712	29,287	319,219	50
Japanese	12,884	45,302	14,938	71,603	16,132	92,818	76
Portuguese	2,611	15,652	4,157	35,760	5,614	61,885	23
Slovene	807	5,246	1,220	10,640	1,477	15,181	32
Spanish	712	4,487	1,209	10,978	1,709	20,002	31
Swedish	3,889	26,026	6,808	63,925	8,951	102,134	29
Turkish	3,833	20,555	4,568	29,891	5,017	37,914	33

Part II

Methodology

Chapter 3

Language Parameterization

Outline

This chapter explains the main contribution of this thesis: our language parameterization, which is used throughout the thesis to improve the accuracy of unsupervised grammar induction. In the spirit of *Principles and Parameters* approach and the empirical approach, we first start by developing the motivation of this research particularly in the benefits of incorporating syntactic prototypes (prior syntactic knowledge of the language of interest) into unsupervised models. We then explain our alternative language parameterization method which is linguistically motivated and easy to elicit by direct consultation with grammar compendiums or by interview with naïve informants in Section 3.1. Since our language parameters are designed to capture frequent word orders, each language parameter is sorted with respect to their frequency in natural language grammars. Section 3.2 explains the dialog we use to elicit such prior knowledge by interview with naïve informants. Finally, Section 3.3 explains how we encode the acquired prior knowledge into the syntactic prototype.

3.1 Language Parameters

3.1.1 Overview

To embrace the use of word orders as the language parameters in the syntactic prototype, we have to focus on three important factors of design. First, we have to maximize the cross-linguistic coverage of the syntactic prototype by cherry-picking the most frequent word order schemes with great impact. Second, we have to maximize the feasibility of eliciting these language parameters from various sources ranging from syntax textbooks to informants of different levels of linguistic literacy. Third and last, we have to convert the acquired language parameters into a compact syntactic prototype which is used for grammar induction. By these reasons, we devise a linguistic questionnaire which allows linguistic experts to code language parameters on their own, as well as facilitates a short interview with naïve informants.

Our linguistic questionnaire is divided into two parts: language parameters and a mapping table from the tagset to cross-linguistically frequent category classes. The first part regards a very rough overview of word orders in the language; e.g. the order of subject and predicate, etc. Frequent word orders, as our useful assistance, are well studied in The *World Atlas of Language Structures* (Haspelmath et al., 2005). We compiled the book's chapters 81–138 and 143–144 — each of which describing frequent word orders, phrase structures, and clause structures — and classified these parameters into eight categories in Table 3.1. The first part of the linguistic prototype is the linguistic typology of the language. Generalized for cross-linguistic coverage, it composes of four types of information:

1. **Word order:** the majority of the information is of this type, such as the orders of subject and predicate, verb and its arguments, noun and nominal modifiers, etc.

2. **Omissibility:** such as allowance of subject or object drop.

3. **Insertion:** such as allowance of adverb insertion between verb and its arguments.

4. **Transformation:** such as allowance of transforming a gerundial phrase into a noun phrase or an adverb.

This information is encoded as the questionnaire in Appendix A. Without a glance on the corpus, linguists can presume this knowledge based on their intuition.

Hypothetical trend of the accuracy: As per our intention to capture frequent word orders, we sort the eight categories in Table 3.1 with respect to their frequency of occurrence. That is to say, Group 1 are assumed to be found more frequently in natural language grammars while Group 8 are assumed to be less frequent word orders. We hypothesize that the more language parameters we incorporate into the syntactic prototype, the more accuracy we obtain. Because of their frequencies, the hypothetical

Table 3.1: An overview of the language parameters. The numbers on the right are the number of parameters of each group.

Groups	Parameters	No.
1	Basic word order: subject + verb + object + indirect object / free word order	1
2	Subject- and object-control verbs	2
3	Adjectives, adverbs, and auxiliary verbs	4
4	Cardinal numbers and noun classifiers	2
5	Adpositions, nominal modifiers, adverbials, possessive markers	7
6	Gerunds, infinitive markers, nominalizers, and sentential modifiers	6
7	Particles and the existence of copula	5
8	Usage of gerunds, negative markers, the use of dative shift, and the omission of subject and object	6

trend of the accuracy is: the accuracy rises rapidly and starts to saturate as we increase the number of language parameters in the syntactic prototype.

These language parameter values are used to automatically generate an initial grammar, called a *syntactic prototype*. Such grammar defines dependency rules and word order for certain cross-linguistically frequent category classes; namely, intransitive, transitive, and ditransitive verbs, subject- and object-control verbs, adjective, adverb, preposition, relative pronoun, gerund, copula, subordinate conjunction, noun classifier, infinitive marker, and cardinal number. The two parts are associated by rough linguistic classification of the tagset. To be more precise, the categories generated from the typology knowledge are grouped into several classes. Then each class is then mapped to the corresponding POS tagset specific to each corpus.

Our approach is an integration of Chomsky's (1965) renowned *Principles and Parameters* theory (P&P) and the empirical approach for induction of hierarchical linguistic structures. We prescribe a pseudo-Universal Grammar that copes with a small amount of frequent grammar rules with the linguistic questionnaire. The empirical approach then validates the elicited language parameters and empirically infers latent exceptions that are not elicited by the questionnaire from the data.

According to the notion of Principles and Parameters, our syntactic prototype can also be seen as a pseudo-Universal Grammar. Our language parameters are equivalent to the P&P's parameters that control the generation of a syntactic prototype. Our encoding algorithm for syntactic prototypes is equivalent to the P&P's principles where the algorithm determines how to generate a lexicalized grammar from the elicited language parameters. Each resulting syntactic prototype is equivalent to the syntactic competence for the language of interest.

3.1.2 Design of Questionnaire

We divide the questionnaire up to eight questions which cover a vast array of word orders, phrase structures, and clause structures. Each question is designed based on the statistics of frequent syntactic structures in The *World Atlas of Language Structures*. Since the questionnaire is based on word order information, the languages that have fixed word orders are easy to encode. Nevertheless, some other languages are not as easy to be parameterized in this fashion because their word orders are more flexible.

Syntactically speaking, languages are classified into rigid order languages and flexible order languages. Any rigid-order language has a fixed word order because all other word orders are ungrammatical, relatively infrequent, or for peculiar pragmatic uses (Dryer, 2011j). On the other hand, each flexible-order language syntactically allows most or all possible word orders, although some of them are dominant. We focus on capturing fixed or dominant word orders for a language.

The structure of our questionnaire is organized as follows. Question 1 studies the order of subject, verb, direct object, and indirect object. Question 2 investigates the use of simple modifiers such as adjectives and adverbs. Question 3 studies the use of complex verbs, e.g. subject- and object-control verbs. Question 4 describes the word orders of complex modifiers such as prepositions and relative clauses. Question 5 studies the use of gerunds in the language. Question 6 learns the use of subordinate clauses. Question 7 explores two transformational affixes; i.e. the infinitive marker and the nominalizing prefixes. Finally, Question 8 studies the existence of dative shift and the dropping.

Question 1: Sentence

As illustrated in Figure 3.1, Question 1 studies the word order of subject, verb, and object in the language regardless of the use of case markers (Dryer, 2011j; Dryer, 2011i;

Q1: What are *canonical* word-ordering patterns of the subject (S), the verb (V), the direct object (O), and the indirect object (I) in your language?

□ Tick here if you consider that your language rather has fixed word orders.

□ Tick here if there exists a notion of ditransitive verb in your language. Also tick the dominant word orders in the following table. (For example, English's word order is SVIO.)

□ SVOI	□ VSOI	□ SOVI	□ OVSI	□ VOSI	□ OSVI
□ SVIO	□ VSIO	□ SOIV	□ OVIS	□ VOIS	□ OSIV
□ SIVO	□ VISO	□ SIOV	□ OIVS	□ VIOS	□ OISV
□ ISVO	□ IVSO	□ ISOV	□ IOVS	□ IVOS	□ IOSV

□ Otherwise, tick here if there doesn't exist a notion of ditransitive verb in your language. Also tick the dominant word orders in the following table.

□ SOV	□ SVO	□ VSO
□ OSV	□ OVS	□ VOS

□ Otherwise, tick here if you consider that your language strictly has free word order. That means *all* the word orders in the above table are allowed.

Figure 3.1: Questions for sentence structure

Dryer, 2011g). Languages are first parameterized by the existence of indirect object (Haspelmath, 2011a) and the rigidity of the word order. If we assume the existence of indirect object, we can enumerate all possible word orders in the first part of Question 1. Otherwise, the enumeration of subject, verb, and object are listed in the second part. Flexible word order languages with no dominant word orders can also be identified in the last check box.

Question 2: Simple Modifiers

As shown in Figure 3.2, Question 2 is divided into four parts. The first part is designed based the studies of the word orders of adjective, demonstrative, and core noun (Dryer, 2011a; Dryer, 2011e; Dryer, 2011d; Dryer, 2011m) which elaborate the word order of an adjective and a core noun. The second part is designed for studying the word order of adverb and verb phrase. The third part studies the word order of adverb and adjective. Finally, the fourth part, based on the studies of negation (Haspelmath, 2011a; Miestamo, 2011b; Miestamo, 2011a), enumerates all possible patterns of negation when used with verb, adjective, and adverbs.

Q2.1: What is the word order of the adjectives when they combine with a noun?

□ Tick here if you consider that your language allows the adjectives to combine with nouns. Also tick the allowable word orders in the following table. (For example, English allows Adj+N.)

□ Adj+N □ N+Adj

□ Otherwise, tick here if you consider that your language does not allow the adjectives to combine with the nouns.

Q2.2: What is the word order of the adverbs when they combine with a verb phrase?

□ Tick here if you consider that your language allows the adverbs to combine with verb phrases. Also tick the allowable word orders in the following table. (For example, English allows both Adv+VP and VP+Adv.)

□ Adv+VP □ VP+Adv

□ Otherwise, tick here if you consider that your language does not allow the adverbs to combine with the verb phrases.

Question 2.3: What is the word order of the adverbs when they combine with an adjective?

□ Tick here if you consider that your language allows the adverbs to combine with adjectives. Also tick the allowable word orders in the following table. (For example, English allows Adv+Adj.)

□ Adv+Adj □ Adj+Adv

□ Otherwise, tick here if you consider that your language does not allow the adverbs to combine with the adjectives.

Question 2.4: What is the word order of the negators (Neg) when they combine with a verb (V), an adjective (Adj), and an adverb (Adv)?

□ Tick here if there exists a notion of negators in your language. Also tick the allowable word orders in the following table. (For example, English allows Neg+V, Neg+Adj, and Neg+Adv.)

□ Neg+V	□ V+Neg
□ Neg+Adj	□ Adj+Neg
□ Neg+Adv	□ Adv+Neg

□ Otherwise, tick here if there does not exist the notion of negators in your language.

Figure 3.2: Questions for simple modifiers

Question 3: Complex Verbs

Question 3, as illustrated in Figure 3.3, studies the existence of complex verbs, modal verbs, and copulae. In this thesis, a verb is said to be *complex* if it functionally takes another verb or verb phrase as an argument. Complex verbs include subject-control and object-control verbs, because they take a complementing verb phrase. This definition combines the notion of periphrastic causative verbs (Song, 2011b; Song, 2011a) and the '*want*' construction (Haspelmath, 2011b). However, this treatment of nonperiphrastic causative verbs is rather arguable. If the corpus of interest is annotated in the morphological level where the verb stem is separated from the causative marker, the causative marker is considered a complex verb. Otherwise, the causativized verb is canonically treated as one unit of verb.

Question 4: Complex Modifiers

As shown in Figure 3.4 and Figure 3.5, Question 4 entails the usage of complex modifiers; namely, adposition, possessivizer, relative pronoun, sentential modifier, sentential particle, and noun classifier. Question 4.1 is based on the studies of adposition in (Dryer, 2011b; Dryer, 2011n). Question 4.2 is a modification of the word order of genetive and noun phrase in (Dryer, 2011f). Question 4.3 elaborates the use of relative pronoun as explored in (Dryer, 2011h; Dryer, 2011o). Question 4.4 and Question 4.5, based on the study of (Dryer, 2011l; Dryer, 2011k), identify the use of sentential modifier (i.e. an adverbial word or phrase which modifies a sentence) and sentential particle (i.e. single word modifying a sentence). In Question 4.4 we also extend the notion of modifiers to adjectival, adverbial and gerund modifiers. Finally Question 4.6, designed based on (Gil, 2011), studies the usage of noun classifier and numeral in the language.

Question 5: Gerunds

Question 5 is devoted for the usage of gerundials. Based on English, this question characterizes gerundial units by three functions: a noun phrase, a noun modifier, and a predicative adverbial. These functions are explained in Figure 3.6. This parameter becomes prominent in languages in which the copula is used as a progressive auxiliary such as English 'to be'. From our 14 languages of interest, only English and Bulgarian explicitly make extensive use of the copula and gerund throughout the corpora.

Q3.1: Does there exist a notion of copulae in your language?

□ Tick here if it does.

□ Otherwise, tick here if it does not.

Q3.2: What is the word order of the modal verbs when they combine with a verb phrase?

□ Tick here if you consider that there is the notion of modal verbs in your language. Also tick the allowable word orders in the following table. For example, English allows Modal+VP.

□ Modal+VP

□ VP+Modal

□ Otherwise, tick here if the modal verbs don't exist in your language.

Q3.3: What are canonical word orders of the subject (S), the intransitive complex verb (V), and the complementing verb phrase (C)?

□ Tick here if there exists the notion of intransitive complex verbs in your language. Also tick the allowable word orders in the following table, for example, English allows SVC. You can also treat the serial verb construction as this complex verb. For example, Thai allows SVC.

□ SVC
□ SCV
□ CSV
□ VSC
□ VCS
□ CVS

□ Otherwise, tick here if there doesn't exist the notion of intransitive complex verbs in your language.

Q3.4: What are canonical word orders of the subject (S), the transitive complex verb (V), the object (O), and the complementing verb phrase (C)?

□ Tick here if there exists the notion of transitive complex verbs in your language. Also tick the allowable word orders in the following table. For example, English allows SVOC. You can also treat the serial verb construction as this complex verb. For example, Thai allows SVOC.

□ SVOC
□ SVCO
□ SCVO
□ CSVO
□ VSOC
□ VSCO
□ VCSO
□ CVSO
□ SOVC
□ SOCV
□ SCOV
□ CSOV
□ OVSC
□ OVCS
□ OCVS
□ COVS
□ VOSC
□ VOCS
□ VCOS
□ CVOS
□ OSVC
□ OSCV
□ OCSV
□ COSV

□ Otherwise, tick here if there doesn't exist the notion of transitive complex verbs in your language.

Figure 3.3: Questions for complex verbs

Question 4.1: What is the word order of the prepositions/postpositions in your language?

□ Tick here if there exists the notion of prepositions/postpositions in your language. Tick the allowable word orders in the following table. (For example, English allows Prep+NP.)

□ Prep+NP (preposition) □ NP+Post (postposition)

□ Otherwise, tick here if there doesn't exist the notion of prepositions/postpositions in your language.

Question 4.2: What is the word order of the owner (Owner), the possessivizer (Poss), and the ownee (Ownee) in your language?

□ Tick here if there exists the notion of prepositions/postpositions in your language. Also tick the allowable word orders in the following table. (For example, English allows the pattern Owner+Poss+Ownee.)

□ Owner+Ownee+Poss □ Ownee+Owner+Poss □ Owner+Poss+Ownee

□ Ownee+Poss+Owner □ Poss+Owner+Ownee □ Poss+Ownee+Owner

□ Otherwise, tick here if there doesn't exist the notion of possessivizers in your language.

Question 4.3: What is the word order of the relative pronoun (Relpro) and the complementing verb phrase (VP), and that of the relative clause (Relcls) and the core noun phrase (NP) in your language?

□ Tick here if there exists the notion of relative pronouns in your language. Tick the allowable word orders in the following table.

□ Relpro+VP □ VP+Relpro

□ Otherwise, tick here if there doesn't exist the notion of relative pronouns in your language.

Figure 3.4: Questions for complex modifiers

Question 4.4: What is the word order of the modifiers (*Mod) in your language?

□ Tick here if there exists the notion of adjectival modifiers in your language. Tick the allowable word orders in the following table.

□ NP+NMod □ NMod+NP

□ Tick here if there exists the notion of adverbial modifiers in your language. Tick the allowable word orders in the following table.

□ VP+VMod □ VMod+VP

□ Tick here if there exists the notion of gerund modifiers in your language. Tick the allowable word orders in the following table.

□ Gerund+GMod □ GMod+Gerund

□ Tick here if there exists the notion of sentential modifiers in your language. Tick the allowable word orders in the following table.

□ Sent+SMod □ SMod+Sent

□ Otherwise, tick here if there doesn't exist the notion of sentential modifiers in your language.

Question 4.5: What is the word order of the sentence (Sent) and the sentential particle (Part) in your language?

□ Tick here if there exists the notion of sentential particles in your language. Tick the allowable word orders in the following table. (For example, English allows Part+Sent.)

□ Sent+Part □ Part+Sent

□ Otherwise, tick here if there doesn't exist the notion of sentential particles in your language.

Question 4.6: Do you use noun classifiers in your language?

□ Tick here if you use noun classifiers to count things in your language. Tick the allowable word orders in the following table. (CL = noun classifiers)

□ Num+CL □ CL+Num

And what is it used as in your language?

□ Adjective □ Adverb □ Noun modifier □ VP modifier

□ Otherwise, tick here if you don't use noun classifiers in your language.

Figure 3.5: Questions for complex modifiers (cont'd).

Q5: Can a gerund, a transformation of a verb phrase, perform the following functions?

□ A noun phrase.

□ A noun modifier. Also tick the allowable word orders in the following table. (For example, English allows NP+Gerund.)

□ NP+Gerund □ Gerund+NP

□ A predicative adverbial. Also tick the allowable word orders in the following table. (For example, English allows VP+Gerund.)

□ VP+Gerund □ Gerund+VP

□ Otherwise, tick here if there doesn't exist the notion of gerunds in your language.

Figure 3.6: Question for gerunds

Q6: What is the word order for the main clause (Main), the subordinate conjunction (Conj), and the subordinate clause (Subcls) in your language?

□ Tick here if there exists the notion of subordinate conjunctions in your language. Also tick the allowable word orders in the following table. (For example, English allows Main+Conj+Subcls and Conj+Subcls+Main.)

□ Main+Subcls+Conj □ Subcls+Main+Conj

□ Main+Conj+Subcls □ Subcls+Conj+Main

□ Conj+Main+Subcls □ Conj+Subcls+Main

□ Otherwise, tick here if there doesn't exist the notion of subordinate conjunctions in your language.

Figure 3.7: Questions for subordinate conjunctions

Question 6: Subordinate Conjunctions

Question 6 entails the usage of subordinate conjunctions as studied in (Dryer, 2011c; Cristofaro, 2011d; Cristofaro, 2011b; Cristofaro, 2011c). Examples of subordinate conjunction include 'if', 'because', 'while', and 'when'. All possible combinations are listed in Figure 3.7.

Question 7: Transformational Affixes

As shown in Figure 3.8, Question 7 studies two kinds of transformational affixes: i.e. infinitive marker and nominalization affix. Question 7.1 entails the usage of purpose clause (Cristofaro, 2011a) expressed as an infinitival. Question 7.2 characterizes two

Question 7.1: What is the word order for the infinitive marker (Inf) and the verb phrase (VP) in your language?
□ Tick here if there exists the notion of infinitive markers in your language. Also tick the allowable word orders in the following table. (For example, English allows Inf+VP.)

□ Inf+VP □ VP+Inf

□ Otherwise, tick here if there doesn't exist the notion of infinitive markers in your language.
Question 7.2: What is the word order for the nominalizing affixes?
□ Tick here if the nominalizing affixes (Nom) can combine with noun phrases (NP). Also tick the allowable word orders in the following table. (For example, Thai allows Nom+NP.)

□ Nom+NP □ NP+Nom

□ Tick here if the nominalizing affixes (Nom) can combine with verb phrases (VP). Also tick the allowable word orders in the following table. (For example, Thai allows Nom+VP.)

□ Nom+VP □ VP+Nom

□ Otherwise, tick here if there doesn't exist the notion of nomializing affixes in your language.

Figure 3.8: Questions for transformational affixes

kinds of nominalizing affixes: NP nominalizer and VP nominalizer.

Question 8: Relocation and Dropping

As shown in Figure 3.9, Question 8 explains two linguistic phenomena: dative shift and dropping. Question 8.1 regards the definition of dative shift in (Steedman, 2000; Baldridge and Kruijff, 2003) where the beneficial of a ditransitive verb is shifted when the direct object is long. In Question 8.2, we extend the analysis of zero pronouns in (Siewierska, 2011) to the direct and indirect objects.

3.2 Language Parameter Elicitation

3.2.1 Interview Dialog

In the design of our linguistic questionnaire, we take into account the ease of parameter elicitation whereby language parameters are obtained by direct consultation with grammar books and linguistic experts as well as personal interviews with naïve informants and indirect observation from machine translation. It would however be laborious and

Question 8.1: Is dative shift allowed in your language?

☐ Yes.

☐ No.

☐ I don’t know.

Question 8.2: Can you drop out the following parts of the sentence if the context is clear enough?

☐ Subject.

☐ Object.

☐ Indirect object.

☐ None of these.

Figure 3.9: Questions for relocation and dropping

time-consuming to formalize the results of the interview with the naïve informants according to the questionnaire. Indirect observation from machine translation would be rather confusing and unsystematic, as translation pairs may be selected from different sources across languages. To control the source of translation, we devise a dialog for parameter elicitation from naïve informants and machine translation as shown in Figure 3.10 and Figure 3.11.

The dialog for parameter elicitation is designed based on the linguistic questionnaire where both correspond to each other, question against question. In each question, a linguistic unit (either an example sentence or phrase) is given alongside its pattern to observe. An informant (or, less preferably, a machine translation system) is asked to translate such linguistic unit into his own language and provide word alignment between the source and target languages.

3.2.2 Quantification of Human Labor

In our experiments, we first consult the grammar compendiums such as the World Atlas of Language Structures for default language parameters for our 14 languages of interest. Then we confirm such elicited parameters by either short interviews with naïve informants or machine translation.

Machine translation is our supplementary source of language parameters in the case where native speakers for a language are scarce. Most modern machine translation systems are phrase-based, therefore equivalent to a probabilistic finite-state transducer. Although our grammars we endeavor to induce are as expressive as context-free grammars, we are still able to elicit a correct set of hidden language parameters by using

[Question 1] Translate the sentence *Mary gives John a flower* (pattern: [$_\text{S}$ Mary] [$_\text{V}$ gives] [$_\text{I}$ John] [$_\text{O}$ a flower]). Does he have to rephrase it as *Mary gives a flower to John* (or something equivalent) instead?

[Question 2.1] Translate the phrase *small kittens* (pattern: [$_\text{Adj}$ small] [$_\text{N}$ kittens]).

[Question 2.2] Translate the sentence *Mary sits quietly* (pattern: [$_\text{V}$ sits] [$_\text{Adv}$ quietly]).

[Question 2.3] Translate the phrase *strongly bitter tea* (pattern: [$_\text{Adv}$ strongly] [$_\text{Adj}$ bitter]).

[Question 2.4] Translate the following phrases/sentences: (1) *The car does not work* (pattern: [$_\text{Neg}$ not] [$_\text{V}$ work]); (2) *a not complex exercise* (pattern: [$_\text{Neg}$ not] [$_\text{Adj}$ complex]); (3) *not strongly bitter tea* (pattern: [$_\text{Neg}$ not] [$_\text{Adv}$ strongly]).

[Question 3.1] Translate the sentences: (1) *John is a student*; (2) *John is tall*; (3) *John is in the classroom*. Is there anything equivalent to the verb *to be*?

[Question 3.2] Translate the sentence *Mary can swim* (pattern: [$_\text{Modal}$ can] [$_\text{V}$ swim]).

[Question 3.3] Translate the sentence *Mary wants to swim* (pattern: [$_\text{V}$ want] [$_\text{C}$ swim]).

[Question 3.4] Translate the sentence *John asks Mary to hold the door for him* (pattern: [$_\text{V}$ ask] [$_\text{O}$ Mary] [$_\text{C}$ hold the door]).

[Question 4.1] Translate the following phrases/sentences: (1) *a gift in the box* (pattern: [$_\text{Prep}$ in] [$_\text{NP}$ the box]); (2) *Mary walks into the classroom* (pattern: [$_\text{Prep}$ into] [$_\text{NP}$ the room]).

[Question 4.2] Translate the phrase *John's car* (pattern: [$_\text{Owner}$ John] [$_\text{Poss}$'s] [$_\text{Ownee}$ car]). Also ask the informant if he can directly say that or he has to rephrase it as *a car of John's* (or something equivalent) instead.

[Question 4.3] Translate the sentence *John lifts the box that contains many books* (pattern: [$_\text{NP}$ box] [$_\text{Relpro}$ that] [$_\text{VP}$ contains many books]).

Figure 3.10: Dialog for indirect parameter elicitation via translation

[Question 4.4] Translate the sentences: (1) *John is the man on the bench* (pattern: [$_{\text{NP}}$ man] [$_{\text{NMod}}$ on the bench]); (2) John walks on the shore (pattern: [$_{\text{VP}}$ walk] [$_{\text{VMod}}$ on the shore]); (3) John is the man running on the shore (pattern: [$_{\text{Gerund}}$ running] [$_{\text{GMod}}$ on the shore]); (4) *On Monday, John will hand in his homework* (pattern: [$_{\text{SMod}}$ On Monday] [$_{\text{S}}$ John will hand in his homework]).

[Question 4.5] Ask the informant if there are any adverb-like words which seem to modify the verb, as in *over* in *Mary starts the process <u>over</u>* (pattern: [$_{\text{VP}}$ start the process] [$_{\text{Part}}$ over]).

[Question 4.6] Translate the phrase *three cars*. Does he have to rephrase it as *three bodies of car* (pattern: [$_{\text{Num}}$ three] [$_{\text{CL}}$ bodies] [$_{\text{NP}}$ car])?

[Question 5] Translate the following phrases/sentences: (1) *Running is good* ([$_{\text{Gerund}}$ running] as a noun phrase); (2) *a running man* ([$_{\text{Gerund}}$ running] as an adjectival); (3) *John is running* ([$_{\text{Gerund}}$ running] as a non-finite verb). Check if any of these is grammatical in the language.

[Question 6] Translate the following sentences: (1) *If you press this button, the door will open* (pattern: [$_{\text{Conj}}$ if] [$_{\text{Subcls}}$ you press this button] [$_{\text{Main}}$ the door will open]); (2) *The door will open if you press this button* (pattern: [$_{\text{Main}}$ the door will open] [$_{\text{Conj}}$ if] [$_{\text{Subcls}}$ you press this button]).

[Question 7.1] Translate the sentence *Mary carefully reads her draft to identify the inconsistency* (pattern: [$_{\text{Inf}}$ to] [$_{\text{VP}}$ identify the inconsistency]).

[Question 7.2] Ask the informant if: (1) there are any bound morphemes that transform a verb into a noun phrase such as *travel* > *traveler* (pattern: [$_{\text{VP}}$ travel] [$_{\text{Nom}}$ -er]); (2) there are any bound morphemes that augment the meaning of a noun, such as the Thai bound morpheme *nák* in *tennis* 'tennis' > *nák tennis* 'tennis player' (pattern: [$_{\text{Nom}}$ nák] [$_{\text{NP}}$ tennis]). Note that each bound morpheme does not have any meaning on its own.

[Question 8.1] Translate the sentence *John introduces to Mary his long-time friends from high school* (pattern: [$_{\text{Dative}}$ to Mary] [$_{\text{O}}$ his long-time friends from high school]). Also ask the informant if he has to relocate the dative part to a particular position if the direct object is elongated.

[Question 8.2] Translate the sentence *Mary gives John a flower* (pattern: pattern: [$_{\text{S}}$ Mary] [$_{\text{V}}$ gives] [$_{\text{I}}$ John] [$_{\text{O}}$ a flower]) and consider the grammaticality of the following omissions: (1) (*She*) *gives John a flower*; (2) *Mary gives* (*him*) *a flower*; (3) *Mary gives John* (*it*).

Figure 3.11: Dialog for indirect parameter elicitation via translation (cont'd)

example sentences simple enough to correctly cover them.

Short interviews of this kind were held with Arabic, Chinese, English, Japanese, and German native speakers. Due to scarcity of native speakers at the time of experiments, syntactic prototypes for the remaining languages were obtained from sentences generated from English by automatic machine translation. Machine translation, of course, provides only the 1-best translation and word alignment according to its model, and is likely to be less accurate than human informants. Google Translate was used to provide translation and word alignment for Bulgarian, Czech, Danish, Dutch, Portuguese, Slovene, Spanish, Swedish, and Turkish.

Once the word alignment is acquired in either way, we can indicate default word orders by analyzing the word orders of the target language. This process normally takes up to two to four hours per previously unseen language.

Once we obtain the language parameters, we then study the POS annotation guidelines for each language and mapped each tag to one or more language-specific category classes by Table 3.2. For example, a mapping table from English Penn Treebank POS tagset to our cross-linguistic tagset is shown in Table 3.3. We also analyze the guidelines to determine if the head of the coordinate structures is the first conjunct or the conjunction itself with respect to the annotation scheme. To thoroughly scrutinize the usage of each POS tag and assign them to appropriate classes it typically takes around four to six hours. It therefore takes six to ten hours to build a syntactic prototype for each language.

3.3 Encoding of Syntactic Prototypes

Once we can characterize an input language with our language parameters, it is now useful to elaborate how we convert them into a computable syntactic prototype. This section begins with our backbone grammar formalism, Categorial Dependency Grammar, which is a kind of lexicalized grammar. Then we explain how to algorithmically transform each parameter into syntactic categories for the lexical inventory.

3.3.1 Categorial Dependency Grammar

For encoding our syntactic prototypes, we extend Categorial Grammar (CG) (Ajdukiewicz, 1935; Bar-Hillel, 1953) with headedness resulting in Categorial Dependency Grammar (CDG), where its syntactic derivations define constituency and dependency in parallel.

Table 3.2: Mapping table between our cross-linguistic tagset and the corpus-specific tagset

Generalized Tagset	Corpus-specific Tags
Noun (`n`)	
Adjective (`adj`)	
Nominal modifier (`nmod`)	
Verb (`v`)	
Intransitive verb (`vi`)	
Transitive verb (`vt`)	
Ditransitive verb (`vd`)	
Complex verb (`vcomp`)	
Complex intransitive verb (`vicomp`)	
Complex transitive verb (`vtcomp`)	
Modal verb (`modal`)	
Copula (`copula`)	
Gerund (`gerund`)	
Adverb (`adv`)	
Particle (`part`)	
Adverbial modifier (`vmod`)	
Sentential modifier (`smod`)	
Gerund's modifier (`gmod`)	
Preposition/postposition (`adposition`)	
Relative pronoun (`relpro`)	
Conjunction (`conj`)	
Subordinate conjunction (`subconj`)	
Classifier (`cl`)	
substituting adjective (`adjcl`)	
substituting adverb (`advcl`)	
substituting nominal modifier (`nmodcl`)	
substituting adverbial modifier (`vmodcl`)	
Possessive marker (`poss`)	
Infinitive marker (`inf`)	
NP nominalizer (`npnom`)	
VP nominalizer (`vpnom`)	
Negator (`neg`)	
Verb phrase (`vp`)	
Adpositional phrase (`pp`)	

Table 3.3: Mapping table for English Penn Treebank. Unused rows are not shown.

Generalized Tagset	**Corpus-specific Tags**
Noun (`n`)	CD, DT, EX, FW, JJ, JJR, JJS, NN, NNPS, NNS, PRP, PRP$, WP
Adjective (`adj`)	CD, DT, FW, JJ, JJR, JJS, LS, NN, NNPS, NNS, PDT, PRP$, VBG, VBN, WDT, WP$
Nominal modifier (`nmod`)	POS
Verb (`v`)	VB, VBD, VBP, VBZ, VBG, VBN
Modal verb (`modal`)	MD
Copula (`copula`)	VB, VBD, VBP, VBZ, VBG, VBN
Gerund (`gerund`)	VBG, VBN
Adverb (`adv`)	LS, RB, RBR, RBS
Particle (`part`)	RP, TO
Sentential modifier (`smod`)	CC, LS, RB, RBR, RBS, UH
Preposition/postposition (`adposition`)	FW, IN, TO
Relative pronoun (`relpro`)	WDT
Conjunction (`conj`)	CC, SYM
Subordinate conjunction (`subconj`)	IN, WRB
Possessive marker (`poss`)	POS
Infinitive marker (`inf`)	TO
Verb phrase (`vp`)	MD

CDG is based on the notion of headedness in the slashes similar to predicate-argument dependencies in CCG (Hockenmaier, 2003a; Clark and Curran, 2007) and PF-CCG (Koller and Kuhlmann, 2009). CDG and PF-CCG differ from each other in that CDG's attachment directions can be customized while PF-CCG's ones are predetermined by the syntactic categories. Headedness of CDG and CCG are both customizable, but CDG's headedness is more limited than that of CCG. This is because CDG does not allow CCG's head-passing mechanism which is required for unbounded dependencies.

To make this point clear, let us go over some basic concepts of CG and we will introduce the dependency-enhanced version afterwards. In the original CG, a constituent (i.e. a phrase) is considered as the combination of a function and its zero or more arguments. For example, a transitive verb 'eat' performs as a function, and a noun phrase 'sandwiches' as its argument. These words are combined to form a verb phrase.

Each constituent is assigned one or more syntactic categories, which are represented as either an atomic category or a complex category. For example, we can assign 'John' as an atomic category np, which of course represents a noun phrase. A complex category can be denoted by X/Y and $X\backslash Y$, where X and Y are any syntactic categories. A phrase of category X/Y can combine with another phrase of category Y on the right side to become a larger phrase of category X. On the other hand, a phrase of category $X\backslash Y$ can combine with another phrase of category Y on the left side to become a larger phrase of category X. The construction of linguistic constituents can also be defined by the notion of functions and arguments as shown below.

$$\begin{aligned} X/Y \qquad Y &\Rightarrow X \\ Y \qquad X\backslash Y &\Rightarrow X \end{aligned} \tag{3.1}$$

Thus the intransitive verb 'sleep' can be assigned the category $\mathsf{s}\backslash\mathsf{np}$, meaning that it requires a noun phrase np on the left side to form a sentence s.

For example, a simplified English grammar is given below.

$$\begin{aligned} \text{John, sandwiches} &\vdash \mathsf{np} \\ \text{eats} &\vdash (\mathsf{s}\backslash\mathsf{np})/\mathsf{np} \end{aligned} \tag{3.2}$$

The notation $\vdash$ denotes a lexical entry where we assign word forms listed on the left side to have the syntactic categories specified on the right side. The syntactic derivation of the sentence 'John eats sandwiches' is illustrated Fig 3.12.

Categorial grammar can be extended to construct the dependency structure in parallel to the constituency structure. One approach is to incorporate the headedness into

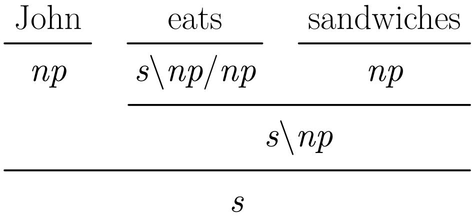


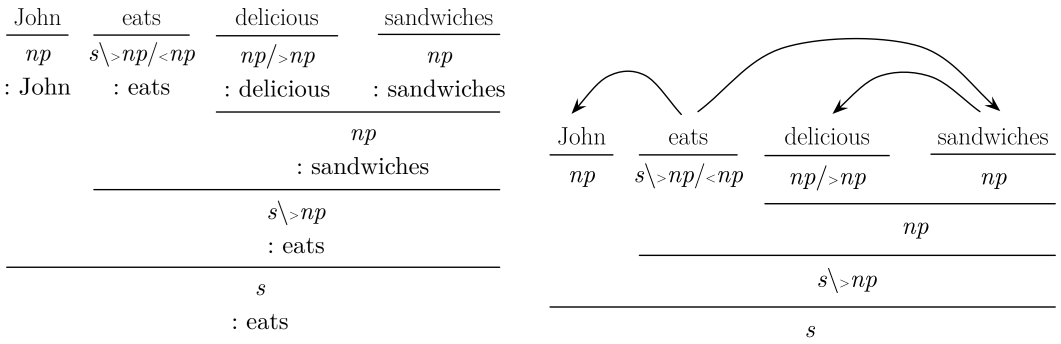
Figure 3.12: The syntactic derivation of ‘John eats sandwiches’ based on categorial grammar

the slashes. The notation $<$ is annotated if the dependency is to be linked from the head of the right-side constituent to that of the left-side constituent, while the notation $>$ is annotated for the opposite attachment. These attachment directions are similar to that of head-outward dependency structure (Collins, 1999) except that our directions point toward the head instead of the dependent. For example, an adjective (say ‘big’) can be assigned with the category $\mathsf{np}/_{>}\mathsf{np}$ so that when it combines with a noun phrase (say ‘books’), the head of the resulting constituent is the noun phrase. Likewise, a transitive verb (such as ‘eats’) can be assigned the category $\mathsf{s}\backslash_{>}\mathsf{np}/_{<}\mathsf{np}$ so that when it combines with its object, the head of the verb phrase is still the verb. We henceforward call this extension as *categorial dependency grammar* (CDG). The derivation rules for CDG are formulated in Eq (3.3).

$$\begin{array}{llcl}
X/_{<}Y : d_1 & Y : d_2 & \Rightarrow & X : h(d_1) \\
X/_{>}Y : d_1 & Y : d_2 & \Rightarrow & X : h(d_2) \\
Y : d_1 & X\backslash_{<}Y : d_2 & \Rightarrow & X : h(d_1) \\
Y : d_1 & X\backslash_{>}Y : d_2 & \Rightarrow & X : h(d_2)
\end{array} \tag{3.3}$$

where d_1 and d_2 are dependency structures, and $h(d)$ means that the head of the newly constructed constituent is d. For instance, the first rule specifies that the head of the result constituent X is on the left side (i.e. d_1) and the dependent is on the right side (i.e. d_2). Let us consider the combination of a transitive verb $\mathsf{s}\backslash_{>}\mathsf{np}/_{<}\mathsf{np}$ (such as ‘eats’) and a noun phrase np (such as ‘sandwiches’). We obtain that the head of the verb phrase is the verb and the dependent is the noun phrase.

Compared with CDG, PF-CCG’s dependency direction always points towards the function; i.e. slash categories of PF-CCG can always be rewritten as $X/_{<}Y$ and $X\backslash_{>}Y$ in CDG. According to this, an adjective would have category $\mathsf{np}/_{<}\mathsf{np}$, resulting in the combination of the adjective and a noun phrase having the head as the adjective. This is contrary to linguistic intuition where the core noun is considered the head of a noun



(a) Dependency-driven derivation

(b) Equivalent dependency structure (each arrow is drawn from a head to its dependent)

Figure 3.13: Syntactic derivation of 'John eats delicious sandwiches' based on *categorial dependency grammar*. The syntactic head of each constituent is denoted by a colon. Each arrow is drawn from the head word to the dependent word.

phrase.

Although the headedness is incorporated to the syntactic categories, we have not introduced any additional application rules to the original categorial grammar. CDG has the same expressive power as categorial grammar and CFG does not generate unbounded or *nonprojective* dependency. Therefore, we neither expand the search space of grammar induction nor compromise the expressive power with the data sparsity problem.

Let us extend the grammar in Eq (3.2) into a CDG version as follows.

$$\begin{aligned} \text{John, sandwiches} &\vdash \mathsf{np} \\ \text{delicious} &\vdash \mathsf{np}/_{>}\mathsf{np} \\ \text{eats} &\vdash (\mathsf{s}\backslash_{>}\mathsf{np})/_{<}\mathsf{np} \end{aligned} \tag{3.4}$$

The syntactic derivation of the sentence 'John eats delicious sandwiches' is illustrated in Figure 3.13. Figure 3.13(a) shows dependency-driven derivation, in which the heads of constituents are propagated. Figure 3.13(b) reflects the formation of the dependency structure corresponding to the dependency-driven derivation. The dependency direction in this figure starts from the head and points towards the dependent.

In the experiments, conjunctions, such as 'and' and 'or,' are assigned with the schematic category $X\backslash_{<}X/_{<}X$, where X is any category. It means that we settle the head of the coordinate structure on the left conjunct. This annotation scheme is also used in CCGbank (Hockenmaier, 2003a) and C&C Parser (Clark and Curran, 2007). For example, the head of the coordinate structure ‘sandwiches and bananas’ is ‘sandwiches.’[1]

A syntactic prototype generated from the language parameters may not be capable of handling less frequent linguistic phenomena, resulting in unparsable sentences. We additionally define the wildcard category $\star$ which combines any syntactic categories and produces the wildcard itself as follows.

$$\begin{aligned} \star : d_1 \quad X : d_2 \quad &\Rightarrow \quad \{\star : h(d_1) \leftarrow h(d_2), \\ &\qquad \star : h(d_1) \rightarrow h(d_2)\} \\ X : d_1 \quad \star : d_2 \quad &\Rightarrow \quad \{\star : h(d_1) \leftarrow h(d_2), \\ &\qquad \star : h(d_1) \rightarrow h(d_2)\} \end{aligned} \tag{3.5}$$

The wildcard is assigned to unknown words and large constituents to complete the parses of an unparsable sentence. This special category is used in the case where a sentence is unparsable by the lexicon. All unknown words and largest, non-governed constituents are assigned with the wildcard category ‘$\star$’ and the sentence is reparsed. An algorithm for CDG parsing and assigning wildcards will be elaborated in Chapter 4.

3.3.2 Construction of Lexicon Inventory

This section presents our method for constructing a syntactic prototype from the language parameters acquired by the linguistic questionnaire. Based on CDG, a syntactic prototype is in fact an inventory of lexical entries in which each terminal symbol (POS tag in this case) is assigned with one or more syntactic categories. As aforementioned in Section 3.2, we facilitate its maintenance by introducing the cross-linguistic categories that interlink the terminal symbols (POS tagset) and the syntactic categories automatically generated from the parameters.

The organization of the lexicon inventory closely follows this scheme. Syntactic categories for each cross-linguistic category are generated from a specific algorithm which takes one or more language parameters. First, basic categories and control flags of those algorithms are explained. The algorithms for generating syntactic categories are then illustrated and described. Finally, we incorporate all of them to build the lexicon inventory.

[1] For ease of development, we implement the conjunction as a separate category $\&$ instead of a schema. However, it retains the same semantics as its schematic version.

Table 3.4: Headedness flags for lexicon inventory construction. If a flag is set true, such category becomes the head; e.g. if h_{adj} is set true, the adjective becomes the head instead of the core noun.

Flag	Default	Flag	Default	Flag	Default
h_{adj}	$\mathbb{F}$	h_{adv}	$\mathbb{F}$	$h_{\mathrm{sub-phr}}$	$\mathbb{T}$
h_{nmod}	$\mathbb{F}$	h_{neg}	$\mathbb{F}$	$h_{\mathrm{poss-nmod}}$	$\mathbb{F}$
h_{verb}	$\mathbb{T}$	h_{modal}	$\mathbb{T}$	$h_{\mathrm{poss-phr}}$	$\mathbb{F}$
h_{copula}	$\mathbb{F}$	h_{adpos}	$\mathbb{T}$	h_{inf}	$\mathbb{T}$
h_{vmod}	$\mathbb{F}$	h_{relpro}	$\mathbb{T}$	h_{npnom}	$\mathbb{T}$
h_{gmod}	$\mathbb{F}$	h_{part}	$\mathbb{F}$	h_{vpnom}	$\mathbb{T}$
h_{smod}	$\mathbb{F}$	$h_{\mathrm{sub-smod}}$	$\mathbb{F}$	h_{cl}	$\mathbb{F}$

Basic Categories, Control Flags, and Word Order Parameters

As building blocks, there are four basic categories in our lexicon inventory: s for sentence, np for noun phrase, conj for conjunction, and num for numeral. It can be seen that the categories s and np alone can form a verb system such as intransitive, transitive, and ditransitive verbs by varying the number of np arguments. One can also form adjectives and adverbs based on the constructed verbs and a number of np's. More complex categories such as prepositions, relative pronouns, and subordinate conjunctions can then be hierarchically constructed from the previously constructed categories.

There are also 21 headedness flags as shown in Table 3.4 which is used to assign attachment direction of each category by the function $\mathtt{attdir}(\cdot)$. For example, if h_{adj} is set true, the adjective becomes the head instead of the core noun; otherwise the adjective becomes the dependent of the core noun. These flags are configurable to approximate the annotation scheme of the corpus of interest, where their default values are an approximation of the head percolation heuristics (Collins, 1999).

Once language parameters are elicited, there will be one or more word order parameters in each question. We will henceforward represent the set of word order parameters as Q_n, where n is an identifier of the question. For instance, $Q_{2.5}$ is the set of all word orders in Question 2.5.

Verb Systems

There are five kinds of verbs generated in the inventory: intransitive, transitive, ditransitive, intransitive complex, and transitive complex verbs.

Question 1 controls the generation of intransitive, transitive, and ditransitive verbs. Intransitive verbs are generated by the function $\texttt{vi}(Q_1)$ described in Algorithm 3.1. First we find d, the attachment direction of the intransitive verb, from h_{verb}. If h_{verb} is true, the verb becomes the head of the sentence; otherwise it does not. It however does not make any sense to change the value of h_{verb} as it is linguistically agreed that the verb is the head of the sentence. If the sentence word order is verb-medial (such as $\text{S}\underline{\text{V}}\text{IO}$) or verb-final ($\text{SIO}\underline{\text{V}}$), the subject must be on the left side; thus the category $\texttt{s}\backslash_d\texttt{np}$ is generated. Otherwise, the word order is verb-initial, such as $\underline{\text{V}}\text{SIO}$; in this case, the category $\texttt{s}/_d\texttt{np}$ is generated.

Algorithm 3.1 $\texttt{vi}(Q_1)$: generate the intransitive verbs.

```
let $d \leftarrow \texttt{attdir}(h_{\text{verb}})$    ▷ attachment direction of the intransitive verbs
for each word order parameter $q \in Q_1$ do
    if $q$ is verb-medial or -final then
        yield $\texttt{s}\backslash_d\texttt{np}$
    else if $q$ is verb-initial then
        yield $\texttt{s}/_d\texttt{np}$
    end if
end for
```

In Algorithm 3.2, transitive verbs are generated by the function $\texttt{vt}(Q_1)$. Transitive verbs are built upon intransitive verbs, the results of $\texttt{vi}(Q_1)$. If the word order is verb-final (such as $\text{SIO}\underline{\text{V}}$), the syntactic category $\texttt{s}\backslash_d\texttt{np}\backslash_d\texttt{np}$ is generated from the intransitive verb $\texttt{s}\backslash_d\texttt{np}$. Otherwise, the word order is either verb-initial (e.g. $\underline{\text{V}}\text{SIO}$) or verb-medial (e.g. $\text{S}\underline{\text{V}}\text{IO}$) and the syntactic categories $\texttt{s}/_d\texttt{np}/_d\texttt{np}$ and $\texttt{s}\backslash_d\texttt{np}/_d\texttt{np}$ are generated, respectively.

Ditransitive verbs are generated by the function $\texttt{vd}(Q_1)$ explained in Algorithm 3.3. Similarly, ditransitive verbs are built upon the results of $\texttt{vt}(Q_1)$. However, generating a ditransitive verb from a transitive verb differs from the previous verb algorithms in that we cannot directly attach an np argument to the transitive verb. We instead have to insert it to the transitive verb category. For example, if the word order is $\text{SVO}\underline{\text{I}}$, we first generate a transitive verb category $\texttt{s}\backslash_d\texttt{np}_\texttt{S}/_d\texttt{np}_\texttt{O}$, where each np is tagged with its syntactic role (S for subject and O for object). We have to insert an np argument into the category, resulting in $\texttt{s}\backslash_d\texttt{np}_\texttt{S}\underline{/_d\texttt{np}_\texttt{I}}/_d\texttt{np}_\texttt{O}$.

Complex verbs are built upon the intransitive and transitive verbs. Controlled by Question 3.3, intransitive complex verbs are generated by the function $\texttt{vicomp}(Q_{3.3})$.

Algorithm 3.2 vt(Q_1): generate the transitive verbs.

```
let $d \leftarrow$ attdir($h_{\text{verb}}$)    ▷ *attachment direction of the transitive verbs*
for each word order parameter $q \in Q_1$ do
  for each intransitive verb $x \in$ vi($Q_1$) do
    if $q$ is verb-final then
      yield $x\backslash_d$np
    else if $q$ is verb-initial or -medial then
      yield $x/_d$np
    end if
  end for
end for
```

Algorithm 3.3 vd(Q_1): generate the ditransitive verbs.

```
let $d \leftarrow$ attdir($h_{\text{verb}}$)    ▷ *attachment direction of the ditransitive verbs*
for each word order parameter $q \in Q_1$ do
  if $q$ allows ditransitivity of the verbs then
    for each transitive verb $x \in$ vt($Q_1$) do
      Insert the indirect object np into $x$ w.r.t. $q$
      yield $x$
    end for
  end if
end for
```

Algorithm 3.4 produces intransitive complex verbs by inserting a complementing intransitive verb to another intransitive verb. The transitive complex verbs are generated by the function vtcomp($Q_{3.4}$) controlled by Question 3.4. Likewise, Algorithm 3.5 produces transitive complex verbs by inserting a complementing intransitive verb to another intransitive verb.

Algorithm 3.4 vicomp($Q_{3.3}$): generate the intransitive complex verbs.

```
let $d \leftarrow \texttt{attdir}(h_{\text{verb}})$
for each word order parameter $q \in Q_{3.3}$ do
  for each intransitive verb $x \in \texttt{vi}(Q_1)$ do
    for each intransitive verb $y \in \texttt{vi}(Q_1)$ do
      Insert the complementing verb phrase $y$ into $x$ w.r.t. $q$
      yield $x$
    end for
  end for
end for
```

Algorithm 3.5 vtcomp($Q_{3.4}$): generate the transitive complex verbs.

```
let $d \leftarrow \texttt{attdir}(h_{\text{verb}})$
for each word order parameter $q \in Q_{3.4}$ do
  for each transitive verb $x \in \texttt{vt}(Q_1)$ do
    for each intransitive verb $y \in \texttt{vi}(Q_1)$ do
      Insert the complementing verb phrase $y$ into $x$ w.r.t. $q$.
      yield $x$
    end for
  end for
end for
```

All these verbs are combined in the function v($Q_1, Q_{3.3}, Q_{3.4}$) in Algorithm 3.6. This function returns all syntactic categories: intransitive verbs produced by vi(Q_1), transitive verbs by vt(Q_1), ditransitive verbs vd(Q_1), intransitive complex verbs by vicomp($Q_{3.3}$), and transitive complex verbs by vtcomp($Q_{3.4}$).

Algorithm 3.6 v($Q_1, Q_{3.3}, Q_{3.4}$): generate all the verbs.

```
return $\texttt{vi}(Q_1) \cup \texttt{vt}(Q_1) \cup \texttt{vd}(Q_1) \cup \texttt{vicomp}(Q_{3.3}) \cup \texttt{vtcomp}(Q_{3.4})$
```

Noun Modifiers

There are two types of noun modifiers: i.e. adjective and nominal modifier. The main use of the adjective is in the lexical level; i.e. this cross-linguistic category is mainly used by lexicons. On the other hand, the nominal modifier is mainly for internal use in generating the adjectivals such as preposition phrase and relative clause.

The adjectives are generated by the function $\texttt{adj}(Q_{2.1})$ as illustrated in Algorithm 3.7 according to the parameters in Question 2.1. Syntactic categories are straightforwardly generated from the adjective-noun word order. The nominal modifiers are generated by the function $\texttt{nmod}(Q_{4.4})$ described in Algorithm 3.8 following the parameters in Question 4.4. Likewise, syntactic categories are straightforwardly generated from the noun-nominal modifer word order.

Algorithm 3.7 $\texttt{adj}(Q_{2.1})$: generate the adjectives.

```
let $d \leftarrow \texttt{attdir}(h_{\text{adj}})$    ▷ attachment direction of the adjectives
for each word order parameter $q \in Q_{2.1}$ do
  if $q = \texttt{Noun} + \texttt{Adj}$ then
    yield $\texttt{np}\backslash_d\texttt{np}$
  else if $q = \texttt{Adj} + \texttt{Noun}$ then
    yield $\texttt{np}/_d\texttt{np}$
  end if
end for
```

Algorithm 3.8 $\texttt{nmod}(Q_{4.4})$: generate the nominal modifiers.

```
let $d \leftarrow \texttt{attdir}(h_{\text{nmod}})$    ▷ attachment direction of the nominal modifiers
for each word order parameter $q \in Q_{4.4}$ do
  if $q = \texttt{NP} + \texttt{NMod}$ then
    yield $\texttt{np}\backslash_d\texttt{np}$
  else if $q = \texttt{NMod} + \texttt{NP}$ then
    yield $\texttt{np}/_d\texttt{np}$
  end if
end for
```

Gerunds

There are three types of gerund to be generated; namely, gerundial noun phrase, gerundial noun modifier, and gerundial verb modifier.

The gerundial unit is a gerund that performs as a particular syntactic category; for instance, gerundial noun phrase performs as a noun phrase. Syntactic categories of the gerund noun phrase are generated from the function $\mathtt{npgerund}(Q_5)$ described in Algorithm 3.9 by replacing the innermost intransitive verb inside each verb with np. The same method applies to the generation of gerundial noun modifier and gerundial verb modifier by replacing the innermost intransitive verb with a nominal modifier and a verb modifier, respectively, as described in $\mathtt{nmodgerund}(Q_5)$ in Algorithm 3.10 and $\mathtt{vmodgerund}(Q_5)$ in Algorithm 3.11.

All gerunds are combined in the function $\mathtt{gerund}(Q_5)$ in Algorithm 3.12. This function returns all syntactic categories generated from $\mathtt{npgerund}(Q_5)$, $\mathtt{nmodgerund}(Q_5)$, and $\mathtt{vmodgerund}(Q_5)$.

Algorithm 3.9 $\mathtt{npgerund}(Q_5)$: generate the gerundial noun phrases.

```
for each word order parameter $q \in Q_5$ do
  for each verb $x \in \mathtt{v}(Q_1, Q_{3.3}, Q_{3.4})$ do
    if $q$ allows a gerund to be a noun phrase then
      Replace the innermost intransitive verb of $x$ with np.
      yield $x$
    end if
  end for
end for
```

Adverbials

There are six types of adverbials: adverb, verb modifier, gerund modifier, sentence modifier, negator, and modal. The adverb differs from the modifiers in that the adverb is used in the lexical level while the modifiers are used for internal use in generating several adverbial units such as preposition phrase.

Illustrated in Algorithm 3.13, syntactic categories for the adverb are straightforwardly generated in $\mathtt{adv}(Q_{2.2})$ that is controlled by the parameters from Question 2.2. Each adverb category is basically a complex category that takes an intransitive verb to generate another one. Generation of verb modifier, gerund modifier, and sentence mod-

Algorithm 3.10 `nmodgerund`(Q_5): generate the gerundial noun modifiers.

```
for each word order parameter $q \in Q_5$ do
  for each verb $x \in \mathtt{v}(Q_1, Q_{3.3}, Q_{3.4})$ do
    if $q$ allows a gerund to be a nominal modifier then
      for each nominal modifier $y \in \mathtt{nmod}(Q_{4.4})$ do
        Replace the innermost intransitive verb of $x$ with $y$.
        yield $x$
      end for
    end if
  end for
end for
```

Algorithm 3.11 `vmodgerund`(Q_5): generate the gerundial noun modifiers.

```
for each word order parameter $q \in Q_5$ do
  for each verb $x \in \mathtt{v}(Q_1, Q_{3.3}, Q_{3.4})$ do
    if $q$ allows a gerund to be a nominal modifier then
      for each verb modifier $y \in \mathtt{vmod}(Q_{4.4})$ do
        Replace the innermost intransitive verb of $x$ with $y$.
        yield $x$
      end for
    end if
  end for
end for
```

Algorithm 3.12 `gerund`(Q_5): generate all gerunds.

```
return $\mathtt{npgerund}(Q_5) \cup \mathtt{nmodgerund}(Q_5) \cup \mathtt{vmodgerund}(Q_5)$
```

ifier follows the same procedure but it is instead controlled by the parameters in Question 4.4, as shown in $\texttt{vmod}(Q_{4.4})$ in Algorithm 3.14, $\texttt{gmod}(Q_{4.4})$ in Algorithm 3.15, and $\texttt{smod}(Q_{4.4})$ in Algorithm 3.16, respectively. The attachment directions of these categories are determined by the control flags h_{adv}, h_{vmod}, h_{gmod}, and h_{smod}, respectively.

Algorithm 3.13 $\texttt{adv}(Q_{2.2})$: generate the verb modifiers.

```
let $d \leftarrow \texttt{attdir}(h_{\text{adv}})$    ▷ attachment direction of the adverb
for each word order parameter $q \in Q_{2.2}$ do
  for each intransitive verb $x \in \texttt{vi}(Q_1)$ do
    if $q = \texttt{VP} + \texttt{Adv}$ then
      yield $x\backslash_d x$
    else if $q = \texttt{Adv} + \texttt{VP}$ then
      yield $x/_d x$
    end if
  end for
end for
```

Algorithm 3.14 $\texttt{vmod}(Q_{4.4})$: generate the verb modifiers.

```
let $d \leftarrow \texttt{attdir}(h_{\text{vmod}})$    ▷ attachment direction of the verb modifier
for each word order parameter $q \in Q_{4.4}$ do
  for each intransitive verb $x \in \texttt{vi}(Q_1)$ do
    if $q = \texttt{VP} + \texttt{VMod}$ then
      yield $x\backslash_d x$
    else if $q = \texttt{VMod} + \texttt{VP}$ then
      yield $x/_d x$
    end if
  end for
end for
```

Syntactic categories for the negator are generated by the function $\texttt{neg}(Q_{2.4})$ in Algorithm 3.17. Controlled by the parameters in Question 2.4, we generate the categories in a fashion similar to $\texttt{adv}(Q_{2.2})$ but in this case we treat each negator as an adverb which modifies a verb, an adjective, an adverb, a noun modifier, and a verb modifier. Syntactic categories for the modals are generated with respect to the parameters in Question 3.2 by the function $\texttt{modal}(Q_{3.2})$ in Algorithm 3.18. The attachment

Algorithm 3.15 `gmod`$(Q_{4.4})$: generate the gerund modifiers.

```
let $d \leftarrow \texttt{attdir}(h_{\text{gmod}})$    ▷ attachment direction of the gerund modifier
for each word order parameter $q \in Q_{4.4}$ do
    for each gerund $x \in \texttt{gerund}(Q_5)$ do
        if $q = \texttt{Gerund} + \texttt{GMod}$ then
            yield $x\backslash_d x$
        else if $q = \texttt{GMod} + \texttt{Gerund}$ then
            yield $x/_d x$
        end if
    end for
end for
```

Algorithm 3.16 `smod`$(Q_{4.4})$: generate the sentential modifiers.

```
let $d \leftarrow \texttt{attdir}(h_{\text{smod}})$    ▷ attachment direction of the sentential modifier
for each word order parameter $q \in Q_{4.4}$ do
    if $q = \texttt{S} + \texttt{SMod}$ then
        yield $\texttt{s}\backslash_d\texttt{s}$
    else if $q = \texttt{SMod} + \texttt{S}$ then
        yield $\texttt{s}/_d\texttt{s}$
    end if
end for
```

directions of these categories are determined by the control flags h_{neg} and h_{modal}, respectively.

Algorithm 3.17 $\texttt{neg}(Q_{2.4})$: generate the negators.

```
let $d \leftarrow \texttt{attdir}(h_{\text{neg}})$    ▷ attachment direction of the negators
for each word order parameter $q \in Q_{2.4}$ do
  if $q = \texttt{Neg} + \texttt{V}$ then
    yield $x/_d x$ for all verbs $x \in \texttt{v}(Q_1, Q_{3.3}, Q_{3.4})$
  else if $q = \texttt{V} + \texttt{Neg}$ then
    yield $x\backslash_d x$ for all verbs $x \in \texttt{v}(Q_1, Q_{3.3}, Q_{3.4})$
  else if $q = \texttt{Neg} + \texttt{Adj}$ then
    yield $x/_d x$ for all adjectives $x \in \texttt{adj}(Q_{2.1})$
  else if $q = \texttt{Adj} + \texttt{Neg}$ then
    yield $x\backslash_d x$ for all adjectives $x \in \texttt{adj}(Q_{2.1})$
  else if $q = \texttt{Neg} + \texttt{Adv}$ then
    yield $x/_d x$ for all adverbs $x \in \texttt{adv}(Q_{2.2})$
  else if $q = \texttt{Adv} + \texttt{Neg}$ then
    yield $x\backslash_d x$ for all adverbs $x \in \texttt{adv}(Q_{2.2})$
  else if $q = \texttt{Neg} + \texttt{NMod}$ then
    yield $x/_d x$ for all nominal modifiers $x \in \texttt{nmod}(Q_{4.4})$
  else if $q = \texttt{NMod} + \texttt{Neg}$ then
    yield $x\backslash_d x$ for all nominal modifiers $x \in \texttt{nmod}(Q_{4.4})$
  else if $q = \texttt{Neg} + \texttt{VMod}$ then
    yield $x/_d x$ for all verb modifiers $x \in \texttt{vmod}(Q_{4.4})$
  else if $q = \texttt{VMod} + \texttt{Neg}$ then
    yield $x\backslash_d x$ for all verb modifiers $x \in \texttt{vmod}(Q_{4.4})$
  end if
end for
```

It should be noted that these adverbial syntactic categories are designed for CDG which is as expressively powerful as CFG. Function composition, as seen in CCG, is therefore not taken into account at this stage.

Adpositions

We generate two kinds of adposition: preposition and postposition. Algorithm 3.19 and Algorithm 3.20 display the procedures of generating syntactic categories for preposi-

Algorithm 3.18 $\texttt{modal}(Q_{3.2})$: generate the modals.

```
let $d \leftarrow \texttt{attdir}(h_{\text{modal}})$    ▷ attachment direction of the modals
for each word order parameter $q \in Q_{3.2}$ do
    for each intransitive verb $x \in \texttt{vi}(Q_1)$ do
        if $q = \texttt{VP} + \texttt{Modal}$ then
            yield $x\backslash_d x$
        else if $q = \texttt{Modal} + \texttt{VP}$ then
            yield $x/_d x$
        end if
    end for
end for
```

tions in the function $\texttt{prep}(Q_{4.1})$ and postpositions in the function $\texttt{post}(Q_{4.1})$ with respect to Question 4.4. These categories are generated by taking either a noun modifier generated from $\texttt{nmod}(Q_{4.4})$, a verb modifier from $\texttt{vmod}(Q_{4.4})$, a gerund modifier from $\texttt{gmod}(Q_{4.4})$, or a sentence modifier from $\texttt{smod}(Q_{4.4})$ and combining it with an np argument. The collection of all syntactic categories of the prepositions and the postpositions are produced by the function $\texttt{adpos}(Q_{4.1})$ in Algorithm 3.21.

Algorithm 3.19 $\texttt{prep}(Q_{4.1})$: generate all prepositions.

```
let $d \leftarrow \texttt{attdir}(h_{\text{adpos}})$    ▷ attachment direction of the preposition
let $P \leftarrow \texttt{nmod}(Q_{4.4}) \cup \texttt{vmod}(Q_{4.4}) \cup \texttt{gmod}(Q_{4.4}) \cup \texttt{smod}(Q_{4.4})$
for each word order parameter $q \in Q_{4.1}$ do
    for each preposition phrase $p \in P$ do
        if $q = \texttt{Prep} + \texttt{NP}$ then
            yield $p/_d\texttt{np}$
        end if
    end for
end for
```

Relative Pronouns

Syntactic categories for the relative pronouns are generated by the function $\texttt{relpro}(Q_{4.3})$ in Algorithm 3.22 with respect to Question 4.3. Piecewisely, relative pronouns are built upon the noun modifiers generated from $\texttt{nmod}(Q_{4.4})$ as the inner part and the argument

Algorithm 3.20 `post`$(Q_{4.1})$: generate all postpositions.

```
let $d \leftarrow$ attdir$(h_{\text{adpos}})$    ▷ attachment direction of the postposition
let $P \leftarrow$ nmod$(Q_{4.4}) \cup$ vmod$(Q_{4.4}) \cup$ gmod$(Q_{4.4}) \cup$ smod$(Q_{4.4})$
for each word order parameter $q \in Q_{4.1}$ do
  for each postposition phrase $p \in P$ do
    if $q =$ NP + Post then
      yield $p\backslash_d$np
    end if
  end for
end for
```

Algorithm 3.21 `adpos`$(Q_{4.1})$: generate all adpositions.

```
return prep$(Q_{4.1}) \cup$ post$(Q_{4.1})$
```

generated from `vi`(Q_1). The attachment direction is determined by the control flag h_{relpro}.

Algorithm 3.22 `relpro`$(Q_{4.3})$: generate all relative pronouns.

```
let $d \leftarrow$ attdir$(h_{\text{relpro}})$    ▷ attachment direction of the relative pronouns
for each word order parameter $q \in Q_{4.3}$ do
  for each nominal modifier $x \in$ nmod$(Q_{4.4})$ do
    for each intransitive verb $y \in$ vi$(Q_1)$ do
      if $q =$ Relpro + VP then
        yield $x/_d y$
      else if $q =$ VP + Relpro then
        yield $x\backslash_d y$
      end if
    end for
  end for
end for
```

Copulae

Syntactic categories for the copulae are generated from the function `copula`$(Q_{3.1})$ displayed in Algorithm 3.23. Each copula is a verb which takes a subject of the category `np` and a complement which can either be an adjective generated from `adj`$(Q_{2.1})$ or a noun modifier generated from $\texttt{nmod}(Q_{4.4})$. The procedure is controlled by the parameters in Question 3.1 and the attachment direction is determined by h_{copula}.

Algorithm 3.23 $\texttt{copula}(Q_{3.1})$: generate all corpulae.

```
let $d \leftarrow \texttt{attdir}(h_{\text{copula}})$    ▷ attachment direction of the copulae
for each word order parameter $q \in Q_{3.1}$ do
  if $q$ allows the existence of copula then
    for each intransitive verb $x \in \texttt{vi}(Q_1)$ do
      for each $y \in \texttt{adj}(Q_{2.1}) \cup \texttt{nmod}(Q_{4.4})$ do
        yield $x/_d y$
      end for
    end for
  end if
end for
```

Particles

Syntactic categories of the sentence particles are generated by the function $\texttt{part}(Q_{4.5})$ in Algorithm 3.24. Each particle is treated as an adverb which modifies any verb generated from $\texttt{v}(Q_1, Q_{3.3}, Q_{3.4})$. The attachment direction of the particles is determined by the control flag h_{part}.

Algorithm 3.24 $\texttt{part}(Q_{4.5})$: generate all particles.

```
let $d \leftarrow \texttt{attdir}(h_{\text{part}})$    ▷ attachment direction of the particles
for each word order parameter $q \in Q_{3.2}$ do
  for each verb $x \in \texttt{v}(Q_1, Q_{3.3}, Q_{3.4})$ do
    if $q = \texttt{VP} + \texttt{Part}$ then
      yield $x\backslash_d x$
    else if $q = \texttt{Part} + \texttt{VP}$ then
      yield $x/_d x$
    end if
  end for
end for
```

Subordinate Conjunctions

Syntactic categories of the subordinate conjunctions are generated by the function $\texttt{subconj}(Q_6)$ in Algorithm 3.25 according to the parameters in Question 6. Each subordinate conjunction joins the main clause and a subordinate clause which can be either a sentence of the category s, an adjective generated from $\texttt{adj}(Q_{2.1})$, or a noun modifier generated from $\texttt{nmod}(Q_{4.4})$. There are two control flags to control the attachment directions: $h_{\text{sub-phr}}$ and $h_{\text{sub-smod}}$. $h_{\text{sub-phr}}$ determines the attachment direction of the subordinate conjunction when combined with the complement. $h_{\text{sub-smod}}$, on the other hand, determines the attachment direction of the subordinate clause when combined with the main clause.

Algorithm 3.25 $\texttt{subconj}(Q_6)$: generate all subordinate conjunctions.

```
let $d \leftarrow \texttt{attdir}(h_{\text{sub-phr}})$    ▷ attachment direction of the subordinator
let $d' \leftarrow \texttt{attdir}(h_{\text{sub-smod}})$    ▷ attachment direction of the clause
for each $x \in \{\mathsf{s}\} \cup \texttt{adj}(Q_{2.1}) \cup \texttt{nmod}(Q_{4.4})$ do
  for each word order parameter $q \in Q_6$ do
    if $q = \texttt{Main} + \texttt{Conj} + \texttt{Subcls}$ then
      yield $\mathsf{s}\backslash_{d'}\mathsf{s}/_{d}x$
    else if $q = \texttt{Subcls} + \texttt{Conj} + \texttt{Main}$ then
      yield $\mathsf{s}/_{d'}\mathsf{s}\backslash_{d}x$
    else if $q = \texttt{Main} + \texttt{Subcls} + \texttt{Conj}$ then
      yield $\mathsf{s}\backslash_{d'}\mathsf{s}\backslash_{d}x$
    else if $q = \texttt{Subcls} + \texttt{Main} + \texttt{Conj}$ then
      yield $\mathsf{s}\backslash_{d}x\backslash_{d'}\mathsf{s}$
    else if $q = \texttt{Conj} + \texttt{Main} + \texttt{Subcls}$ then
      yield $\mathsf{s}/_{d}x/_{d'}\mathsf{s}$
    else if $q = \texttt{Conj} + \texttt{Subcls} + \texttt{Main}$ then
      yield $\mathsf{s}/_{d'}\mathsf{s}/_{d}x$
    end if
  end for
end for
```

Possessive Markers

Syntactic categories of the possessive markers are generated from the function $\texttt{poss}(Q_{4.2})$ in Algorithm 3.26. Each possessive marker associates two noun phrases of the category np: one as the owner and the other one as the ownee. The categories are generated according to the parameters in Question 4.2. There are two control flags to control the attachment directions: $h_{\mathrm{poss-phr}}$ and $h_{\mathrm{poss-nmod}}$. The flag $h_{\mathrm{poss-phr}}$ determines the attachment direction of the possesive marker when combined with the owner, while the flag $h_{\mathrm{poss-nmod}}$ determines the attachment direction of the genitive when combined with the ownee.

Algorithm 3.26 $\texttt{poss}(Q_{4.2})$: generate all possessive markers.

```
let $d \leftarrow \texttt{attdir}(h_{\mathrm{poss-phr}})$    ▷ attachment direction of the marker
let $d' \leftarrow \texttt{attdir}(h_{\mathrm{poss-nmod}})$    ▷ attachment direction of the genitive
for each word order parameter $q \in Q_{4.2}$ do
    if $q = \texttt{Ownee} + \texttt{Poss} + \texttt{Owner}$ then
        yield $\mathsf{np}\backslash_{d'}\mathsf{np}/_{d}x$
    else if $q = \texttt{Owner} + \texttt{Poss} + \texttt{Ownee}$ then
        yield $\mathsf{np}/_{d'}\mathsf{np}\backslash_{d}x$
    else if $q = \texttt{Ownee} + \texttt{Owner} + \texttt{Poss}$ then
        yield $\mathsf{np}\backslash_{d'}\mathsf{s}\backslash_{d}x$
    else if $q = \texttt{Owner} + \texttt{Ownee} + \texttt{Poss}$ then
        yield $\mathsf{np}\backslash_{d}x\backslash_{d'}\mathsf{np}$
    else if $q = \texttt{Poss} + \texttt{Ownee} + \texttt{Owner}$ then
        yield $\mathsf{np}/_{d}x/_{d'}\mathsf{np}$
    else if $q = \texttt{Poss} + \texttt{Owner} + \texttt{Ownee}$ then
        yield $\mathsf{np}/_{d'}\mathsf{np}/_{d}x$
    end if
end for
```

Infinitive Markers

Syntactic categories of the infinitive markers are generated by the function $\texttt{inf}(Q_{7.1})$ in Algorithm 3.27. In our syntactic prototype, any infinitive phrase is treated as a noun phrase of the category np. Each infinitive marker therefore takes an infinitive verb generated from $\texttt{vi}(Q_1)$ and converts it to an np. The flag h_{inf} determines the attachment direction of the infinitive marker when combined with an intransitive verb.

Algorithm 3.27 `inf`$(Q_{7.1})$: generate all infinitive markers.

```
let $d \leftarrow \mathtt{attdir}(h_{\mathrm{inf}})$    ▷ attachment direction of the infinitive markers
for each word order parameter $q \in Q_{7.1}$ do
    for each intransitive verb $x \in \mathtt{vi}(Q_1)$ do
        if $q = \mathtt{VP} + \mathtt{Inf}$ then
            yield $\mathsf{np}\backslash_d x$
        else if $q = \mathtt{Inf} + \mathtt{VP}$ then
            yield $\mathsf{np}/_d x$
        end if
    end for
end for
```

Nominalizers

There are two kinds of nominalizers: NP nominalizer and VP nominalizer. An NP nominalizer is a bound morpheme that combines with a noun phrase to become another noun phrase. On the other hand, a VP nominalizer combines with a verb phrase to become a noun phrase.

As seen in Algorithm 3.28, syntactic categories of NP nominalizers are generated by the function `npnom`$(Q_{7.2})$ according to the parameters in Question 7.2. The attachment direction is determined of the NP nominalizers by the control flag h_{npnom}. Meanwhile, syntactic categories of VP nominalizers are generated by the function `vpnom`$(Q_{7.2})$ according to the same set of parameters. The control flag h_{vpnom} determines the attachment direction of the VP nominalizers.

Algorithm 3.28 `npnom`$(Q_{7.2})$: generate all NP nominalizer.

```
let $d \leftarrow \mathtt{attdir}(h_{\mathrm{npnom}})$    ▷ attachment direction of the NP nominalizer
for each word order parameter $q \in Q_{7.2}$ do
    if $q = \mathtt{NP} + \mathtt{Nom}$ then
        yield $\mathsf{np}\backslash_d \mathsf{np}$
    else if $q = \mathtt{Nom} + \mathtt{NP}$ then
        yield $\mathsf{np}/_d \mathsf{np}$
    end if
end for
```

Algorithm 3.29 $\texttt{vpnom}(Q_{7.2})$: generate all verb nominalizers.

```
let $d \leftarrow \texttt{attdir}(h_{\text{vpnom}})$    ▷ attachment direction of the verb nominalizer
for each word order parameter $q \in Q_{7.2}$ do
  for each intransitive verb $x \in \texttt{vi}(Q_1)$ do
    if $q = \texttt{VP} + \texttt{Nom}$ then
      yield $\mathsf{np}\backslash_d x$
    else if $q = \texttt{Nom} + \texttt{VP}$ then
      yield $\mathsf{np}/_d x$
    end if
  end for
end for
```

Classifiers

Classifiers are categorized into four types: adjectival classifier, adverbial classifier, noun-modifying classifier, and verb-modifying classifier. The adjectival classifier is distinguished from its counterpart, noun-modifying classifier, in that when combined with a numeral, the numeral phrase becomes an adjective unit rather than a nominal modifier. The adverbial classifier is also distinguished from the verb-modifying classifier because when it combines with a numeral, the resulting numeral phrase becomes an adverb unit rather than a verb modifier.

The generation of the classifier is controlled by the parameters in Question 4.6. The functions $\texttt{adjcl}(Q_{4.6})$ in Algorithm 3.30, $\texttt{advcl}(Q_{4.6})$ in Algorithm 3.31, $\texttt{nmodcl}(Q_{4.6})$ in Algorithm 3.32, and $\texttt{vmodcl}(Q_{4.6})$ in Algorithm 3.33 generates syntactic categories for the adjective classifier, the adverb classifier, the noun-modifying classifier, and the verb-modifying classifier, respectively. The control flag h_{cl} determines the attachment direction of the classifier when combined with a numeral.

The collection of all possible categories for the classifiers are generated by the function $\texttt{cl}(Q_{4.6})$ in Algorithm 3.34.

Unary Derivation Rules

Unary derivation rules are an additional part of the syntactic prototype. The use of these unary rules is permitted only when the language is identified as free word-order in Question 1. As seen in the function $\texttt{unary}(Q_1)$ in Algorithm 3.35, a noun phrase is converted into an argument of the verb by transforming into a modifier of s. A

Algorithm 3.30 $\texttt{adjcl}(Q_{4.6})$: generate all adjectival classifiers.

```
let $d \leftarrow \texttt{attdir}(h_{\text{cl}})$    ▷ attachment direction of the classifiers
for each word order parameter $q \in Q_{4.6}$ do
  for each adjective $x \in \texttt{adj}(Q_{2.1})$ do
    if $q = \texttt{Num} + \texttt{Cl}$ then
      yield $x\backslash_d\mathsf{num}$
    else if $q = \texttt{Cl} + \texttt{Num}$ then
      yield $x/_d\mathsf{num}$
    end if
  end for
end for
```

Algorithm 3.31 $\texttt{advcl}(Q_{4.6})$: generate all adverbial classifiers.

```
let $d \leftarrow \texttt{attdir}(h_{\text{cl}})$    ▷ attachment direction of the classifiers
for each word order parameter $q \in Q_{4.6}$ do
  for each adverb $x \in \texttt{adv}(Q_{2.1})$ do
    if $q = \texttt{Num} + \texttt{Cl}$ then
      yield $x\backslash_d\mathsf{num}$
    else if $q = \texttt{Cl} + \texttt{Num}$ then
      yield $x/_d\mathsf{num}$
    end if
  end for
end for
```

Algorithm 3.32 $\texttt{nmodcl}(Q_{4.6})$: generate all classifiers for nominal modifiers.

```
let $d \leftarrow \texttt{attdir}(h_{\text{cl}})$    ▷ attachment direction of the classifiers
for each word order parameter $q \in Q_{4.6}$ do
  for each nominal modifier $x \in \texttt{nmod}(Q_{4.4})$ do
    if $q = \texttt{Num} + \texttt{Cl}$ then
      yield $x\backslash_d\mathsf{num}$
    else if $q = \texttt{Cl} + \texttt{Num}$ then
      yield $x/_d\mathsf{num}$
    end if
  end for
end for
```

Algorithm 3.33 `vmodcl`$(Q_{4.6})$: generate all classifiers for verb modifiers.

```
let $d \leftarrow \texttt{attdir}(h_{\text{cl}})$    ▷ attachment direction of the classifiers
for each word order parameter $q \in Q_{4.6}$ do
    for each verb modifier $x \in \texttt{vmod}(Q_{4.4})$ do
        if $q = \texttt{Num} + \texttt{Cl}$ then
            yield $x\backslash_d\textsf{num}$
        else if $q = \texttt{Cl} + \texttt{Num}$ then
            yield $x/_d\textsf{num}$
        end if
    end for
end for
```

Algorithm 3.34 `cl`$(Q_{4.6})$: generate all classifiers.

```
return $\texttt{adjcl}(Q_{4.6}) \cup \texttt{advcl}(Q_{4.6}) \cup \texttt{nmodcl}(Q_{4.6}) \cup \texttt{vmodcl}(Q_{4.6})$
```

noun phrase is also converted into an adjective to approximate the noun declension system whereby nouns are declined to express their cases, thus able to modify each other systematically.

3.4 Summary

We have explained the main contribution of this thesis: our language parameterization, which are designed for improving the accuracy of unsupervised grammar induction. We have developed the motivation of this research and the benefits of incorporating syntactic prototypes into the unsupervised parsing models. We have reviewed the previous work prior to this research and explained our language parameterization method. Our method is linguistically motivated and easy to elicit by direct consultation with grammar compendiums or by interview with naïve informants.

To answer Research Question 1: Our language parameters are designed to capture frequent word orders. Each language parameter is sorted with respect to their frequency in natural language. We hypothesize that the more language parameters we incorporate into the syntactic prototype, the more accuracy we obtain. Because of their frequencies, the hypothetical trend of the accuracy is: the accuracy rises rapidly and starts to saturate as we increase the number of language parameters in the syntactic prototype.

Algorithm 3.35 $\texttt{unary}(Q_1)$: generate necessary unary derivation rules.

```
let $d \leftarrow \texttt{attdir}(h_{\mathrm{vmod}})$    ▷ attachment direction of the verb modifier
let $d' \leftarrow \texttt{attdir}(h_{\mathrm{adj}})$    ▷ attachment direction of the adjective
for each $Q_1$ allows free word order do
  ▷ All verbs must only be assigned as $\mathsf{s}$.
  ▷ A noun phrase becomes an argument of the verb.
  Allow the transformation of $\mathsf{np}$ into $\mathsf{s}\backslash_d\mathsf{s}$ and $\mathsf{s}/_d\mathsf{s}$.
  ▷ A noun phrase can modify each other like an adjective.
  for each $q \in Q_{2.1}$ do
    if $q = \texttt{Noun} + \texttt{Adj}$ then
      Allow the transformation of $\mathsf{np}$ into $\mathsf{np}\backslash_{d'}\mathsf{np}$
    else if $q = \texttt{Adj} + \texttt{Noun}$ then
      Allow the transformation of $\mathsf{np}$ into $\mathsf{np}/_{d'}\mathsf{np}$
    end if
  end for
end for
```

We have described the dialog we can use to elicit such prior knowledge by interview with naïve informants when the grammar compendiums for our languages of interest are not available. We have also explained how we encode the acquired prior knowledge into the syntactic prototype.

To answer Research Question 2: Human labor for language parameter elicitation is quantifiable. The process of eliciting the language parameters either via direct consultation with grammar compendium or via interview with naïve informants normally takes up to two hours per previously unseen language. Once we obtain the language parameters, we study the POS annotation guidelines for each language and map each tag to one or more language-specific category classes. To thoroughly scrutinize the usage of each POS tag and assign them to appropriate classes it typically takes around four to six hours. It therefore takes six to ten hours to build a syntactic prototype for each language.

In the next chapter, we will give an overview of our prototype-driven parser system.

Chapter 4

Grammar Induction

Outline

This chapter explains our method of grammar induction using the prior syntactic knowledge parameterized by word orders. In Section 4.1, we explain the system overview and the algorithms we use in each step. In Section 4.2, we describe our parsing models which are used and assessed in the upcoming experiments.

4.1 System Overview

This section details our method of syntactic structure recovery by exploiting the linguistic prototype. As illustrated in Figure 4.1, the system consists of three steps: structure enumeration, parameter estimation, and structure selection. In the first step, we enumerate all possible syntactic structures (i.e. constituents and dependencies) for each sentence in a POS-annotated corpus with CKY Algorithm. We constrain the size of the search space by using the mapping table between POS tags and syntactic categories. We then estimate the model parameters—the probabilities of substructures—with EM Algorithm. The more frequently seen a substructure is, the more probability it is iteratively assigned. Finally, we select the most probable structures from the corpus with the result language model. Let us explain each step as follows.

4.1.1 Structure Enumeration

The first step is to enumerate all possible parses for each sentence; i.e. we used the mapping between POS tags and syntactic categories to define the lexicon and built a

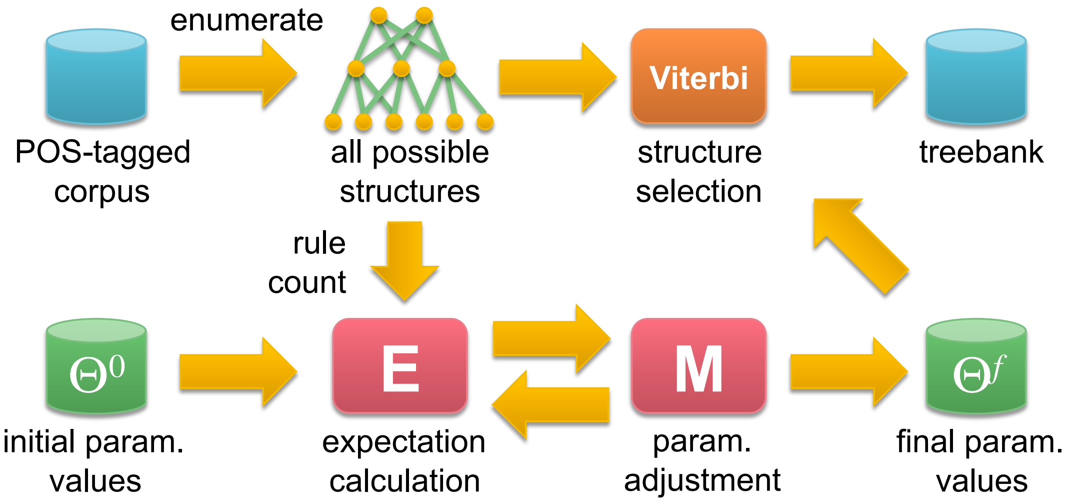


Figure 4.1: System overview

parse chart for each sentence. To elucidate this point, let us imagine that we parse each sentence with a very permissive grammar that allows any pairs of words and phrases to combine. We therefore obtain all possible constituent structures. However, the search space of the parser could astronomically expand when we generate dependency structures in parallel with the constituent structures. In order to shrink down the search space, we make use of the mapping between POS tags and syntactic categories. We then eliminate from the chart all edges that are not used in any trees.

Eliminating unnecessary edges from the training data—the packed charts, is crucial for any unsupervised learning techniques because they are sensitive and easy to be distorted by noises from irrelevant training data. The more we eradicate irrelevant training data, the more accurate our model is. With cheap linguistic prototype, we can achieve higher accuracy of syntactic structure recovery.

For efficiency reasons, we employ CKY Parsing Algorithm (Cocke and Schwartz, 1970; Kasami, 1965; Younger, 1967) as summarized in Algorithm 4.1. If the chart is complete; i.e. yielding the start symbol S, we recursively eliminate non-governed subtrees in the chart in a top-down fashion with Algorithms 4.2 and 4.3. Otherwise, we apply wildcard derivations into the chart to complete it via the algorithm described in Algorithm 4.4. We pack the chart to achieve both speed and space compactness. We also apply the right-branching preference to eliminate spurious ambiguity caused by conjunctions. For example, suppose that POS tags DT, JJ, and NN are assigned the categories $np/_{>}np$ and np. The chart of the sentence 'DT JJ JJ NN' produced by the CKY algorithm is shown in Figure 4.2.

Algorithm 4.1 `parse`(w_1^N, G): parse a sentence w_1^N with a CDG G.

```
let $C \leftarrow$ an empty packed chart
▷ Lexicon initialization
for $i \leftarrow 1$ to $N$ do
  for each lexicon entry $w_i \vdash A$ in $G$ do
    add_item($C, i, i, A, w_i$)
  end for
end for
▷ Structure enumeration
for edge length $l \leftarrow 2$ to $N$ do
  for $i \leftarrow 1$ to $N - l + 1$ do
    let $j \leftarrow i + l$
    ▷ Split index $k$
    for $k \leftarrow i$ to $j - 1$ do
      for each left daughter $I_L \in C[i, k]$ do
        let $A_L \leftarrow \text{type}(I_L)$
        for each right daughter $I_R \in C[k + 1, j]$ do
          let $A_R \leftarrow \text{type}(I_R)$
          if combination $A_L A_R \Rightarrow A$ is allowed in $G$ then
            add_item($C, i, j, A, (I_L, I_R)$)
          end if
        end for
      end for
    end for
    ▷ Perform unary derivations on edge $[i, j]$.
    for each item $I \in C[i, j]$ do
      let $A \leftarrow \text{type}(I)$
      if unary derivation $A \Rightarrow A'$ is allowed in $G$ then
        for each $(I_L, I_R) \in \text{prod}(I)$ do
          add_item($C, i, j, A', (I_L, I_R)$)
        end for
      end if
    end for
  end for
end for
return $C$
```

Algorithm 4.2 `find_govn_trees`(C, I, U): Find all governed subtrees where C is a packed chart, I is an item ID, and U is the accumulator of item IDs.

```
let $U' \leftarrow U$
if $I \notin U$ then
  update $U' \leftarrow U' \cup \{I\}$
  if $i \neq j$ then
    for each $(I_L, I_R) \in \mathrm{prod}(I)$ do
      update $U' \leftarrow U' \cup$ find_govn_trees$(C, I_L, U')$
      update $U' \leftarrow U' \cup$ find_govn_trees$(C, I_R, U'')$
    end for
  end if
end if
return $U'$
```

Algorithm 4.3 `elim_nongovn`(C): Eliminate all nongoverned subtrees in the packed chart, where C is a packed chart.

```
let $U \leftarrow \emptyset$
for each item $I \in C[1, N]$ do
  update $U \leftarrow U \cup$ find_govn_trees$(C, 1, N, I, U)$
end for
for $l \leftarrow 2$ to $N$ do
  for $i \leftarrow 1$ to $N - l + 1$ do
    let $j \leftarrow i + l - 1$
    for each item $I \in C[i, j]$ do
      if $I \notin U$ then
        del_item$(C, i, j, I)$
      end if
    end for
  end for
end for
```

Algorithm 4.4 `parse_wildcard`(w_1^N, C, i, j): Apply wildcard derivations on maximally non-governed edges in the range $[i, j]$ of the packed chart Q, given input sentence w_1^N.

```
if $i = j$ and $C[i, j] = \emptyset$ then
  add_item$(C, i, i, \star, w_i)$
else if $C[i, j] = \emptyset$ then
  for $k \leftarrow i$ to $j - 1$ do
    if $C[i', k] = \emptyset$ and $C[k + 1, j'] = \emptyset$ for all $i' < i$ and $j < j'$ then
      parse_wildcard$(w_1^N, C, i, k)$
      parse_wildcard$(w_1^N, C, k + 1, j)$
      for each item $I_L \in C[i, k]$ do
        for each item $I_R \in C[k + 1, j]$ do
          add_item$(C, i, j, \star, (I_L, I_R))$
        end for
      end for
    end if
  end for
end if
```

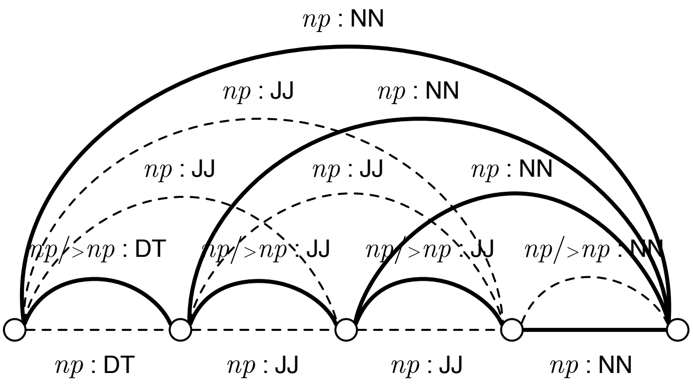


Figure 4.2: The chart of the sentence ‘DT JJ JJ NN’ obtained from the CKY Algorithm and eliminated excessive edges. The dashed arrows are the edges that are eliminated when finished.

4.1.2 Parameter Estimation and Decoding

We use the variational Bayesian Inside-Outside Algorithm (Kurihara and Sato, 2006) to approximate the parameters of the parsing models. As aforementioned in Section 2.3.2, VBEM (Attias, 2000; Ghahramani and Beal, 2000; Beal, 2003) seeks to maximize the a posteriori distribution of the dataset.

The algorithm iterates the two processes: expectation calculation and parameter maximization. The first step calculates the expectation (the average of probabilities) of each substructure out of the corpus. The second step adjusts the parameters of the substructures by the calculated expectations. The two processes are iteratated repeatedly until the entire parameters become stable, determined by the expectations converging. An EM algorithm, called Inside-Outside Algorithm (Baker, 1979; Lari and Young, 1990), is utilized for approximating the parameters of PCFG.

Once we estimate the model parameters, we use Viterbi Algorithm (Viterbi, 1967; Forney, 1973) to find the most likely parse of a sentence. Formally, we have to find the most likely tree t^* for a given input sentence s, such that

$$t^* \quad = \quad \arg\max_{t \in T(s)} P(t|G) \tag{4.1}$$

where $T(s)$ is the set of all possible trees for s.

4.2 Generative Parsing Models

A parsing model assigns a probability to a syntactic analysis of a string in a language. In practice, we cannot directly measure the probability of a tree by counting tree frequency due to data sparsity. We therefore need to factorize the tree probability into the product of non-overlapping substructure probabilities. The probability of a syntactic tree t with respect to a grammar G is defined as follows:

$$P(t|G) \quad = \quad \pi(r|G)\Phi(t|G) \prod_{t_i \in \mathrm{dtrs}(t)} P(t_i|G) \tag{4.2}$$

where t is a syntactic tree, r is the grammar rule used in the topmost derivation from the root node to its immediate daughters, $\Phi(t|G)$ is a feature function for the root node, and each $t_i \in \mathrm{dtrs}(t)$ is an immediate subtree of the root node of t. The probability of a grammar rule r, denoted by $\pi(r|G)$, is called a *parameter* of the parsing model π.

We employ the Variational Bayesian EM Algorithm to estimate the parameters of the parsing models from all the parsed sentences.

In the experiments in the next chapters, we use six generative parsing models, which are categorized and sorted by parametric expressiveness of their feature functions. Model 0 is the simplest model in which only a probabilistic context-free grammar is used. Model 1 and Model 2 are generative models extended from Model 0 with feature functions defined on dependency generation. Model 3 and Model 4 are extended from Model 1 and Model 2, respectively, with feature functions defined on lexical emission probability. Finally, Model 5, the most complex one, is a mixture of all feature functions.

Hypothetical trend: These models are ordered with respect to their expressiveness; i.e. Model 0 has the least number of parameters while Model 5 has the most number of parameters. With respect to the parameters we will gradually introduce to the models, we anticipate that the more parametrically complex the model is, the more accuracy we can attain from it. By incorporating the knowledge of frequent word order, we expect that if this prior knowledge adequately captures most frequent word order, a more complex model should perform better. However, once we increase the model's complexity, the accuracy may also be traded off due to the data sparsity issue.

4.2.1 Model 0: Probabilistic Context-Free Grammar (PCFG)

Model 0, the simplest parsing model in our repository, is essentially a probabilistic context-free grammar (Charniak, 1997; Johnson, 1998). The probability of a tree is the product of the probabilities of each context-free rule used to construct the tree; i.e. it treats each rule as an elementary structure. According to Eq (4.2), we define the feature function of Model 0 as follows.

$$\Phi_0(t|G) \;=\; 1 \tag{4.3}$$

Substituting Eq (4.3) to Eq (4.2), the probability of a tree t in Model 0 with respect to a PCFG G is therefore

$$\begin{aligned} P(t|G) &= \pi(r|G) \prod_{t_i \in \mathrm{dtrs}(t)} P(t_i|G) \\ &= \begin{cases} \pi(C : w \to w) & \text{lexicon} \\ \pi(C : w \to \alpha) \prod_{t_i \in \mathrm{dtrs}(t)} P(t_i|G) & \text{branching} \end{cases} \end{aligned} \tag{4.4}$$

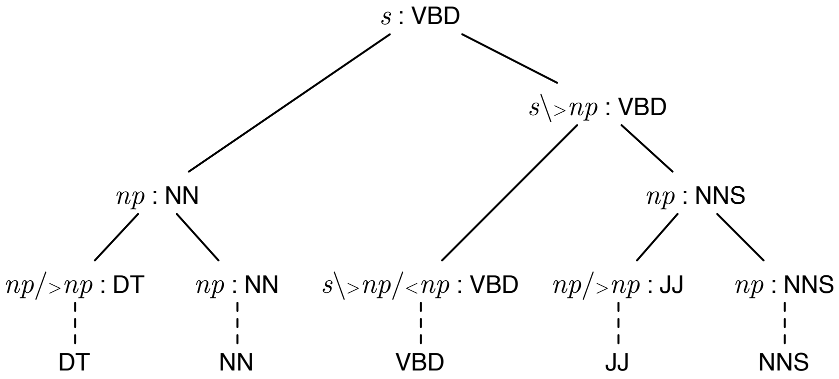


Figure 4.3: Dependency-driven syntactic analysis of the sentence 'DT NN VBD JJ NNS'

where C is a syntactic category, w is a terminal symbol (head word), and α is a sequence of syntactic categories.

When applying the idea of PCFG into CDG, we found one peculiar feature: the probabilities of all preterminal rules $A \rightarrow w$ can be omitted. This regards the fact that every preterminal category C with the head w always generates the word w, resulting in $P(C : w \rightarrow w) = 1$ for all $C : w$. We obtain the probability of a tree t with respect to a CDG G:

$$P(t|G) = \begin{cases} 1 & \text{lexicon} \\ \pi(C : w \rightarrow \alpha) \prod_{t_i \in \text{dtrs}(t)} P(t_i|G) & \text{branching} \end{cases} \tag{4.5}$$

The probability of the tree in Figure 4.3 can be computed by the product of the parameters of each rule as follows. (The product below is written in top-down depth-first order.)

$$\begin{aligned} P_{\text{CDG}}(t|s, G) = {} & \pi(s : \text{VBD} \rightarrow np : \text{NN}, s\backslash_{>}np : \text{VBD}) \\ & \times\pi(np : \text{NN} \rightarrow np/_{>}np : \text{DT}, np : \text{NN}) \\ & \times\pi(s\backslash_{>}np : \text{VBD} \rightarrow s\backslash_{>}np/_{<}np : \text{VBD}, np : \text{NNS}) \\ & \times\pi(np : \text{NNS} \rightarrow np/_{>}np : \text{JJ}, np : \text{NNS}) \end{aligned} \tag{4.6}$$

4.2.2 Model 1: Role-Emission Model

This model is an extension of Model 0 into which we incorporate the role-emission probabilities of all categories as a feature function. Besides the rule probabilities, we also take into account the probabilities of each child category performing as a head or a dependent. This model was motivated by Collins' (1999) head-outward dependency model (Collins, 1999) and Hockenmaier's (2003) CCG generative model (Hockenmaier, 2003b; Hockenmaier, 2003a).

In this model, the syntactic derivation is seen as a generative process. Given a parent category, we decide whether to generate an expansion or a leaf node. If we decide to generate an expansion, then we decide to generate the head and the daughter categories of such parent. The head and the dependent generated are then seen as new parents and we recursively generate the rest of the tree from them. Otherwise, if we decide to generate a leaf node, we just generate a lexical item. Suppose that G is a CDG and each $q \in \mathcal{Q}$ is a packed chart in the dataset $\mathcal{Q}$, the parameters are classified into three types:

1. $\pi_{\text{exp}}(\alpha|C:w,G)$: probability of the category $C:w$ generating a production α, which is equivalent to $\pi(C:w \rightarrow \alpha|G)$,

2. $\pi_{\text{head}}(C:w|G)$: probability of $C:w$ emitting the head role,

$$\pi_{\text{head}}(C:w|G) \quad = \quad \frac{\sum_{q\in\mathcal{Q}}\sum_{I\in\text{items}(q)} R_I^{\text{h}}(C:w)}{\sum_{q\in\mathcal{Q}}\sum_{I\in\text{items}(q)}\sum_{r\in\{\text{h,d}\}} R_I^{r}(C:w)} \tag{4.7}$$

3. $\pi_{\text{dep}}(C:w|G)$: probability of $C:w$ emitting the dependent role,

$$\pi_{\text{dep}}(C:w|G) \quad = \quad \frac{\sum_{q\in\mathcal{Q}}\sum_{I\in\text{items}(q)} R_I^{\text{d}}(C:w)}{\sum_{q\in\mathcal{Q}}\sum_{I\in\text{items}(q)}\sum_{r\in\{\text{h,d}\}} R_I^{r}(C:w)} \tag{4.8}$$

The quantity $R_I^r(C:w)$ is the expected count of probabilities of rules in parse item I whose production has category $C:w$ emitting the role $r \in \{\text{h}, \text{d}\}$; i.e.

$$R_I^r(C:w) \quad = \quad \sum_{\alpha\in\text{prod}(I)} \pi(C':w' \rightarrow \alpha|G)\rho_\alpha^r(C:w) \tag{4.9}$$

where $\text{type}(I) = C':w'$ and

$$\rho_\alpha^{\text{h}}(C:w) \quad = \quad \begin{cases} 1 & \text{if } \alpha \text{ has } C:w \text{ as the head} \\ 0 & \text{otherwise} \end{cases} \tag{4.10}$$

$$\rho_\alpha^{\text{d}}(C:w) \quad = \quad \begin{cases} 1 & \text{if } \alpha \text{ has } C:w \text{ as the dependent} \\ 0 & \text{otherwise} \end{cases} \tag{4.11}$$

The parameters for all preterminal rules are 1 because the category $C : w$ always generates the lexical item w.[1] It should be noted that the parameter $\pi_{\text{dep}}(D|C, dir)$ differs from Hockenmaier's (2003a) daughter parameter in that we do not take the head category into account.

From the above generative process, we can define the feature function for Model 1 as follows.

$$\Phi_1(t|G) \quad = \quad \begin{cases} \pi_{\text{head}}(H : w|G)\pi_{\text{dep}}(D : w'|G) & \text{branching} \\ 1 & \text{lexicon} \end{cases} \tag{4.12}$$

Substituting Eq (4.12) to Eq (4.2), we can recursively define the probability of a tree t having the category $C : w$ as follows.

$$P(t|G) \quad = \quad \begin{cases} \pi_{\text{exp}}(H : w, D : w'|C : w, G) & \text{head-left} \\ \quad \times\pi_{\text{head}}(H : w|G) \times \pi_{\text{dep}}(D : w'|G) & \\ \quad \times P(H : w|G) \times P(D : w'|G) & \\ \pi_{\text{exp}}(D : w', H : w|C : w, G) & \text{head-right} \\ \quad \times\pi_{\text{dep}}(D : w'|G) \times \pi_{\text{head}}(H : w|G) & \\ \quad \times P(D : w'|G) \times P(H : w|G) & \\ 1 & \text{lexical item} \end{cases} \tag{4.13}$$

To clarify the idea of tree conversion, let t be the tree in Figure 4.4. The probability of t is given by:

$$\begin{aligned} P(t|G) \quad = \quad & \pi_{\text{exp}}(s : \text{VBD} \to np : \text{NNS}, s\backslash_{>}np : \text{VBD}|G) \\ & \times\pi_{\text{dep}}(np : \text{NNS}|G) \\ & \times\pi_{\text{head}}(s\backslash_{>}np : \text{VBD}|G) \end{aligned} \tag{4.14}$$

4.2.3 Model 2: Mother-Daughter Model

Model 2 is an extension of Model 1 in which each daughter node is instead generated from its headword. The syntactic derivation is seen as a generative process. Given a parent category, we decide whether to generate an expansion or a leaf node. If we decide to generate an expansion, then we decide to generate the mother category of

[1] We can derive this from the fact that $\pi_{\text{lex}}(w|C : w, G) = \frac{P(w,C,w,G)}{P(C,w,G)} = \frac{P(C,w,G)}{P(C,w,G)} = 1$.

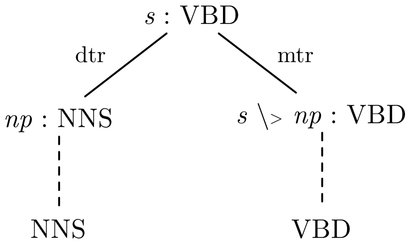


Figure 4.4: Role-emission generative model of a syntactic derivation of the sentence ‘NNS VBD’

such parent and its word. Then we generate the daughter category and its word given the mother word. The head and daughter nodes generated are then seen as new parents and we recursively generate the rest of the tree from them. Otherwise, if we decide to generate a leaf node, we just generate a lexical item. Suppose that G is a CDG and each $q \in \mathcal{Q}$ is a packed chart in the dataset $\mathcal{Q}$, the parameters are classified into three types:

1. $\pi_{\text{exp}}(\alpha|C : w, G)$: probability of the category $C : w$ to generate a production α, which is equivalent to $\pi(C : w \rightarrow \alpha|G)$,

2. $\pi_{\text{mtr}}(C : w|G)$: probability of $C : w$ emitting the mother role,

$$\pi_{\text{mtr}}(C : w|G) \quad = \quad \frac{\sum_{q \in \mathcal{Q}} \sum_{I \in \text{items}(q)} R_I^{\text{h}}(C : w)}{\sum_{q \in \mathcal{Q}} \sum_{I \in \text{items}(q)} \sum_{r \in \{\text{h},\text{d}\}} R_I^{r}(C : w)} \tag{4.15}$$

3. $\pi_{\text{dtr}}(C : w'|w, G)$: probability of $C : w'$ emitting the daughter role given the headword w,

$$\pi_{\text{dtr}}(C : w'|w, G) \quad = \quad \frac{\sum_{q \in \mathcal{Q}} \sum_{I \in \text{items}(q)} Q_I(C : w'; w)}{\sum_{q \in \mathcal{Q}} \sum_{I \in \text{items}(q)} \sum_{w''} Q_I(C : w'; w'')} \tag{4.16}$$

The quantity $Q_I(C : w'; w)$ is the expected count of probabilities of rules in parse item I whose production has category $C : w'$ as the daughter generated from the headword w. If $\text{type}(I) = X : w$, where X is any syntactic category,

$$Q_I(C : w'; w) \quad = \sum_{\alpha \in \text{prod}(I)} \pi(X : w \rightarrow \alpha|G) R_I^{\text{d}}(C : w) \tag{4.17}$$

Otherwise, $Q_I(C : w'; w) = 0$. It is worth noting that the feature π_{dtr} is close to Hockenmaier’s (2003a) daughter parameter.

From the above generative process, we can define the feature function for Model 2 as follows.

$$\Phi_2(t|G) \quad = \quad \begin{cases} \pi_{\text{mtr}}(C:w|G)\pi_{\text{dtr}}(C:w'|w,G) & \text{branching} \\ 1 & \text{lexicon} \end{cases} \tag{4.18}$$

Substituting Eq (4.18) to Eq (4.2), we can recursively define the probability of a tree t having the category $C : w$ as follows.

$$P(t|G) \quad = \quad \begin{cases} \pi_{\text{exp}}(H:w, D:w'|C:w,G) & \text{head-left} \\ \quad \times\pi_{\text{mtr}}(H:w|G) \times \pi_{\text{dtr}}(D:w'|w,G) & \\ \quad \times P(H:w|G) \times P(D:w'|G) & \\ \pi_{\text{exp}}(D:w', H:w|C:w,G) & \text{head-right} \\ \quad \times\pi_{\text{dtr}}(D:w'|w,G) \times \pi_{\text{mtr}}(H:w|G) & \\ \quad \times P(D:w'|G) \times P(H:w|G) & \\ 1 & \text{lexical item} \end{cases} \tag{4.19}$$

For example, we can compute the probability of the tree in Figure 4.4 as follows.

$$\begin{aligned} P(t|G) \quad = \quad & \pi_{\text{exp}}(s:\text{VBD} \to np:\text{NNS}, s\backslash_{>}np:\text{VBD}|G) \\ & \times\pi_{\text{dtr}}(np:\text{NNS}|\text{VBD},G) \\ & \times\pi_{\text{mtr}}(s\backslash_{>}np:\text{VBD}|G) \end{aligned} \tag{4.20}$$

4.2.4 Model 3: Role-Emission + Lexicon-Emission Model

We extend Model 1 with the probabilities of lexicon emission—i.e. the probability of each preterminal syntactic category C generating a word w. This probability is a ratio of the total number of the category C generating the head w and the total number of all head words w' the category C can generate in the entire tree set. Let us define the lexicon-emission probability:

$$\pi_{\text{lex}}(w|C,G) \quad = \quad \frac{\sum_{q\in\mathcal{Q}}\sum_{I\in\text{items}(q)}\#_I^{\text{lex}}(C:w)}{\sum_{q\in\mathcal{Q}}\sum_{I\in\text{items}(q)}\sum_{w_k}\#_I^{\text{lex}}(C:w_k)} \tag{4.21}$$

where each $q \in \mathcal{Q}$ is a packed chart, each I is a parsing item in q, and $\#_I^{\text{lex}}(C:w)$ is defined as:

$$\#_I^{\text{lex}}(C:w) \quad = \quad \begin{cases} \pi(C:w \to w) & \text{if type}(I) = C:w \text{ and } I \text{ is a lexicon item} \\ 0 & \text{otherwise} \end{cases}$$

We include $\pi_{\text{lex}}(w|C,G)$ as an additional parameter of Model 1 by defining the feature function as follows.

$$\Phi_3(t|G) = \begin{cases} \pi_{\text{head}}(H : w|G)\pi_{\text{dep}}(D : w'|G) & \text{branching} \\ \pi_{\text{lex}}(w|C,G) & \text{lexicon} \end{cases} \tag{4.22}$$

Substituting Eq (4.22) to Eq (4.2), the probability of a tree becomes a product of (1) the probabilities of each rule constituting the tree (2) each category emitting the head/dependent role and (3) the probabilities of each preterminal category emitting its lexical item. We therefore have that:

$$P(t|G) = \begin{cases} \pi_{\text{exp}}(H : w, D : w'|C : w, G) & \text{head-left} \\ \quad \times\pi_{\text{head}}(H : w|G) \times \pi_{\text{dep}}(D : w'|G) & \\ \quad \times P(H : w|G) \times P(D : w'|G) & \\ \pi_{\text{exp}}(D : w', H : w|C : w, G) & \text{head-right} \\ \quad \times\pi_{\text{dep}}(D : w'|G) \times \pi_{\text{head}}(H : w|G) & \\ \quad \times P(D : w'|G) \times P(H : w|G) & \\ \pi_{\text{lex}}(w|C,G) & \text{lexical item} \end{cases} \tag{4.23}$$

4.2.5 Model 4: Mother-Daughter + Headword-Emission Model

We furthermore extend Model 2 with the probabilities of headword emission—i.e. the probability of a syntactic category C generating the headword of the constituent w. This probability is a ratio of the total number of the category C generating the head w and the total number of all head words w' the category C can generate in the entire tree set. Let us define the headword-emission probability:

$$\pi_{\text{headword}}(w|C,G) = \frac{\sum_{q\in\mathcal{Q}}\sum_{I\in\text{items}(q)} \#_I^{\text{hw}}(C : w)}{\sum_{q\in\mathcal{Q}}\sum_{I\in\text{items}(q)}\sum_{w_k} \#_I^{\text{hw}}(C : w_k)} \tag{4.24}$$

where each $q \in \mathcal{Q}$ is a packed chart, each I is a parsing item in q, and $\#_I^{\text{hw}}(C : w)$ is defined as:

$$\#_I^{\text{hw}}(C : w) = \begin{cases} \sum_{\alpha\in\text{prod}(I)} \pi(C : w \rightarrow \alpha) & \text{if type}(I) = C : w \\ 0 & \text{otherwise} \end{cases}$$

where each α is a production in item I.

We include $\pi_{\text{headword}}(w|C,G)$ as an additional parameter of Model 2 by defining the feature function as follows.

$$\Phi_4(t|G) \quad = \quad \begin{cases} \pi_{\text{mtr}}(H:w|G)\pi_{\text{dtr}}(D:w'|G) & \text{branching} \\ \qquad \times\pi_{\text{headword}}(w|C,G) & \\ \pi_{\text{headword}}(w|C,G) & \text{lexicon} \end{cases} \tag{4.25}$$

Substituting Eq (4.25) to Eq (4.2), the probability of a tree becomes a product of (1) the probabilities of each rule constituting the tree (2) each category emitting the head/dependent role and (3) the probabilities of each constituent's category emitting its head word. We therefore have that:

$$P(t|G) \quad = \quad \begin{cases} \pi_{\text{headword}}(w|C,G) & \text{head-left} \\ \quad \times\pi_{\text{exp}}(H:w,D:w'|C:w,G) & \\ \quad \times\pi_{\text{mtr}}(H:w|G)\times\pi_{\text{dtr}}(D:w'|G) & \\ \quad \times P(H:w|G)\times P(D:w'|G) & \\ \pi_{\text{headword}}(w|C,G) & \text{head-right} \\ \quad \times\pi_{\text{exp}}(D:w',H:w|C:w,G) & \\ \quad \times\pi_{\text{dtr}}(D:w'|G)\times\pi_{\text{mtr}}(H:w|G) & \\ \quad \times P(D:w'|G)\times P(H:w|G) & \\ \pi_{\text{headword}}(w|C,G) & \text{lexical item} \end{cases} \tag{4.26}$$

4.2.6 Model 5: Mixture of All

Model 5 is the mixture of Models 0-4; i.e. we extend the probabilisitic context-free grammar with all features aforementioned. There are seven parameters to be trained in this model:

1. $\pi_{\text{exp}}(\alpha|C:w,G)$: the probability of a tree is the product of the probabilities of all rules constituting the tree,

2. $\pi_{\text{head}}(C:w|G)$: the probability of each category emitting its head role,

3. $\pi_{\text{dep}}(C:w|G)$: the probability of each category emitting its dependent role,

4. $\pi_{\text{mtr}}(C:w|G)$: the probability of each category emitting its mother role,

5. $\pi_{\text{dtr}}(C : w'|w, G)$: the probability of each daughter is generated from its head,

6. $\pi_{\text{lex}}(w|C, G)$: the probability of each preterminal category emitting its word, and

7. $\pi_{\text{headword}}(w|C, G)$: the probability of each constituent's category emitting its headword.

The feature function of Model 5 is as follows.

$$\Phi_5(t|G) \;=\; = \begin{cases} \pi_{\text{head}}(H : w|G)\pi_{\text{dep}}(D : w'|G) & \text{branching} \\ \quad \times\pi_{\text{mtr}}(C : w|G)\pi_{\text{dtr}}(C : w'|w, G) & \\ \quad \times\pi_{\text{headword}}(w|C, G) & \\ \pi_{\text{lex}}(w|C, G)\pi_{\text{headword}}(w|C, G) & \text{lexicon} \end{cases} \tag{4.27}$$

The probability of a tree t with respect to a PCFG G is based on Models 0-4 as recursively defined below.

$$P(t|G) \;=\; \begin{cases} \pi_{\text{exp}}(H : w, D : w'|C : w, G) & \text{head-left} \\ \quad \times\pi_{\text{head}}(H : w|G) \times \pi_{\text{dep}}(D : w'|G) & \\ \quad \times\pi_{\text{mtr}}(H : w|G) \times \pi_{\text{dtr}}(D : w'|w, G) & \\ \quad \times\pi_{\text{headword}}(w|C, G) & \\ \quad \times P(H : w|G) \times P(D : w'|G) & \\ \pi_{\text{exp}}(D : w', H : w|C : w, G) & \text{head-right} \\ \quad \times\pi_{\text{dep}}(D : w'|w, G) \times \pi_{\text{head}}(H : w|G) & \\ \quad \times\pi_{\text{dtr}}(D : w'|w, G) \times \pi_{\text{mtr}}(H : w|G) & \\ \quad \times\pi_{\text{headword}}(w|C, G) & \\ \quad \times P(D : w'|G) \times P(H : w|G) & \\ \pi_{\text{lex}}(w|C, G) \times \pi_{\text{headword}}(w|C, G) & \text{lexical item} \end{cases} \tag{4.28}$$

4.2.7 Summary

To summarize this section, let us review the parsing models we have introduced. Sorted by the parametric expressiveness, the following are the six models used in the experiments in which all of them are evaluated across the board.

1. **Model 0** (probabilistic context-free grammar): the probability of a tree is the product of the probabilities of all rules constituting the tree.

2. **Model 1** (role-emission model): the probability of a tree is the product of (1) the probabilities of all branchings and (2) the probabilities of each category emitting its head/daughter role.

3. **Model 2** (mother-daughter model): the probability of a tree is the product of (1) the probabilities of all branchings and (2) the probabilities of each head is generated and (3) the probabilities of each daughter is generated from its head.

4. **Model 3** (role-emission + lexicon-emission model): the probability of a tree is the product of (1) the probabilities of all rules constituting the tree and (2) the probabilities of each preterminal category emitting its word.

5. **Model 4** (mother-daughter + headword-emission model): the probability of a tree is the product of (1) the probabilities of all branchings, (2) the probabilities of each category emitting its head/daughter role, and (3) the probabilities of each constituent's category emitting its head word.

6. **Model 5** (mixture of all): the probability of a tree is the product of (1) the probability of a tree is the product of the probabilities of all rules constituting the tree (2) the probabilities of each category emitting its head/daughter role (3) the probabilities of each head is generated (4) the probabilities of each daughter is generated from its head (5) the probabilities of each preterminal category emitting its word and (6) the probabilities of each constituent's category emitting its head word.

These models are ordered with respect to their expressiveness; i.e. Model 0 has the least number of parameters while Model 5 has the most number of parameters. With respect to the parameters we gradually introduced to the models, we anticipate that the more parametrically complex the model is, the more accuracy we can attain from it. By incorporating the knowledge of frequent word order, we predict that if this prior knowledge adequately capture most frequent word order, a more complex model should perform better.

Part III

Experiments and Discussion

Chapter 5

Multilingual Experiments

Outline

This chapter presents our experiment settings and the comparison of experiment results with the state-of-the-art techniques. To make our experiments replicable, we explain how to initialize the model parameterss and describe our controlled variables. The results are presented in three aspects: (1) multilingual experiments in which we compare our results with the related work and with PASCAL Challenge (2) long-tail dependencies (3) scalability of language parameters.

5.1 Methods

5.1.1 Gold Standard and Test Corpus

We use the CoNLL-2006 dependency banks as the gold standard and testing data we have already prepared following the procedure described in Section 2.5.3. We evaluate our method on sentence lengths 10, 15, and 20 of all 14 languages.

5.1.2 Training and Evaluation

Our experimental procedure is similar to standard completely unsupervised parser induction. We are attempting to recover syntactic structures from sequences of POS tags instead of surface words to avoid the data sparsity issue. We do not separate the testing dataset from the training one. We consider this practice sound and safe from potential data-overfitting issues because the syntactic prototypes are derived directly from the interview and the treebank manuals, and are blind to the gold-standard dependency structures. Guided by these syntactic prototypes, we approximate the parsing model with Variational Bayesian EM, treating POS tag sequences as observed and the syntactic structures produced by the CKY algorithm as unobserved. We then reproduce the treebanks from each POS tag sequence with the models and evaluate the accuracy of directed dependency recovery.

In evaluation, we assess the accuracy of the parser by head-to-head comparison of the dependency relations of our parsed trees with the gold-standard dependency relations described in the HEAD field in the CoNLL-X Shared Task 2006's format. We do not take into account any labeled dependency relations described in DEPREL and PDEPREL.

5.1.3 Language Parameters

We make use of the language parameters for each language elicited from consultation with our grammar compendiums. We closely follow the method described in Section 3.2.

5.1.4 Parameter Initialization

We set the initial values of the rule parameters at random. We set the initial parameter for a rule of the form $A \to X$ by sampling from the uniform distribution:

$$\pi^{(0)}(A \to X) \quad \sim \quad \mathrm{Uniform}(\frac{\alpha_{A\to X}}{100}, \alpha_{A\to X}) \tag{5.1}$$

where each $q \in \mathcal{Q}$ is a packed chart in the dataset $\mathcal{Q}$, the quantity $\alpha_{A\to X}$ is the pseudo-probability of the rule defined by:

$$\alpha_{A\to X} \quad = \quad \frac{\sum_{q\in\mathcal{Q}} \sum_{I\in\mathrm{items}(q)} \#_I(A \to X)}{\sum_{q\in\mathcal{Q}} \sum_{I\in\mathrm{items}(q)} \sum_{X'} \#_I(A \to X')} \tag{5.2}$$

and $\#_I(A \to X)$ is defined as:

$$\#_I(A \to X) \quad = \quad \begin{cases} 1 & \text{if } \mathrm{type}(I) = A \text{ and } X \in \mathrm{prod}(I) \\ 0 & \text{otherwise} \end{cases} \tag{5.3}$$

Once we obtain all initial parameters $\pi^{(0)}$ we normalize them before estimating them with the EM Algorithm.

5.1.5 Controlled Variables

There are three controlled variables in our experiments: prior hyperparameters, the expressiveness of parsing models, and the number of EM iterations used in parameter adjustment.

Prior hyperparameters: We assign the hyperparameters for wildcard rules and sentence formation rules with 10.0, and 10000.0, respectively, while the normal CDG derivation rules are assigned with 1.0. We believe that the parameter estimation for wildcard rules and sentence formation should be less sensitive to noise.

Parsing models: We investigate the effects of the expressiveness of each parsing model in prototype-driven grammar induction. Recalling from Section 4.2, we experiment on six parsing models: Model 0 to Model 5. Each of these is assessed for its expressiveness — capability of capturing frequent word orders. We anticipate that the more expressive a parsing model is, the more accuracy we can achieve in grammar induction given that the language parameters can adequately capture most frequent word orders.

Number of EM iterations: We fix the number of iterations for the EM algorithm to at most five iterations for each experiment. We set the convergence threshold for the EM Algorithm to 10^{-6}.

Number of language parameters: We investigate the effects of our language parameters in prototype-driven grammar induction. We vary the number of language parameters used in the syntactic prototype in four levels: the first 3 parameters, the first 16 parameters, the first 27 parameters, and all parameters. Hypothetically, the accuracy rapidly rises and starts to saturate as we increase the number of language parameters, because the first parameters are designed to capture most frequent word orders.

5.1.6 Baseline Systems

We compare our results with three prototype-driven parsers (Naseem et al., 2010; Boonkwan and Steedman, 2011; Bisk and Hockenmaier, 2012b) and two unsupervised parsers (Gillenwater et al., 2010; Cohn et al., 2010). Among those, our former system (Boonkwan and Steedman, 2011) plays an important role to this thesis as it is a proof-of-concept study towards the use of our syntactic prototype.

The most significant difference between Boonkwan and Steedman (2011) and this thesis is the use of category penalty score in decoding, which is based on the assumption that simpler categories tend to be used more frequently than the more complex ones. From the experiments, the penalty score is likely to overwhelm the estimated model parameters. Additional offline experiments conducted after we published this paper show that we can expunge the use of category penalty score by simply adjusting the prior hyperparameters to those described in Section 5.1.5. Therefore, the category penalty score is no longer used in this thesis.

5.2 Results

5.2.1 Experiment 1

This section presents experimental results for grammar induction over all languages attempted. We compare our results with three prototype-driven parsers (Naseem et al., 2010; Boonkwan and Steedman, 2011; Bisk and Hockenmaier, 2012b) and two unsupervised parsers (Gillenwater et al., 2010; Cohn et al., 2010). The accuracy comparison is shown in Table 5.1. We report only the best results yielded from our models which are also annotated to the numbers (e.g. 77.14 (M_0) is the number from Model 0). Two of them (Cohn et al., 2010; Bisk and Hockenmaier, 2012b) only report their performance in EN10, so theirs will be mentioned in the content instead.

Our method significantly outperforms the state-of-the-art techniques on 13 out of 14 languages (BU10, CH10, CZ10, DA10, DU10, EN10, DE10, JA10, PO10, ES10, SL10, SV10, and TU10). The language parameters acquired by short interview (as in CH10 and JA10) perform as well as those induced from translations and word alignment produced by Google Translate (as in DA10, DU10, PO10, ES10, and SV10) do. There is however no prior work on AR10 so it is presented without any compared baselines.

It is worth comparing the performance of each technique on EN10. Our system outperforms the state-of-the-art techniques including Boonkwan and Steedman's (2011) prototype-driven grammar induction method. Our method also outperforms Bisk and Hockenmaier's (2012b) semi-supervised CCG parser ($F_1 = 71.5\%$) and Cohn et al.'s (2010)'s unsupervised TSG parser ($F_1 = 65.9\%$).

Table 5.1: Directed dependency accuracy (F_1) of grammar induction against corpora 10. We compare our results with three baselines: #1: Naseem et al. (2010) #2: Gillenwater et al. (2010) #3: Boonkwan and Steedman (2011).

Languages	**Our Best**	**Averages**	**S.D.**	**Baselines**		
				#1	**#2**	**#3**
PO10	**77.14** (M_0)	72.73	2.89	71.5	49.5	-
EN10	**76.81** (M_5)	72.79	3.49	71.9	64.4	75.47
JA10	**72.53** (M_2)	67.17	4.75	-	59.4	68.55
ES10	**71.43** (M_0)	68.49	2.20	64.8	57.9	-
SV10	**70.36** (M_0)	62.70	2.42	63.3	41.4	-
CH10	**66.09** (M_0)	61.47	2.76	-	35.77	62.25
BU10	**65.76** (M_2)	59.13	6.03	-	59.8	-
AR10	64.60 (M_0)	55.28	4.90	-	-	-
CZ10	**63.76** (M_1)	63.18	0.51	-	54.6	54.54
DA10	**63.64** (M_5)	60.05	2.00	51.9	-	-
TU10	**63.62** (M_2)	58.96	3.24	-	56.9	-
DE10	**59.91** (M_0)	54.51	2.89	-	45.7	56.71
SL10	**58.44** (M_1)	52.34	7.03	50.6	51.2	-
DU10	**53.97** (M_0)	50.12	3.67	-	38.8	-

Table 5.2: Average directed dependency accuracies (F_1) of each model against corpora 10, ordered by their averages.

Models	**Average F_1**	**S.D.**
M_0	64.52	6.70
M_2	63.11	8.29
M_1	62.16	8.23
M_5	60.60	7.81
M_3	59.18	6.68
M_4	58.54	8.11

We also report the across-language average accuracies of each model on corpora 10 as shown in Table 5.2. The trade-off between the accuracy and the model expressivity is obvious. In general, Model 0 seems to outperform the other models although it is the least expressive. We hypothesize that the other models bring about the issue of data sparsity, thus deteriorating the accuracies. Models 1 and 2 outperform Models 3 and 4 although the former models are less expressive due to the same reason. Model 5, however, outperforms Models 3 and 4, possibly because its expressivity starts to help capture frequent dependency types.

5.2.2 Experiment 2

Recently, Gelling and Cohn (2012) organized a competition on linguistic structure induction in which various unsupervised parsers are compared against each other on lengths 10 and 15. Slightly different from ours, the set of languages for the competition include Arabic, Basque, Czech, Danish, English (PTB), English (CHILDES Corpus), Portuguese, Slovene, and Swedish; therefore, we compare only the languages available in our repository.

Since the experiment protocol for the competition is confidential, concealing the amount of training data, we present the evaluation on lengths 10 and 15 of the models trained on length 15, as shown in Table 5.3. We do not combine these results with Table 5.1 due to incompatibility of the training and evaluation schemes. To the best of our knowledge, they declared the sentence lengths they used for training (10 and 15) and the sentence length for evaluation (10), but we do not know if they separated the training and test sets or combined them, leaving us no choice but to separate the results into two tables.

In the competition, our system outperforms the participating systems on Czech, Danish, Dutch, English, and Portuguese when evaluating the models trained on length 10. Our results on Arabic and Slovene are slightly inferior to those of PASCAL Challenge's participants while the result of Portuguese is significantly lower by 6%. Our system underperforms in Arabic, Slovene, and Swedish, most of which done by Tu (2012), when assessed on lengths 10, whereas our system outperforms the others on Swedish on length 15. It is noticeable that his Swedish accuracy significantly drops by 10% while ours does by only 1%. It implies that his Swedish model overfits frequent dependency types as the accuracy is deteriorated by long-tail dependencies.

5.2.3 Experiment 3

We investigate the scalability of our syntactic prototypes on long-tail dependencies. Figure 5.1 (full detail in Table 5.4) shows F_1 scores of directed dependency accuracy on various sentence lengths (up to 10, 15, and 20 words) on each language when our best parsing models are used. The accuracy trends of most languages conform to each other where the accuracy decreases and seem to saturate as the input sentences get longer. It is worth noticing that the accuracy in Bulgarian, Chinese, Danish, Dutch, Spanish, Portuguese, and Turkish vary within 10% F_1 range in corpora 20. It suggests to us that it is easier to capture most frequently used rules with the syntactic prototypes.

Using for calculating the similarity between dependency structures annotated in different schemes, the TEDEVAL scores of our system show that the F_1 scores drop because of the discrepancy of annotation schemes. Regardless of the dependency schemes, all scores seem much less sensitive to the long-tail dependencies because these scores drop within less than 10% in corpora 20 for all languages.

5.2.4 Experiment 4

We study the effects of syntactic prototypes in grammar induction by varying the number of syntactic constraints, when our best parsing models are used. The constraints are arranged in order of frequency in Table 3.1 and fed into the system. In Figure 5.2 (full detail in Table 5.5), the overall trend is that accuracy on all languages increases as we add more syntactic constraints to the syntactic prototypes. However, in Arabic, Czech, Dutch, Spanish, Japanese, and Slovene, the accuracy marginally decreases when we used all constraints, signifying inherent conflicts of the syntactic prototypes.

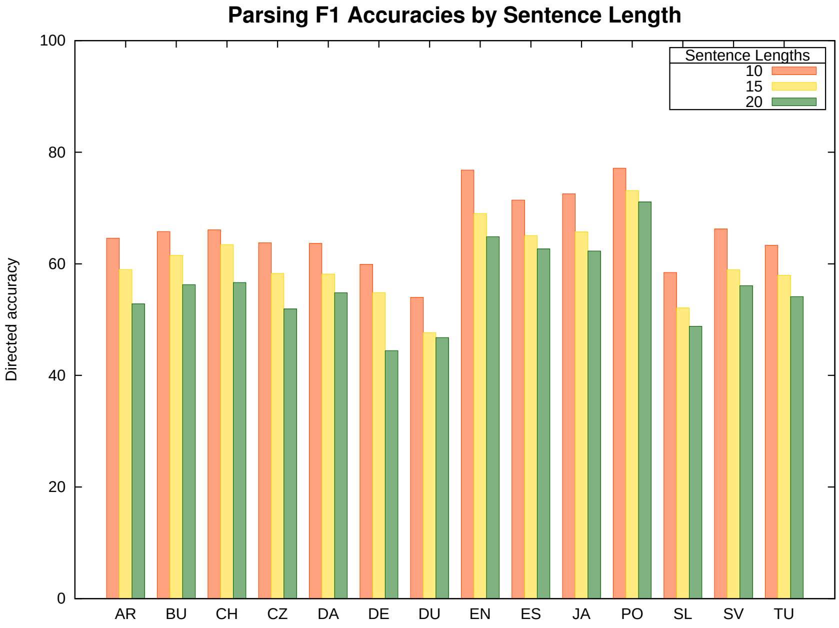
Table 5.3: Directed dependency accuracy (F_1) of grammar induction against corpora *of length 10* where the models are trained on lengths 10 and 15. We compare our results with three baselines: #1: Bisk and Hockenmaier (2012) #2: maximum accuracies quoted from PASCAL Challenge (Gelling et al., 2012) #3: Blunsom and Cohn (2010).

Model	**Evaluated**	**Baselines**		
trained on	**on length 10**	**#1**	**#2**	**#3**
AR10	64.60 (M_0)	41.6	**66.67**	60.8
BU10	65.76 (M_2)	-	-	-
CH10	66.09 (M_0)	-	-	-
CZ10	**63.76** (M_1)	45.0	61.34	47.9
DA10	**63.64** (M_5)	46.4	61.38	44.7
DU10	**53.97** (M_0)	49.7	51.72	51.8
EN10	**76.81** (M_5)	68.2	74.67	68.6

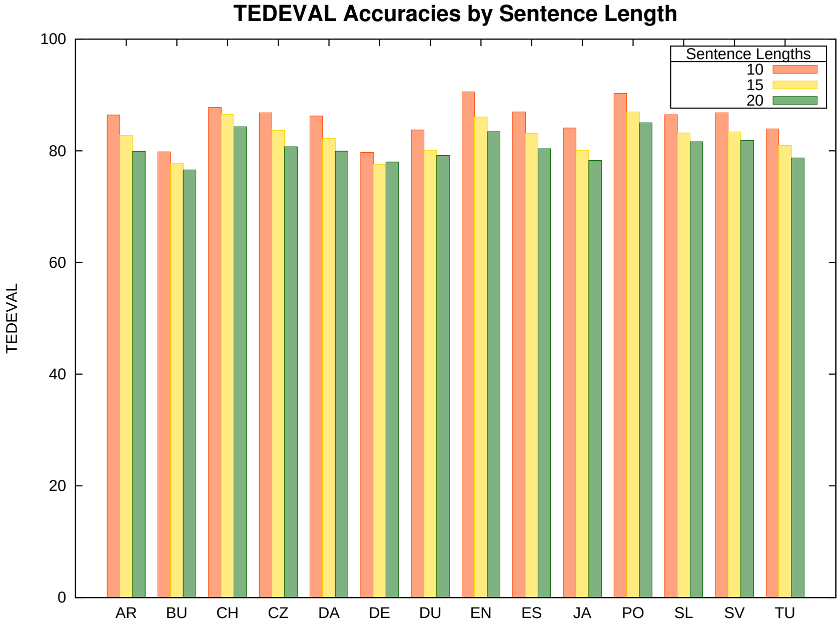
Model	**Evaluated**	**Baselines**		
trained on	**on length 10**	**#1**	**#2**	**#3**
DE10	59.91 (M_0)	-	-	-
JA10	72.53 (M_2)	-	-	-
PO10	**77.14** (M_0)	70.8	76.28	52.4
ES10	71.43 (M_0)	-	-	-
SL10	58.44 (M_1)	49.6	**67.63**	62.6
SV10	70.36 (M_0)	63.7	**76.54**	63.2
TU10	63.62 (M_2)	-	-	-

Model	**Evaluated**	**Baselines**		
trained on	**on length 10**	**#1**	**#2**	**#3**
AR15	66.07 (M_0)	43.7	**68.16**	58.4
BU15	67.53 (M_1)	-	-	-
CH15	66.44 (M_0)	-	-	-
CZ15	**63.65** (M_1)	38.9	56.49	43.1
DA15	**63.42** (M_5)	43.8	57.11	39.4
DU15	**56.28** (M_3)	43.6	52.26	52.0
EN15	**75.26** (M_5)	59.6	67.43	63.3

Model	**Evaluated**	**Baselines**		
trained on	**on length 10**	**#1**	**#2**	**#3**
DE15	63.11 (M_0)	-	-	-
JA15	72.71 (M_2)	-	-	-
PO15	**76.33** (M_0)	67.2	69.47	50.2
ES15	71.01 (M_0)	-	-	-
SL15	59.56 (M_1)	49.6	**63.34**	57.9
SV15	**69.09** (M_0)	57.0	66.79	56.6
TU15	62.63 (M_2)	-	-	-



(a) F_1 accuracies



(b) TEDEVAL accuracies

Figure 5.1: Accuracy of dependency recovery on corpora of various sentence lengths

Table 5.4: Accuracies of dependency recovery on corpora of various sentence lengths

Language	**F1**			**TEDEVAL**		
	10	**15**	**20**	**10**	**15**	**20**
AR	64.60	58.97	52.83	86.41	82.72	79.92
BU	65.76	61.51	56.27	79.82	77.78	76.59
CH	66.09	63.44	56.63	87.75	86.51	84.31
CZ	63.76	58.27	51.92	86.80	83.65	80.74
DA	63.64	58.16	54.82	86.23	82.21	79.93
DU	53.97	47.67	46.77	83.73	80.03	79.16
EN	76.81	69.00	64.86	90.55	86.05	83.43
DE	59.91	54.83	44.44	79.71	77.59	77.99
JA	72.53	65.71	62.30	84.09	80.03	78.29
PO	77.14	73.10	71.08	90.29	86.97	85.01
SL	58.44	52.09	48.80	86.46	83.17	81.62
ES	71.43	65.06	62.67	86.97	83.13	80.37
SV	66.24	58.93	56.09	86.81	83.38	81.84
TU	63.32	57.94	54.12	83.92	80.97	78.72

Table 5.5: Accuracies of dependency recovery on corpora of length 10 via different amounts of syntactic constraints

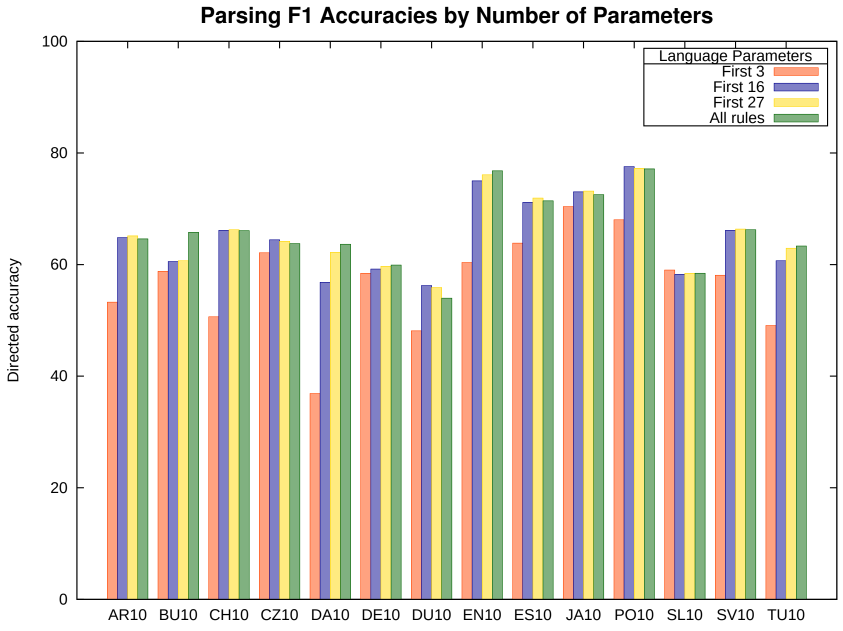
Languages	F1				TEDEVAL			
	First 3	First 16	First 27	All	First 3	First 16	First 27	All
AR10	53.26	64.83	65.14	64.60	83.04	86.50	86.73	86.41
BU10	58.78	60.55	60.69	65.76	78.92	79.96	79.70	79.82
CH10	50.65	66.13	66.24	66.09	85.67	87.61	87.79	87.75
CZ10	62.11	64.43	64.15	63.76	86.46	86.96	86.94	86.80
DA10	36.89	56.82	62.18	63.64	75.39	83.59	85.82	86.23
DU10	48.13	56.25	55.89	53.97	82.03	84.25	84.17	83.73
EN10	60.36	74.99	76.08	76.81	88.66	90.05	90.29	90.55
DE10	58.43	59.20	59.69	59.91	79.90	80.11	80.23	79.71
JA10	70.40	73.04	73.17	72.53	83.80	83.92	84.05	84.09
PO10	68.02	77.55	77.21	77.14	87.39	90.13	90.33	90.29
SL10	59.01	58.24	58.42	58.44	86.01	85.94	86.38	86.46
ES10	63.85	71.12	71.92	71.43	85.50	86.90	87.03	86.97
SV10	58.09	66.13	66.37	66.24	84.83	86.06	86.31	86.81
TU10	49.07	60.69	62.93	63.32	80.67	83.69	83.91	83.92

Moreover, the accuracy of Slovene only slightly improves when we applied all rules in the syntactic prototype. This suggests us that there are perhaps peculiar linguistic structure in Slovene that we have to take into account in the future.

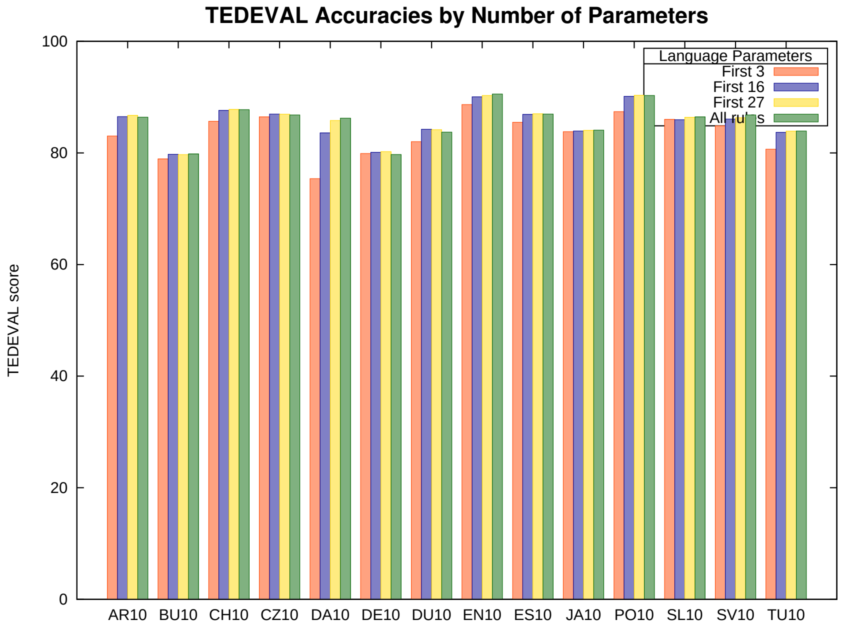
In contrast, we observe that the TEDEVAL accuracies tend to follow the hypothesis that: the more language parameters put to the syntactic prototype, the more accuracy we can achieve. From the TEDEVAL graph, it is worth noticing that the first three rules seem to cope with most frequent dependency types. The later-introduced language parameters are shown to improve the coverage in a smaller degree because the accuracy tends to saturate after the first 16 parameters.

5.3 Summary

We have presented our experiment settings and compared our experiment results with the state-of-the-art techniques. We have explained how to initialize the model parameters and described our controlled variables. Compared with the related work, our



(a) F_1 accuracies



(b) TEDEVAL accuracies

Figure 5.2: Accuracies of dependency recovery on corpora of length 10 via different amounts of syntactic constraints. We varied the amounts of given language parameters as follows: first 3 rules (groups 1-2), first 16 rules (groups 1-5), first 27 rules (groups 1-7), and all rules.

method outperforms those state-of-the-art techniques.

To answer Research Question 3: With only 10 man-hours for preparing syntactic prototypes, our method improves the accuracy of directed dependency recovery over the state-of-the-art Gillenwater et al.'s (2010) completely unsupervised parser in: (1) Chinese by 30.32% (2) Swedish by 28.96% (3) Portuguese by 37.64% (4) Dutch by 15.17% (5) German by 14.21% (6) Spanish by 13.53% (7) Japanese by 13.13% (8) English by 12.41% (9) Czech by 9.16% (10) Slovene by 7.24% (11) Turkish by 6.72% and (12) Bulgarian by 5.96%.

We have noted that although the directed dependency accuracies of some languages are below 60%, their TEDEVAL scores are still satisfactory (approximately 80%). This suggests to us that our parsed trees are, in fact, closely related to the gold-standard trees and that our dependency recovery scores are unduly depressed by the vagaries of the PASCAL annotation schemes across the subcorpora.

From the across-language average accuracies of each model, we found that the trade-off between the accuracies and the model expressivity due to the data sparsity issue is obvious. We found that Model 0 seems to outperform the others although it is the least expressive.

We have compared our results with PASCAL Challenge on linguistic structure induction. In five out of eight languages (Czech, Danish, Dutch, English, and Portuguese), our method outperforms the others, while in the other languages (Arabic, Slovene, and Swedish) our results are not significantly inferior.

We have also evaluated our method on different sentence lengths — 10, 15, and 20, to evaluate the scalability to long-tail dependencies of our language parameters. We found that, although the accuracy decreases as the sentence length increases to 15 and 20, the directed dependency accuracy decreases within the range of 10% in Bulgarian, Chinese, Danish, Dutch, Spanish, Portuguese, and Turkish.

Finally, we have studied the effects of language parameters towards the accuracy improvement. We found in almost all languages that the more language parameters we use in the syntactic prototype, the more accuracy we can achieve.

The next chapter will further attempt to analyze the performance improvement and the errors produced by our method.

Chapter 6

Error Analysis

Outline

In this chapter, we present the performance evaluation of our method when applied to each language in four ways. First, we compare the directed dependency accuracies of each model when evaluated on the corpora of length 10 and compare them to their corresponding TEDEVAL scores. Then we analyze the best model that yields the most directed dependency accuracy in terms of performance improvement according to our language parameters. We further analyze the model's capability of coping with long-tail dependencies in longer sentence lengths. Finally, we discuss the errors produced by the parser in each language in terms of over- and under-generation.

6.1 Arabic

We conduct across-the-board experiments to compare the performance of our models on Arabic whose word order is relatively rigid. From Figure 6.1(a), the simplest Model 0, a probabilistic context-free grammar, yields the best directed dependency accuracy on AR10 and it outperforms the others although there are only three word order parameters used in the syntactic prototype. The F_1 accuracy saturates quickly as we start to introduce more parameters into the syntactic prototype, and the F_1 accuracy slightly drops when we use all parameters. This trend also occurs in Model 0's TEDEVAL score in Figure 6.1(b). It should be noted that its TEDEVAL score is remarkably higher than its F_1 — indicating that our parsed trees and the Arabic gold standard ones are, in fact, closely related.

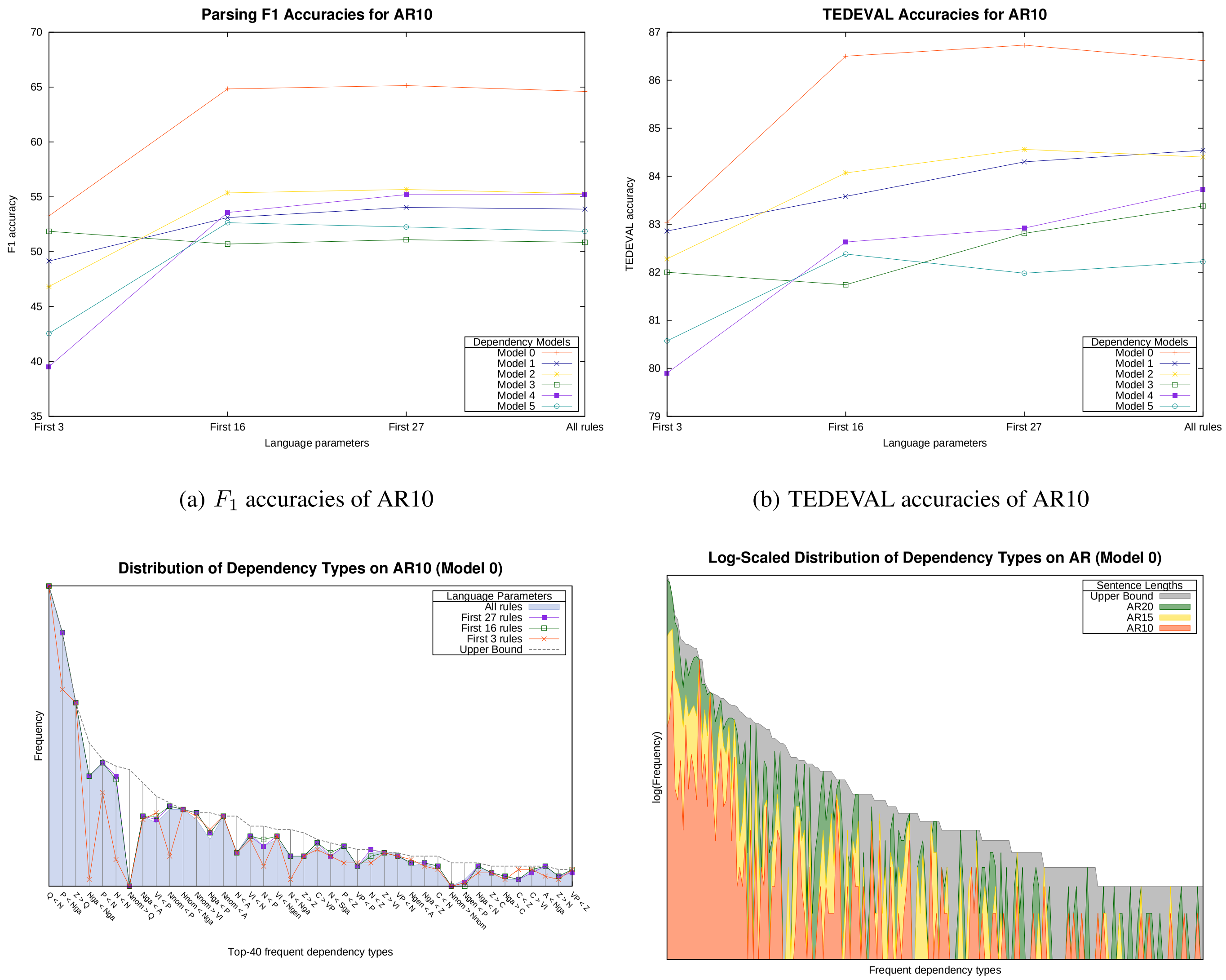
Model 0, the least expressive model, wins the competition while the rest seem to cluster below it. The order of performance is as follows: Models 0 > Models 1, 2 > Models 3, 4 > Model 5. It suggests that complex models do not always increase the coverage of long-tail dependencies. In this case, simpler models seem to outperform the more complex ones. The order of performance also shows that the lexical and headword emission probabilities (shared by Models 3, 4, and 5) are sources of error in Arabic.

The majority of our models follow the trend in which the accuracy improves when we introduce more language parameters. However, there are some distortion in the accuracy trends of Model 3 and Model 5 whose F_1 accuracies are lowest. Model 3 behaves differently from the others where its accuracy falls and later rises, as shown in Figure 6.1(b), when more language parameters are used. Model 5's behavior is akin to the other ones when considered only the F_1, but its TEDEVAL score rises and then drops when full language parameters are used.

We also study the performance improvement in Model 0 when more language parameters are introduced to the syntactic prototype. It can be seen in Figure 6.1(c) that most frequent dependency types are fully captured by the first three parameters, such as Q < N and Z > Q. When we gradually introduce more parameters, partially captured dependency types such as P < Nga, Nga < Nga, P < N, and N < N become fully captured. Incorporating first 27 rules only slightly improves the accuracy because it partially captures several more dependency types and slightly deteriorates the existing captures such as C < Z. Two dependency types Nnom > Q and Nnom > Nnom are never captured due to the discrepancy of annotation schemes.

In Figure 6.1(d) (the graph is in the log scale) we observe Model 0's capability of learning long-tail dependencies. The already captured dependency types are still captured in longer lengths but most of them are rather partially captured than fully captured due to a much larger search space. Some dependency types that do not exist in AR10 become partially captured in longer lengths 15 and 20, while the others are not captured at all.

We finally investigate the over- and under-generation of Model 0 in Arabic. We list the top-10 over- and under-generation of dependency types in Arabic of all lengths in Table 6.1. There are, in general, three categories of problems: NP dependency annotation, PP attachment, and coordinate structure annotation. On length 10, NP structure annotation errors are predominant e.g. Nnom < Q v.s. Nnom > Q and Nnom < Nnom v.s. Nnom > Nnom. The ambiguity of PP attachment starts to take place, while the coordinate structure annotation issue rarely occurs. This is due to the fact



(a) F_1 accuracies of AR10

(b) TEDEVAL accuracies of AR10

(c) Improvement of dependency type coverage on AR10

(d) Coverage of dependency types w.r.t. sentence lengths

Figure 6.1: Parsing accuracies of Arabic

that the Arabic corpus is relatively small in terms of number of sentences compared to the others and Arabic sentences tend to be long. On length 15, NP structure annotation errors are still predominant, while PP attachment and errors from coordinate structure annotation become more frequent e.g. N < P, VP < P, C < VP, and C > VI. On length 20, errors from coordinate structure annotation and PP attachment become predominant, suggesting that they are the cause of the accuracy deterioration on longer sentence lengths.

6.2 Bulgarian

We study the performance of our system in parsing Bulgarian whose word order is quite flexible. In Figure 6.2(a), we found that Model 2 outperforms the others in terms of F_1 when the full set of language parameters are put into the syntactic prototype. The accuracy of Model 2 slowly grows as we incorporate more language parameters and rises when we use all parameters. This trend contradicts its TEDEVAL performance tendency where it performs the second worst while the accuracy saturates after the first 16 language parameters are used. This signifies striking discrepancy between our parse trees and the gold standard ones.

Model 2 wins the competition while the rest seem to cluster below it. The order of performance is as follows: Models 1, 2 > Model 0 > Model 3 > Model 4 > Model 5. It suggests that complex models do not always increase the coverage of long-tail dependencies. In this case, simpler models seem to outperform the more complex ones. The order of performance also shows that generative dependency models perform well on Bulgarian while lexical and headword emission probabilities (in Model 5) deteriorate the accuracy.

In terms of F_1, the majority of the models (Models 0, 3, 4, 5) follow the trend in which the accuracy rapidly increases as we incorporate the first three language parameters, then starts to saturate at the first 27 parameters, and finally drops when all parameters are used. Contrarily, their TEDEVAL scores slowly decay until the first 27 parameters and suddenly rise when all parameters are used. Model 0's TEDEVAL performance slightly drops instead of rising when all parameters are used. Model 1 and Model 2, both being generative dependency models, behave similarly considering their F_1 scores, but they remarkably differ from each other in terms of TEDEVAL. Model 1's TEDEVAL score rises until the first 16 parameters are used then increasingly deteriorates.

Table 6.1: Top-10 errors in Arabic

(a) Over-generation

Type on AR10	Freq	Type on AR15	Freq	Type on AR20	Freq
Nnom < Q	35	Nnom < Q	37	C < VP	64
Nnom < A	20	N < P	31	N < P	52
Nnom < P	13	P > N	24	P > N	43
Nnom < Nnom	7	Nnom < A	22	Nnom > Q	39
VI < P	6	VP < N	19	SD < N	32
Nga < P	6	SD < N	19	Nnom < P	30
Nnom < Nga	5	C < VP	19	N < C	28
Nnom < Z	4	C > VI	19	N < A	28
Ngen < P	4	N < C	17	C > VI	28
N < C	4	C > VI	17	C < VI	28

(b) Under-generation

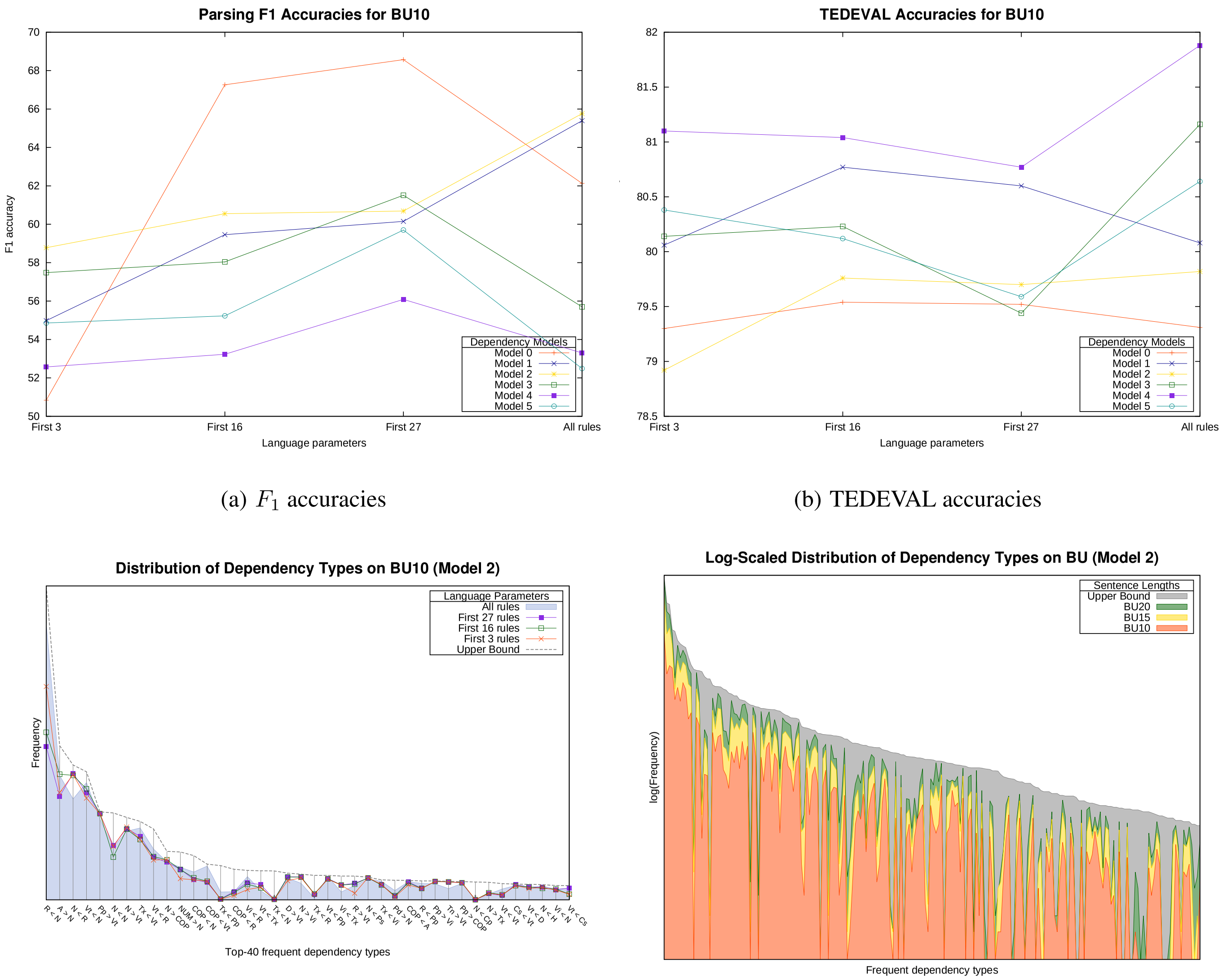
Type on AR10	Freq	Type on AR15	Freq	Type on AR20	Freq
Nnom > Q	35	Nnom > Q	35	VP < P	61
N < A	11	Nga < P	25	C > VP	43
Nga < Nga	10	Nga < A	25	VI < P	36
Nga < A	10	VP < P	23	Nnom > Q	35
VI < P	8	VI < P	20	P < C	33
Nga < P	8	Nga < Nga	16	VP < C	30
N < Nga	8	P < C	15	Nga < P	29
Z > Z	7	N < N	14	Nga < A	27
Nnom > Nnom	7	N < A	14	N > C	27
N < P	6	N < Nga	12	FN > VI	25

It is obvious that the less frequent language parameters (e.g. copula, gerunds, and prodrops) account for the rapid accuracy improvement when all language parameters are used. Since Model 1 and Model 2 are both generative dependency models, we hypothesize that the role-emission probabilities can cope with Bulgarian's quite flexible word order, and the search space of Bulgarian can be greatly restrained by those parameters. However, our annotation scheme for the gerundial structures differs from that of the gold standard, resulting in slightly reduced TEDEVAL scores.

We study the performance improvement in Model 2 when more language parameters are introduced to the syntactic prototype. Figure 6.2(c) shows that most frequent dependency types, such as R < N, A > N, Vt < N, and Tx < Vt, are partially captured and they are further, yet only marginally, captured as more language parameters are introduced. Finally frequent dependency types are dramatically captured by the help of the search space restrained by the less frequent language parameters. Only a few dependency types are less captured when the full set of language parameters are used, such as N < R and Tx < Pp.

In Figure 6.2(d), Model 2 is capable of learning long-tail dependencies. Given the full set of language parameters, the already captured dependency types in BU10 is still captured in the longer sentence lengths and a few dependency types are not captured until longer lengths 15 and 20. Some dependency types are always neglected in all sentence lengths as a result of the discrepancy of annotation schemes.

We finally investigate the over- and under-generation of Model 2 in Bulgarian. We list the top-10 over- and under-generation of dependency types in Bulgarian of all lengths in Table 6.2. There are, in general, four categories of problems: NP dependency annotation, PP attachment, PP dependency annotation, and dependency annotation of the copula. On length 10, PP attachment, the predominant problem, starts to take place as seen in Vt < R and N < R. NP and PP dependency annotation also cause errors such as A < N v.s. A > N and R > N v.s. R < N. There is also ambiguity in determining the copulative structure. On length 15, PP attachment is predominant, while NP and PP annotation errors become more frequent and the copulative structure is less used. On length 20, NP structure annotation errors and PP attachment become predominant, suggesting that they are the cause of the accuracy deterioration on longer sentence lengths.



(a) F_1 accuracies

(b) TEDEVAL accuracies

(c) Improvement of dependency type coverage

(d) Coverage of dependency types w.r.t. sentence lengths

Figure 6.2: Parsing accuracies of Bulgarian

Table 6.2: Top-10 errors in Bulgarian

(a) Over-generation

Type on BU10	Freq	Type on BU15	Freq	Type on BU20	Freq
Vt < R	598	Vt < R	1499	A < N	3108
Vt < N	563	Vt < N	1405	Vt < R	2195
A < N	367	A < N	1019	Vt < N	2079
Pp > Vt	314	Cp > N	857	Cp > N	1309
Cp > N	284	Pp > Vt	567	NUM < N	1122
N < R	276	Vt > Tx	517	R < A	1103
Vt > Tx	270	NUM < N	501	Vt < Vt	814
COP < A	224	N < R	494	Vt > Tx	788
Vi > Tx	199	Vt < Vt	479	Pp > Vt	747
R > N	186	COP < A	425	Vi < R	710

(b) Under-generation

Type on BU10	Freq	Type on BU15	Freq	Type on BU20	Freq
R < N	482	N < N	1521	N < N	2651
N < R	442	N < R	1084	A > N	1746
N < N	442	Tx < N	906	R < N	1577
A > N	389	Tx < R	797	N < R	1523
Tx < N	375	R < N	755	Tx < N	1463
Tx < Pp	343	N < Cp	655	Tx < R	1345
Tx < R	297	A > N	639	N < Cp	1155
COP < R	293	Tx < Pp	571	COP < R	906
Vt < R	262	COP < R	515	Tx < Pp	890
Vt < Tx	260	Vt < R	420	Vt < R	820

6.3 Chinese

Across-the-board experiments are conducted to show the performance of our system on Chinese, whose word order is rigid. From Figure 6.3(a), Model 0 significantly outperforms the other models after the first 16 language parameters are used. The F_1 accuracy rises and seems to saturate when more language parameters are provided. This also occurs in Model 0's TEDEVAL score but it increases in a much narrower range. The TEDEVAL is also remarkably high although there are only three first parameters in the syntactic prototype and it still rises when more parameters are given. This suggests us that our parsed trees and the gold standard trees are closely related.

Model 0 wins the competition while the rest seem to cluster below it. The order of performance is as follows: Model 0 > Model 5 > Models 2, 3, 4 > Model 1. It suggests that complex models do not always increase the coverage of long-tail dependencies. In this case, simpler models seem to outperform the more complex ones. The order of performance also shows that the lexical emission probability (in Model 1) is a source of error.

The majority of the models follow this trend where the F_1 accuracy quickly increases and starts to saturate or slightly drop when the first 16 rules are used, and TEDEVAL scores also closely follow this trend. In Figure 6.3(b), the accuracies of Model 3 and Model 4 drop when the entire set of language parameters are used, as shown by their TEDEVAL accuracies. Model 4 seems to be more sensitive to noise than Model 3 as the decrease of the first's accuracy is larger than that of the latter. In constrast, Model 5 continues to rise when all the language parameters are used. Since Model 4 uses the headword emission probability while Model 3 uses the lexicon emission probability, we therefore infer that data sparsity in such probabilities may account for the decrease of accuracy in these models.

As shown in Figure 6.3(c), the performance of Model 0 is improved by the language parameters. Given the first three parameters, most of frequent dependency types are partially captured such as NN > NN, DE > NN, and NN > DE, resulting in low F_1 accuracy. When we introduce more rules to the syntactic prototype, these dependency types become much more captured. Less frequent dependency types, on the other hand, tend to be captured because of the use of the first three parameters. Incorporating the full set of language parameters does not significantly improve the accuracy as the first 27 parameters have already captured frequent dependency types.

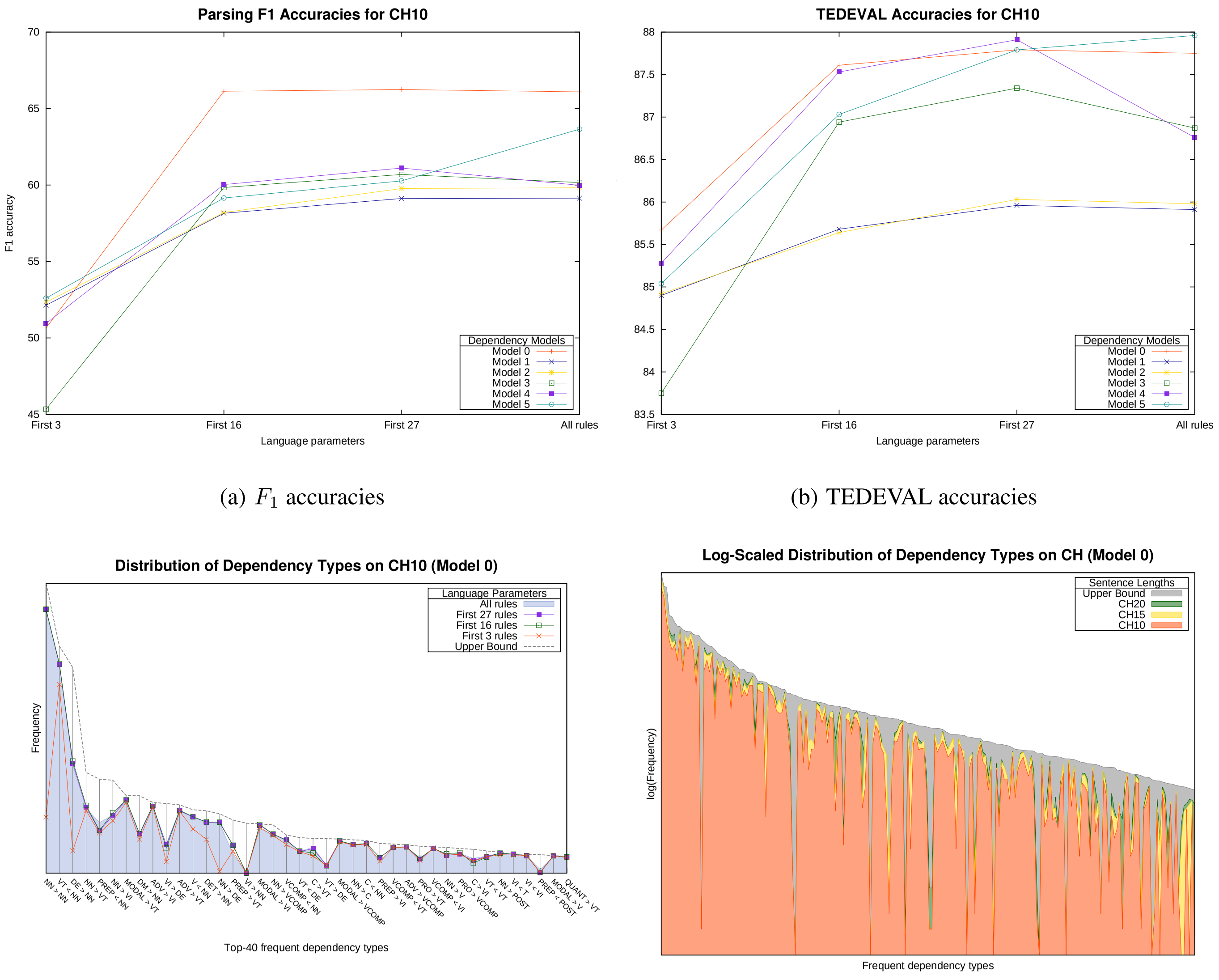
The log-scaled distribution in Figure 6.3(d) shows that a few new long-tail dependency types are captured by Model 0 while the previously captured ones are further captured. Despite the volume of the Chinese corpus, most sentences are relative short at length, resulting in less number of long-tail dependencies to be captured. Our model nevertheless does capture less frequent dependency types in longer lengths 15 and 20 but they are quite rare.

We finally investigate the over- and under-generation of Model 0 in Chinese. We list the top-10 over- and under-generation of dependency types in Chinese of all lengths in Table 6.3. There are, in general, four categories of problems: sentence-like NP ambiguity (SNP), PP attachment, PP dependency annotation, and dependency annotation of the auxiliary. On length 10, sentence-like NP ambiguity is the predominant problem, such as VI < NN v.s. VI > NN. This is because the particle DE can be used as either an adverb or a relative pronoun, resulting in two analyses of VI DE NN: $[_{\text{NP}}$ $[_{\text{RELC}}$ VI DE] NN] and $[_{\text{VP}}$ $[_{\text{VP}}$ VI DE] NN]. Errors from PP dependency annotation also occurs, i.e. PREP > NN v.s. PREP < NN. The auxiliary in Chinese are mis-identified as the head as NN > MODAL. We expect PP attachment to take place as well, but because the preposition is assigned not to be the head of the PP, the problem becomes structurally hidden. On length 15, sentence-like NP ambiguity is still predominant, while PP attachment, and errors from PP dependency annotation and the auxiliary annotation become more frequent. On length 20, errors from annotating sentence-like NPs and auxiliary become predominant, suggesting that they are the cause of the accuracy deterioration on longer sentence lengths.

6.4 Czech

We assess across the table our models on parsing Czech, a morphologically rich language with a flexible word order and non-projective dependency. As shown in Figure 6.4(a), Model 1 yields the best F_1 accuracy on CZ10 and it outperforms the others even though only the first three language parameters are used in the syntactic prototype. The accuracy rises to its maximum when we incorporate the first 16 parameters, and slowly decreases when more parameters are incorporated. Model 1's TEDEVAL closely follows this trend, but it decreases in a smaller range. Our Czech parsed trees are very close to the gold standards regardless of the discrepancy of annotation schemes as the TEDEVAL scores are quite satisfactory.

Model 1 wins the competition while the rest seem to cluster below it. The order of performance is as follows: Models 0, 1 > Models 3, 5 > Models 2, 4. It suggests



(a) F_1 accuracies

(b) TEDEVAL accuracies

(c) Improvement of dependency type coverage

(d) Coverage of dependency types w.r.t. sentence lengths

Figure 6.3: Parsing accuracies of Chinese

Table 6.3: Top-10 errors in Chinese

(a) Over-generation

Type on CH10	Freq	Type on CH15	Freq	Type on CH20	Freq
VI < NN	5168	VI < NN	6453	VI < NN	8722
VI < DE	2990	VI < DE	3919	VI < DE	4619
PREP > NN	2219	PREP > NN	3124	VT < DE	3429
VT < DE	1974	VT < DE	2641	NN > MODAL	3310
VI < VT	1707	NN > DE	2177	VT < NN	2567
NN > MODAL	1696	VI < VT	2138	VI < VT	2418
VI < VI	1674	NN > MODAL	2128	VI < VI	2291
VT < VI	1660	VT < VI	2039	NN > DE	2270
NN > DE	1629	VI < VI	2028	VT < VI	2173
VT < NN	1394	VT < NN	1794	NN > VI	2058

(b) Under-generation

Type on CH10	Freq	Type on CH15	Freq	Type on CH20	Freq
DE > NN	6709	DE > NN	8436	NN > NN	11817
VI > NN	3436	VI > NN	4364	DE > NN	11099
PREP < NN	2983	PREP < NN	3908	DM > NN	4618
VI > DE	2831	VI > DE	3688	VI > NN	4568
DM > NN	2718	DM > NN	3435	VI > DE	3870
NN > VT	2467	NN > VT	3314	PREP < NN	3438
NN > VI	2405	PREP > VT	3308	DET > NN	3391
PREP > VT	2322	NN > VI	2785	NN > VT	2941
NN > NN	1831	VT > DE	2615	PREP > VT	2897
VT > DE	1821	NN > NN	2169	VT > DE	2528

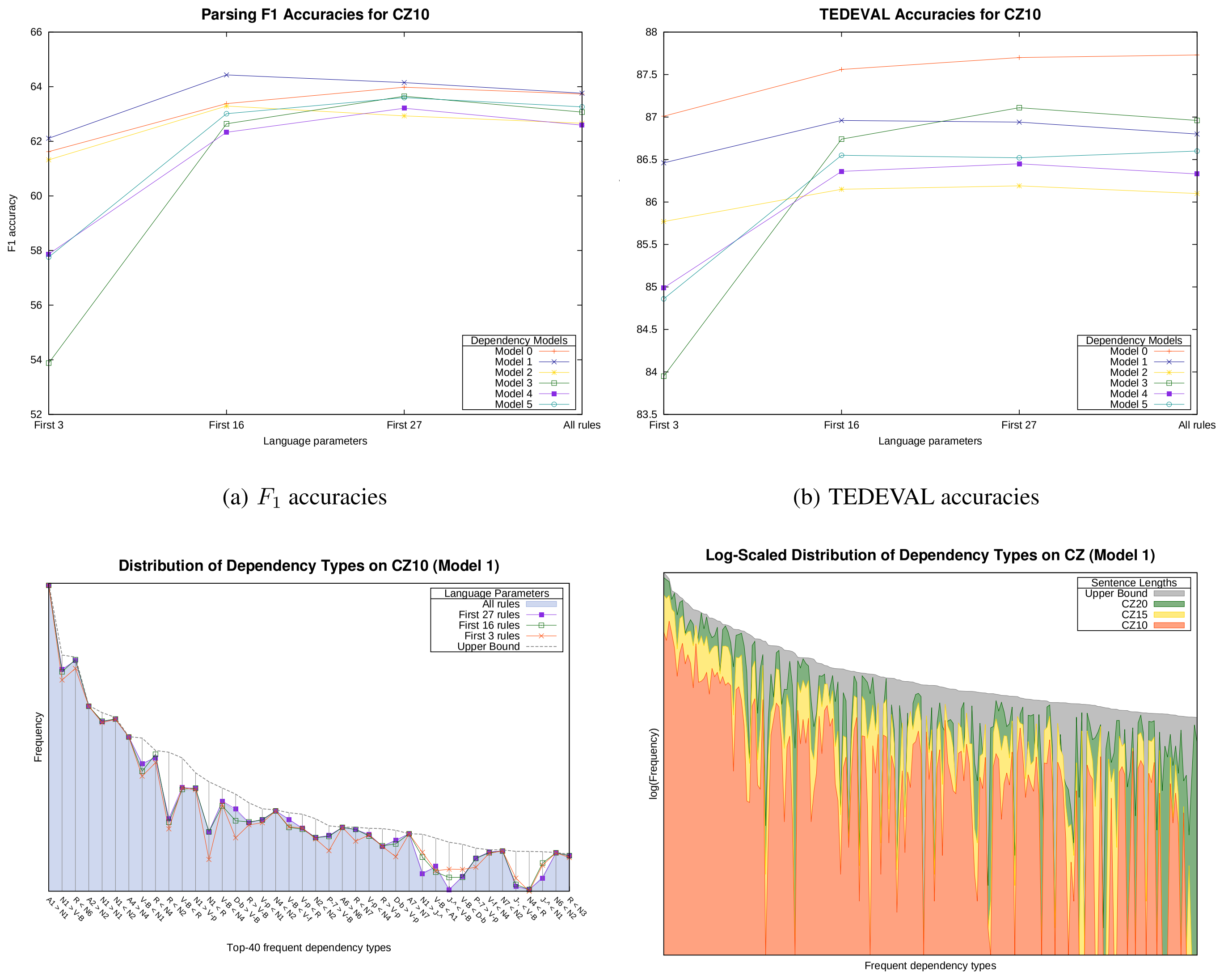
that complex models do not always increase the coverage of long-tail dependencies. In this case, simpler models seem to outperform the more complex ones. The order of performance also shows that the mother-daughter probability (shared by Models 2 and 4) is a source of error.

The majority of our models follow this trend as shown in Figure 6.4(b) but the accuracy starts to decay much later than Model 1. We observe that Model 0 and Model 1 perform well both in terms of F_1 and TEDEVAL. More complex models — i.e. Model 2, Model 3, Model 4, and Model 5 — tend to start out with a lower accuracy when only the first three language parameters are used in the syntactic prototype. The accuracy significantly increases as we feed the first 16 parameters and starts to saturate when even more parameters are used. These complex models, as well as Model 2, yield slightly decaying accuracy when the full set of language parameters are incorporated to the syntactic prototype. We hypothesize that Czech's flexible word order accounts for the deterioration of the accuracy in these complex models. Models 3, 4, and 5 make use of the lexicon emission probability π_{lex} and the headword emission probability π_{head} while Model 2 utilizes the daughter role emission probability π_{dtr}, all of which based on the probability of words and syntactic categories. The flexible word order may have introduced a lot of syntactic categories in the lexicon inventory, exacerbating the data sparsity issue.

Model 1 in Figure 6.4(c) shows that most of frequent dependency types are partially or fully captured by the first three language parameters such as N1 > V-B, R < N6, V-B < N1, and N1 < R. When increasing the number of parameters used, Model 1 still carries on capturing more dependency types. There are two interesting dependency types: J-^ < V-B and N4 < R as they are not captured. First, J-^ < V-B is captured partially by the first three rules, but it is later less captured as we gradually increase the number of the parameters used. N4 < R is never captured even though more language papers are used.

In longer lengths, previously captured dependency types are further captured in longer lengths 15 and 20 as depicted in Figure 6.4(d). Model 1 can also capture some new dependency types while the rest are not captured at all.

We finally investigate the over- and under-generation of Model 1 in Czech. We list the top-10 over- and under-generation of dependency types in Czech of all lengths in Table 6.4. There are, in general, three categories of problems: NP dependency annotation, PP attachment, and coordinate structure dependency annotation. On length 10, errors from coordinate structure annotation are prevent, such as V-B < V-B v.s. J-^ <



(a) F_1 accuracies

(b) TEDEVAL accuracies

(c) Improvement of dependency type coverage

(d) Coverage of dependency types w.r.t. sentence lengths

Figure 6.4: Parsing accuracies of Czech

V-B and V-c < V-p v.s. V-c > V-p. The NP dependency annotation is erroneous such as N1 > N1 and N1 > NX. Of course, PP attachment has been expected and it is observed as V-B < R, N1 < R, and N4 < R. On length 15, errors from the coordinate structure annotation are still predominant, while errors from NP structure annotation and PP attachment are found. On length 20, PP attachment and errors from NP structure annotation become predominant, suggesting that they are the cause of the accuracy deterioration on longer sentence lengths.

6.5 Danish

Experiments for performance assessment are conducted to compare our models in Danish, a language with a rigid word order. In Figure 6.5(a), Model 5, the most complex model of ours, yields the best directed dependency accuracy. The F_1 accuracy quickly rises and starts to saturate as we introduce more language parameters to the syntactic prototype. Model 5's TEDEVAL score also follows this trend but it is not the best model among the others. Although its F_1 accuracy is below 65%, its TEDEVAL score is still satisfactory, indicating that our parsed trees are in fact closely related to the gold standard ones.

Model 5 wins the competition while the rest seem to cluster below it. The order of performance is as follows: Model 5 > Model 4 > Model 1 > Model 0, 2 > Model 3. This seems to follow our hypothetical trend where more complex models capture long-tail dependency types more accurately. The order of performance also shows that the lexical-emission probability (in Model 3) is a source of error.

When compared with Figure 6.5(b), the majority of our models follow the trend where the F_1 and TEDEVAL accuracies quickly rise and start to saturate as more language parameters are incorporated into the syntactic prototype. At the full set of language parameters, the range of the TEDEVAL scores is within only 2% while that of the F_1 scores is more than 5%.

Figure 6.5(c) shows the performance improvement of Model 5. It can be seen that the first three rules are not as capable of capturing frequent dependency types in DA10 as we expected. SP < NC, VA < SP, VA < AN, NC > VA, and PI < NC are examples of frequent dependency types that are only partially captured. These dependency types however are better captured when we incorporate more language parameters into the syntactic prototype. The capture of a few frequent dependency types e.g. AN > NC and NP > NP, however, deteriorates as we increase the number of parameters. We notice

Table 6.4: Top-10 errors in Czech

(a) Over-generation

Type on CZ10	Freq	Type on CZ15	Freq	Type on CZ20	Freq
V-B < V-B	945	V-B < V-B	2992	R > V-B	4857
N1 > N1	923	R > V-B	2413	V-B < V-B	4239
N1 < NX	890	V-B < R	2161	V-c < V-p	3376
V-B < R	809	V-c < V-p	1893	R < R	3371
R > V-B	755	N1 > N1	1868	J-^ > V-B	2888
J-^ > V-B	733	N1 > V-B	1651	R > V-p	2801
V-c < V-p	642	V-p < R	1569	N1 > V-B	2357
N1 < R	560	J-^ > V-B	1488	V-B < V-p	2240
N1 < C-=	544	V-p < V-p	1419	V-B < V-f	2107
V-p < R	527	V-B < V-f	1312	D-b > V-B	2089

(b) Under-generation

Type on CZ10	Freq	Type on CZ15	Freq	Type on CZ20	Freq
R < N2	1177	R < N2	3688	V-B < R	4929
N1 < R	875	N1 < R	2308	N1 > N1	4783
J-^ < V-B	842	J-^ < V-B	1998	R < N2	4581
N4 < R	681	N4 < R	1931	N4 < R	4023
N1 > J-^	674	J-, < V-B	1873	V-B < N1	3713
N1 > NX	659	V-c > V-p	1820	V-c > V-p	3683
V-c > V-p	621	V-B < R	1776	V-p < R	3214
J-, < V-B	620	N1 > J-^	1759	N2 < R	3160
N1 < N1	608	N2 < R	1598	J-, < V-B	2947
V-B < R	582	V-B < N1	1591	J-^ < V-B	2759

that these dependency types do not correspond to the annotation scheme of Danish noun phrases in which attachment directions are assigned in the opposite direction (such as AN < NC and NP < NP).

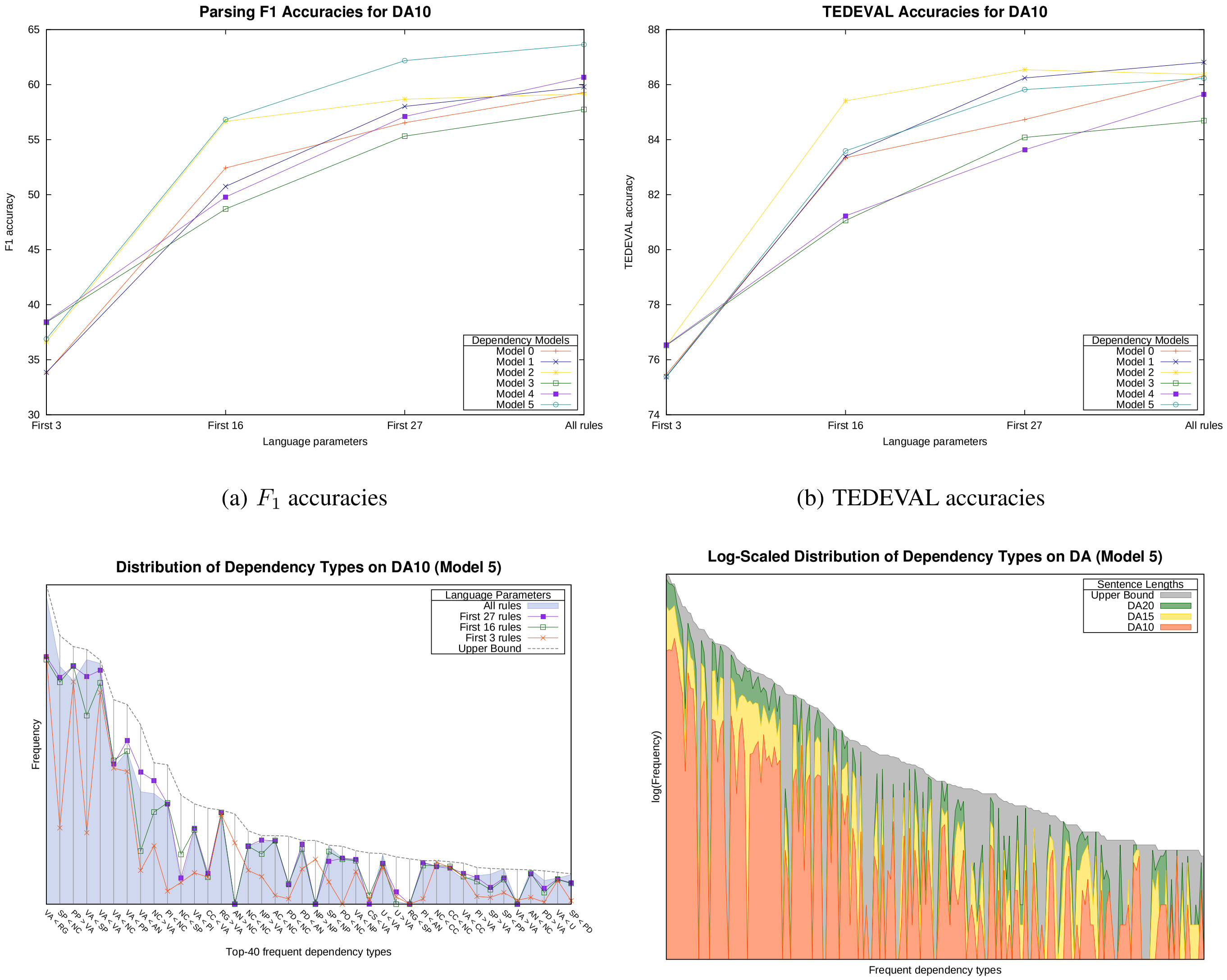
As shown in the log-Zipfian distribution in Figure 6.5(d), Model 5 is able to steadily capture long-tail dependency types from lengths 15 and 20. New less-frequent dependency types are captured in DA15 most of the time. Some dependency types (AN > NC and NP > NP) are always neglected in all sentence lengths as a result of the discrepancy of the annotation schemes.

We finally investigate the over- and under-generation of Model 5 in Danish. We list the top-10 over- and under-generation of dependency types in Danish of all lengths in Table 6.5. There are, in general, three categories of problems: NP dependency annotation, PP attachment, and coordinate structure annotation. On length 10, errors from the NP dependency annotation are the predominant problem, such as AN < NC v.s. AN > NC and NP < NP v.s. NP > NP. PP attachment is also observed as VA < SP and NC < SP, while errors from coordinate structure annotation can be seen e.g. CC > VA v.s. CC < VA, and VA < VA which has no counterpart due to the discrepancy of such head assignment. On length 15, errors from NP dependency annotation are still predominant, while PP attachment and errors from coordinate structure annotation are still found. On length 20, the same proportion of the problems is still preserved, suggesting that they are the cause of the accuracy deterioration on longer sentence lengths.

6.6 Dutch

Despite its fixed word order, Dutch is a challenging language for the task of grammar induction due to its renowned widespread use of non-projective dependency. We study the performance of our models in parsing Dutch. From Figure 6.6(a), Model 0 yields the best F_1 accuracy after using the first 16 parameters. Model 0, as well as the other models follow the same trend in which the F_1 accuracy rises to its maximum when the first 16 parameters are used and then decays. Model 4 continues to rise when the first 27 rules are used and also starts to decay.

Model 0 wins the competition while the rest seem to cluster below it. The order of performance is as follows: Models 0, 5 > Model 4 > Model 3 > Model 2 > Model 1. When taking a look at TEDEVAL, we found that Model 5's score is higher than that of Model 0. This seems to follow our hypothetical trend where more complex



(a) F_1 accuracies

(b) TEDEVAL accuracies

(c) Improvement of dependency type coverage

(d) Coverage of dependency types w.r.t. sentence lengths

Figure 6.5: Parsing accuracies of Danish

Table 6.5: Top-10 errors in Danish

(a) Over-generation

Type on DA10	Freq	Type on DA15	Freq	Type on DA20	Freq
AN < NC	383	AN < NC	917	AN < NC	1568
VA < SP	215	VA < VA	457	VA < VA	744
VA < VA	154	VA < SP	353	VA < SP	568
CC > VA	116	RG < SP	281	U < VA	465
NP < NP	112	U < VA	246	RG < SP	461
PP > RG	93	NP < NP	242	U < VA	416
RG < AN	92	VA < RG	220	RG < SP	356
U < VA	88	PP > RG	195	NP < NP	316
AN < AN	81	CC > VA	187	VA < RG	290
VA < RG	73	RG < AN	180	AN < AN	290

(b) Under-generation

Type on DA10	Freq	Type on DA15	Freq	Type on DA20	Freq
AN > NC	187	NC < SP	560	NC < SP	1047
NC < SP	183	CS < VA	385	CS < VA	770
VA < NC	141	AN > NC	373	AN > NC	633
CC < VA	139	CC < VA	303	U > VA	547
NP > NP	136	U > VA	296	NP > NP	526
CS < VA	132	NP > NP	292	CC < VA	504
PD < NC	106	VA < NC	282	PD < NC	498
VA < PP	100	VA < AN	279	PI < NC	453
RG < SP	97	PD < NC	274	VA < NC	413
U > VA	94	PI < NC	243	VA < NC	388

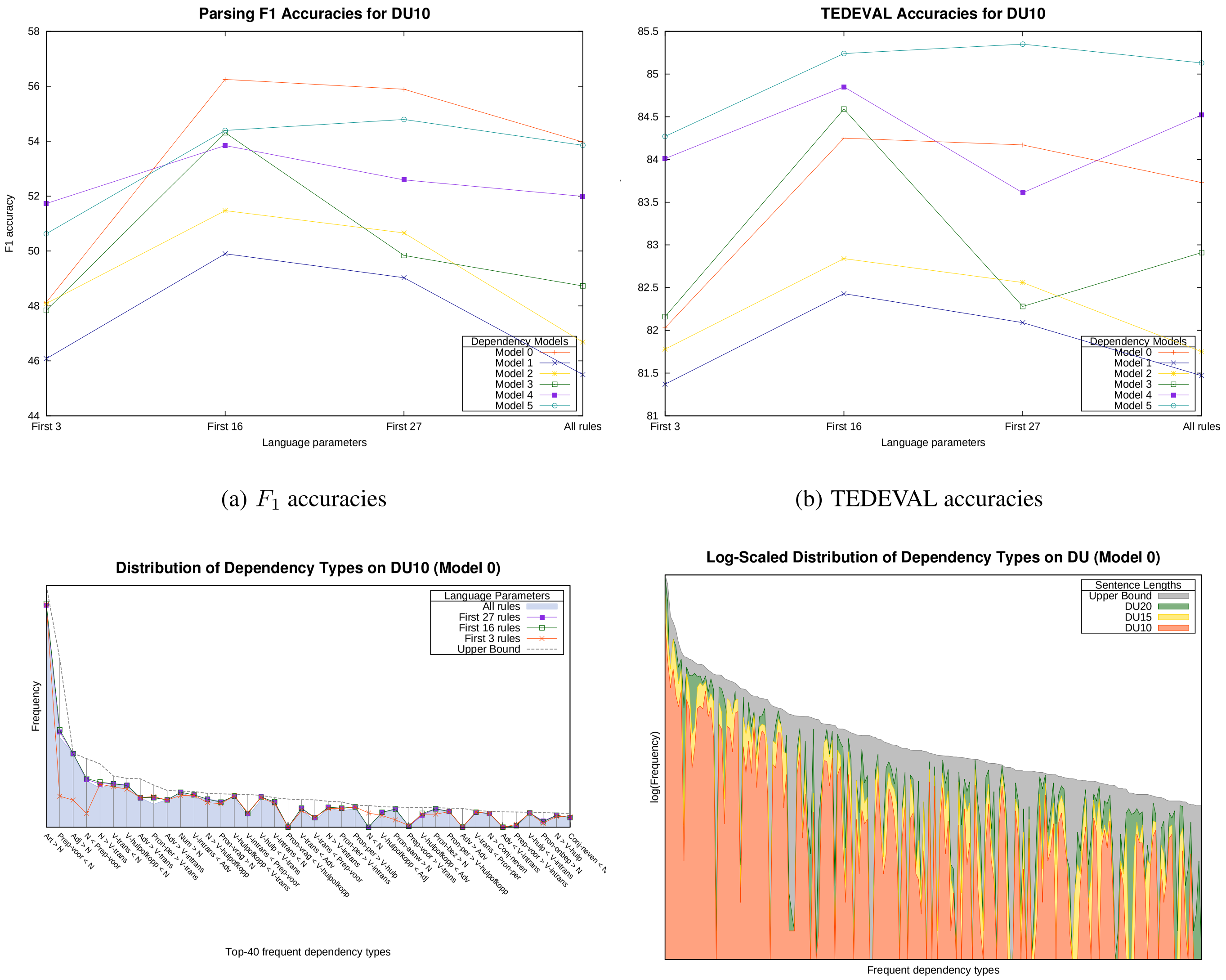
models capture long-tail dependency types more accurately. The order of performance also shows that the generative dependency probability (shared by Models 1 and 2) is a source of error.

When taken Figure 6.6(b) into account we found that the majority of our models also follow this trend and its range of change is similar to that of F_1 accuracies. However, Model 3 and Model 4's TEDEVAL accuracies manifest another trend where they rise at the first three parameters, drop at the first 16 parameters, and rise again when the full set of parameters are used. The full set of language parameters allow Model 3 and Model 4 to construct parsed trees that are once again close to the gold standard ones.

We also investigate the performance improvement of Model 0 in Figure 6.6(c). The first three rules rather capture less frequent dependency types such as Art > N, N > V-trans, V-trans < N, and V-hulpofkopp < N, while more frequent ones are marginally captured such as Prep-voor < N, Adj > N, and N < Prep-voor. When incorporating more language parameters, these dependency types become partially or even fully captured. We also notice that a few dependency types are always neglected no matter how many language parameters we use. The dependency type N < N is intriguing in that it is more captured by the first three parameters than by the full set because it is against the general annotation scheme of Dutch.

Figure 6.6(d) shows that the previously captured dependency types are still captured by Model 0. Interestingly, most of later-found less frequent dependency types are rather captured in length 20 than in length 15. The neglected dependency types are always neglected in longer lengths 15 and 20.

We finally investigate the over- and under-generation of Model 5 in Dutch. We list the top-10 over- and under-generation of dependency types in Dutch of all lengths in Table 6.6. There are, in general, four categories of problems: PP dependency annotation, NP dependency annotation, ambiguous multiword unit (MWU), and PP attachment. On length 10, errors from PP dependency annotation are the predominant problem as seen in the preposition combined with the article (Prep-voor > Art) and as in Pron-vrag > V-hulpofkopp v.s. Pron-vrag < V-hulpofkopp. Errors from NP dependency annotation also take place as in N > N v.s. N < N. MWU can perform as many parts of speech causing syntactic ambiguity. On length 15, the same set of the problems are still found throughout the errors. On length 20, PP attachment suddenly dominates the errors and the other problems slightly fade out. We hypothesize that the data sparsity issue is remedied by more data in longer lengths. This suggests that these



(a) F_1 accuracies

(b) TEDEVAL accuracies

(c) Improvement of dependency type coverage

(d) Coverage of dependency types w.r.t. sentence lengths

Figure 6.6: Parsing accuracies of Dutch

problems are the cause of the accuracy deterioration on longer sentence lengths.

6.7 English

We revisit the study of grammar induction performance of English, having a fixed word order, by conducting across-the-board experiments. As illustrated in Figure 6.7(a), Model 5 always outperforms the others even when only the first three language parameters are given. All of our models follow the trend in which the accuracy rapidly increases when the first 16 parameters are introduced, then it starts to saturate as more language parameters are used. When TEDEVAL accuracies in Figure 6.7(b) are taken into consideration, almost all models also follow this trend excluding Model 4 as its TEDEVAL accuracy drops marginally when the full set of language parameters are used.

Model 5 wins the competition while the rest seem to cluster below it. The order of performance is as follows: Model 5 > Models 0, 1, 2 > Models 3, 4. This seems to follow our hypothetical trend where more complex models capture long-tail dependency types more accurately. The order of performance shows that the lexical and headword generation probability (shared by Models 3 and 4) are sources of error.

We also investigate the performance improvement of English parsing. As shown in Figure 6.7(c), Model 5 with the first three language parameters can capture some frequent dependency types such as DT > NN, JJ > NN, and NN > VBZ. Meanwhile, some dependency types are not captured at all — e.g. NNP > NNP, NN > NN, and NN < IN. When adding more parameters to the syntactic prototype, these dependency types become fully or partially captured. It is however noticeable that the attachment direction for two NNP's was mistakenly assigned by the first three rules but it later becomes correctly assigned when more parameters are used.

From Figure 6.7(d), long-tail dependencies can also be captured in longer lengths 15 and 20. New less frequent dependency types are captured in longer lengths while the previously captured dependency types are futher captured.

We finally investigate the over- and under-generation of Model 5 in English. We list the top-10 over- and under-generation of dependency types in English of all lengths in Table 6.7. There are, in general, three categories of problems: PP attachment, NP dependency annotation, and dependency annotation of cardinal numbers (CD). On length 10, PP attachment is predominant as seen in VBZ < IN, NN < IN, VBN < IN, and NNS < IN. There are also discrepancies in annotating NP structures such as NNP > NNP v.s.

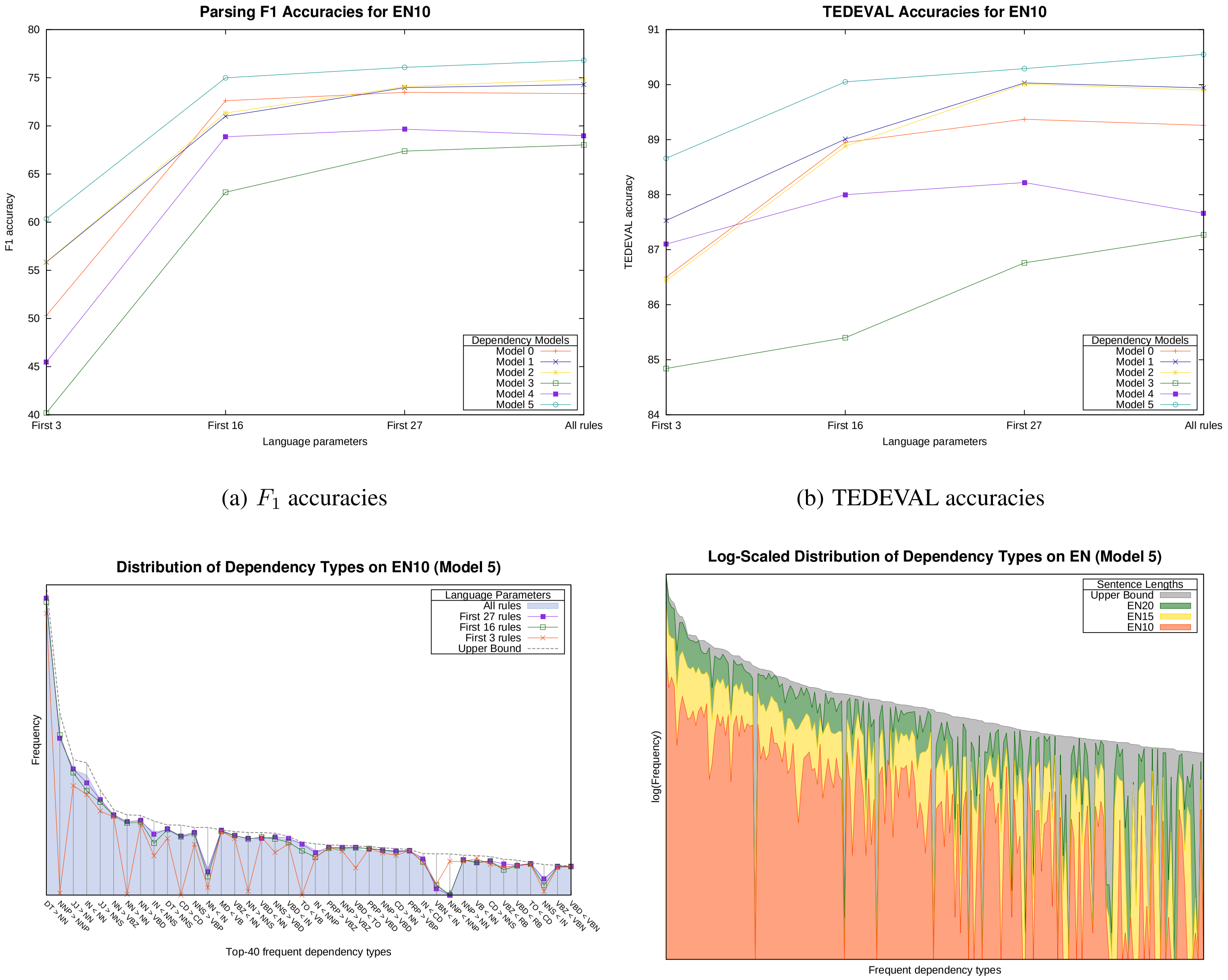
Table 6.6: Top-10 errors in Dutch

(a) Over-generation

Type on DU10	Freq	Type on DU15	Freq	Type on DU20	Freq
Prep-voor > Art	794	Prep-voor > Art	2835	N > Prep-voor	1997
N > N	643	N > N	2230	V-hulpofkopp < Prep-voor	913
Pron-vrag > V-hulpofkopp	521	Art > MWU	732	Prep-voor > V-hulpofkopp	840
V-hulpofkopp < MWU	455	N > MWU	682	N > N	821
Art > MWU	400	N > V-hulpofkopp	596	V-trans > V-trans	817
N > V-hulpofkopp	356	N > V-trans	581	V-trans < N	765
Adv > V-intrans	339	Pron-vrag > V-hulpofkopp	570	Adv > V-hulpofkopp	746
N > MWU	309	V-intrans < N	568	N > V-intrans	727
N > V-trans	306	V-trans < N	542	V-hulpofkopp < Adv	655
N > V-intrans	291	V-hulpofkopp < MWU	533	N > V-trans	655

(b) Under-generation

Type on DU10	Freq	Type on DU15	Freq	Type on DU20	Freq
Prep-voor < N	1241	Prep-voor < N	3457	N < Prep-voor	3620
Pron-vrag < V-hulpofkopp	444	N < Prep-voor	1408	Prep-voor < N	3138
N > V-trans	380	Prep-voor > V-trans	890	Adj > N	1227
N < Prep-voor	350	V-trans < Prep-voor	740	Prep-voor > V-trans	1123
V-intrans < Prep-voor	348	V-intrans < Prep-voor	676	N < N	1004
N < N	343	Adv > V-trans	664	N > V-trans	987
V-trans < Prep-voor	327	N < N	658	Adv > V-trans	966
Art > N	304	N > V-trans	618	Pron-vrag > N	750
Prep-voor > V-trans	303	Art > N	563	Art > N	725
Pron-per > V-trans	301	Prep-voor > V-intrans	512	Adv > Adv	711



(a) F_1 accuracies

(b) TEDEVAL accuracies

(c) Improvement of dependency type coverage

(d) Coverage of dependency types w.r.t. sentence lengths

Figure 6.7: Parsing accuracies of English

NNP < NNP. Cardinal numbers when modifying a noun phrase are mistagged as in NNP > CD v.s. NNP < CD. On length 15, some PP in a series of PPs starts to perform as an NP which is then modified by the succeeding PP, as in IN < IN. PP attachment and the other problems are also found on length 15. On length 20, the problem of PP turning to an NP is found more frequently deteriorating the parsing accuracy.

6.8 German

Performance of grammar induction for German, a rigid word-order language with mild use of non-projective dependency, is demonstrated by several experiments. Figure 6.8(a) shows that Model 0 yields the best F_1 directed dependency accuracy at approximately 60%. It follows the hypothetical trend in which the performance increases as we introduce more language parameters into the syntactic prototype. On the contrary Model 0's TEDEVAL score is the lowest among the others but the difference from the best one is only a marginal 1% range.

Model 0 wins the competition while the rest seem to cluster below it. The order of performance is as follows: Models 0 > Models 2, 5 > Model 1 > Models 3, 4. It suggests that complex models do not always increase the coverage of long-tail dependencies. In this case, simpler models seem to outperform the more complex ones. The order of performance also shows that the generative dependency probability (shared by Models 3 and 4) is a source of error.

With respect to the performance of our models on German, we classify the tendency into three groups: (1) Model 0 that follows the hypothetical trend (2) Model 1, Model 2, and Model 5 whose accuracy rises and suddenly plunges when the full set of the language parameters are used (3) Model 3 and Model 4 whose accuracy rises, plunges, and rises again. On the other hand, their TEDEVAL scores marginally differ from each other as illustrated in Figure 6.8(b).

The accuracy improvement of Model 0 is studied in Figure 6.8(c). Most frequent dependency types are partially captured such as ART > NN, APPR < NN, and VVFIN < NN. When increasing the number of language parameters used in the syntactic prototype, some frequent dependency types become more captured such as NE < NE and NN > VAFIN, while only a few are less captured such as VMFIN < VVINF.

In Figure 6.8(d), long-tail dependency types are captured in lengths 15 and 20. Previously captured dependency types are further captured and a few new dependency types are captured from longer lengths. The neglected dependency types are always

Table 6.7: Top-10 errors in English

(a) Over-generation

Type on EN10	Freq	Type on EN15	Freq	Type on EN20	Freq
VBZ < IN	455	IN < IN	1602	IN < IN	4386
VBD < IN	349	VBZ < IN	990	VBZ < IN	1854
VBP < IN	237	RB > VB	753	NNP > NN	1828
MD < IN	202	VBD < IN	693	VBD < IN	1646
NNP < NN	153	NNP > NN	662	VBN < IN	1557
NNP > CD	147	VBN < IN	586	NN > NN	1330
NNP > NNP	143	NN > NN	583	NN > NNP	1253
NN > NN	142	NN > NNP	575	RB > VB	1141
RB > DT	137	NNP > CD	473	VBG < IN	1072
VBD < RB	106	VBP < IN	430	CD > NN	1037

(b) Under-generation

Type on EN10	Freq	Type on EN15	Freq	Type on EN20	Freq
NN < IN	418	NN < IN	2211	NN < IN	5578
NNP < NNP	415	NNP > NNP	1042	NNS < IN	2368
VBN < IN	306	NNP < NNP	983	VB < IN	2360
NNP > NNP	261	DT > NN	881	NNP < NNP	2091
DT > NN	173	NNS < IN	869	NNP > NNP	1790
VB < IN	168	IN < NN	849	IN < NN	1720
RB > JJ	158	VB < IN	681	DT > NN	1672
NNP < CD	148	JJ > NN	681	JJ > NN	1466
NNS < IN	143	VBD < RB	578	IN < NNS	1242
IN < NN	121	IN < NNS	578	VBD < IN	1211

not captured due to the discrepancy of annotation schemes.

We finally investigate the over- and under-generation of Model 3 in German. We list the top-10 over- and under-generation of dependency types in German of all lengths in Table 6.8. There are, in general, three categories of problems: NP dependency annotation, PP attachment, and PP dependency annotation. On length 10, errors from NP dependency annotation are predominant as seen in ADJA < NN v.s. ADJA > NN and NE < NN v.s. NE > NN. Errors from PP dependency annotation take place in APPR > NN v.s. APPR < NN and NN > APPR v.s. NN < APPR, structurally hiding the underlying PP attachment problem. On lengths 15 and 20, these errors are found throughout the problem space, but PP attachment becomes slightly more obvious than the others. This suggests to us that the annotation scheme for NP should be corrected to improve the directed dependency accuracy.

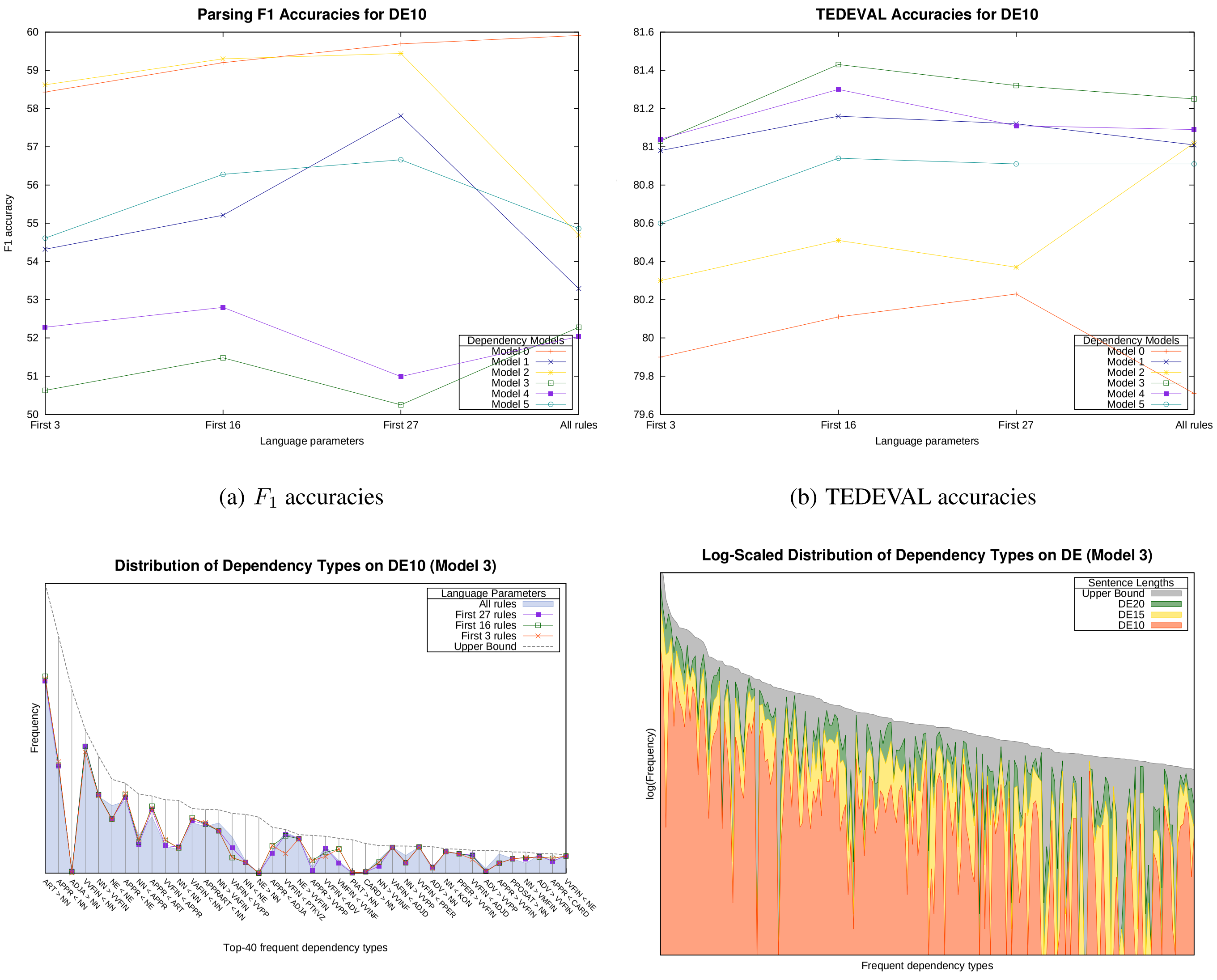
6.9 Japanese

As a free word-order language with a complex verb inflection system and the use of case markers, Japanese is challenging for the grammar induction task. Figure 6.9(a) compares the F_1 accuracies of our models trained and evaluated on JA10. Among those, Model 2 yields the best directed dependency accuracy which at first rises and marginally decays as the number of language parameters increases. Its TEDEVAL accuracy behaves in the similar manner although it is not the best.

Model 2 wins the competition while the rest seem to cluster below it. The order of performance is as follows: Model 2 > Model 1 > Model 0 > Models 3, 4 > Model 5. It suggests that complex models do not always increase the coverage of long-tail dependencies. In this case, simpler models seem to outperform the more complex ones. The order of performance also shows that the lexicon and headword generation probabilities (shared by Models 3, 4, and 5) are sources of error.

When taking into account both Figures 6.9(a) and 6.9(b), Model 0, Model 1, and Model 2 approach the hypothetical trend where their F_1 and TEDEVAL rise and start to saturate as more language parameters are used. In contrast, F_1 accuracies of Model 3, Model 4, and Model 5 plunge after the introduction of the first 16 parameters. Model 4 and Model 5's TEDEVAL scores also decrease but Model 3's TEDEVAL alternatively follows the hypothetical trend.

In Figure 6.9(c), our language parameters, along with Model 2, can partially or even fully capture dependency types in JA10 even though only the first three parameters are



(a) F_1 accuracies

(b) TEDEVAL accuracies

(c) Improvement of dependency type coverage

(d) Coverage of dependency types w.r.t. sentence lengths

Figure 6.8: Parsing accuracies of German

Table 6.8: Top-10 errors in German

(a) Over-generation

Type on DE10	Freq	Type on DE15	Freq	Type on DE20	Freq
ADJA < NN	2047	NN > APPR	2390	NN > APPR	3636
ART > ADJA	1185	CARD < NN	1877	CARD < NN	3291
NE < NN	817	NN > VVPP	1857	NN < ART	3262
VVFIN < ADJA	653	NE < NN	1708	PIAT < NN	2837
NE < ADJA	634	ART > ADJA	1694	ART < APPR	2817
NE < NE	582	ART < APPR	1602	PPER < NN	2723
VVFIN < NN	559	PIAT < NN	1457	PRF < NN	2608
APPR > NN	554	NN < ART	1422	NN > VVPP	2483
VAFIN < ADV	541	NN < APPR	1264	ART > ADJA	2447
NN > APPR	506	NE < ADJA	1176	NN < APPR	2399

(b) Under-generation

Type on DE10	Freq	Type on DE15	Freq	Type on DE20	Freq
ADJA > NN	3409	APPR < NN	7481	APPR < NN	15535
APPR < NN	2529	ADJA < NN	4901	ADJA > NN	7355
ART > NN	1833	ART > NN	4355	ART > NN	7295
NE > NN	1047	NN < NN	2615	NN < NN	4474
NN < NN	924	APPR < ART	2207	APPR > VVPP	4120
NN < NE	875	APPR > VVPP	2088	APPR < ADJA	4106
VVFIN < APPR	868	APPR < ADJA	2026	VVFIN < APPR	3839
NN < APPR	778	VVFIN < APPR	1970	NN < APPR	2777
NN > VVFIN	757	NN < APPR	1811	APPR < APPR	2588
PIAT > NN	621	NN < NE	1541	NE < NE	2587

prescribed. However, some dependency types are not captured at all as we increase the number of language parameters used, such as PVfin > PSSb, Vte > VAUXfin, PQ > Vfin, and ADJiku > VSfin. Small improvement can however be observed as the capture of CDtime > CDtime increases when more parameters are given.

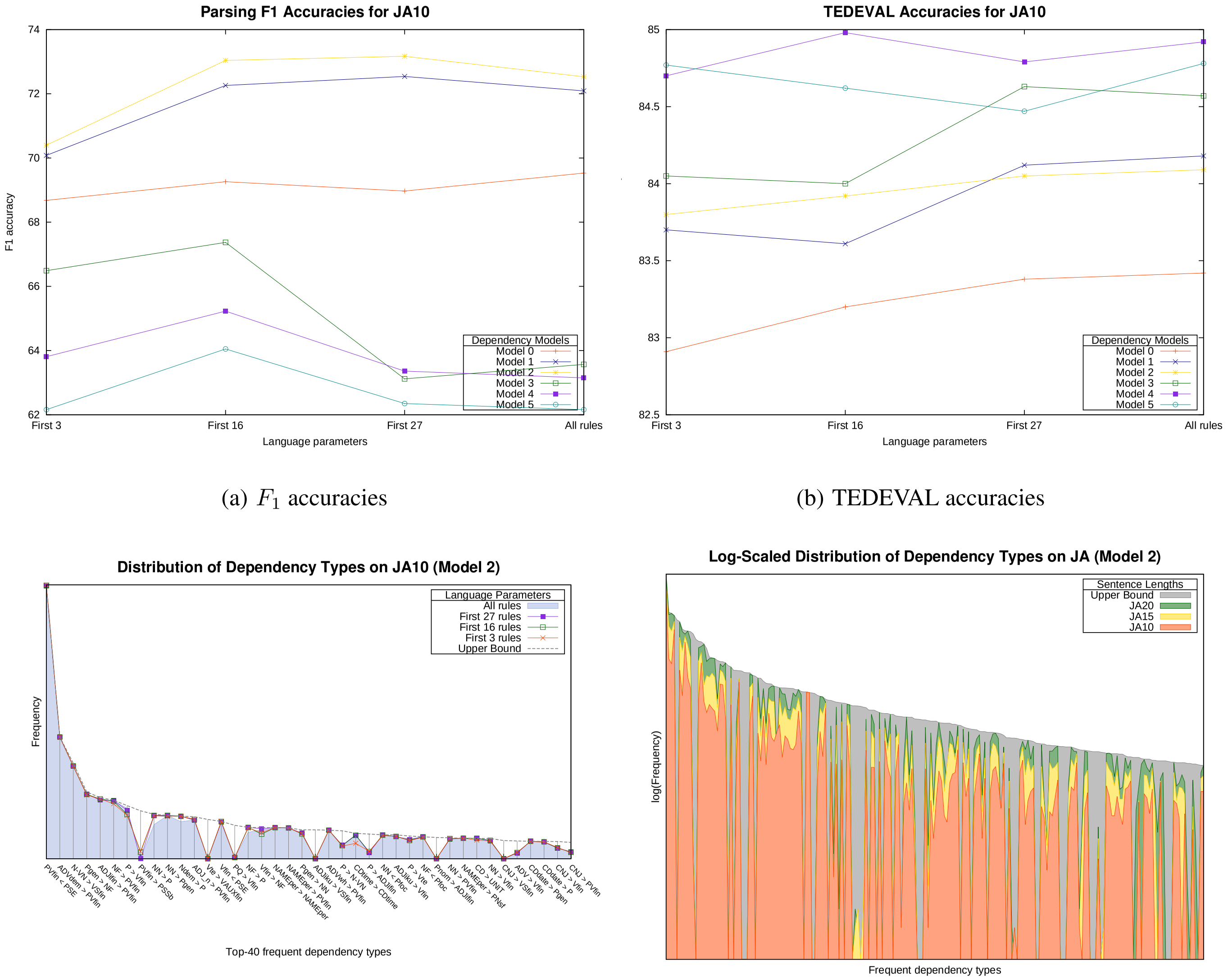
Long-tail dependencies are also coped with by Model 2 as shown in Figure 6.9(d). Though the majority of captured dependency types are from the length 10, we also capture additional less frequent dependency types from lengths 15 and 20. The dependency types neglected in length 10 are still found not captured in longer lengths.

We finally investigate the over- and under-generation of Model 2 in Japanese. We list the top-10 over- and under-generation of dependency types in Japanese of all lengths in Table 6.9. There are, in general, three categories of problems: VP dependency annotation, PP attachment, and coordinate structure dependency annotation. On length 10, errors from VP structure annotation are predominant as seen in PVfin < PSSb v.s. PVfin > PSSb, Vte < VAUXfin v.s. Vte > VAUXfin, and Vfin < PSSa v.s. Vfin > PSSa. We found errors from PP attachment as in P > PVfin, NF < Pnom, and NN < Pnom. Errors from coordinate structure annotation are also found as CNJ > N-VN and CNJ > PSfin. On lengths 15 and 20, these errors are found throughout the problem space, but errors from VP dependency annotation predominant compared to the others. This suggests to us that the annotation scheme for VP should be corrected to improve the directed dependency accuracy.

6.10 Portuguese

We conduct across-the-board experiments for evaluating our system on Portuguese, a rigid word-order language with a verb inflection system. Figures 6.10(a) and 6.10(b) shows that Model 0 outperforms the other models in terms of directed dependency accuracy and TEDEVAL score. In both metrics, Model 0 follows the hypothetical trend in which the accuracy rapidly increases and starts to saturate or decay as the number of language parameters grows. We find that all models seem to follow this hypothetical trend. In particular, Model 1's TEDEVAL decays and saturates after incorporating the first 27 parameters.

Model 0 wins the competition while the rest seem to cluster below it. The order of performance is as follows: Model 0 > Model 2 > Model 1 > Model 5 > Model 3 > Model 4. It suggests that complex models do not always increase the coverage of long-tail dependencies. In this case, simpler models seem to outperform the more complex



(a) F_1 accuracies

(b) TEDEVAL accuracies

(c) Improvement of dependency type coverage

(d) Coverage of dependency types w.r.t. sentence lengths

Figure 6.9: Parsing accuracies of Japanese

Table 6.9: Top-10 errors in Japanese

(a) Over-generation

Type on JA10	Freq	Type on JA15	Freq	Type on JA20	Freq
PVfin < PSSb	506	PVfin < PSSb	1031	PVfin < PSSb	1506
Vte < VAUXfin	431	Vte < VAUXfin	793	Vte < VAUXfin	1071
P > PVfin	330	P > PVfin	440	NN < Pnom	604
ADJiku > N-VN	318	CNJ > N-VN	400	Vfin < PSSa	594
CNJ > N-VN	294	Vfin < PSSa	396	NF < Pnom	538
Vfin < PSSa	182	NN < Pnom	385	P > PVfin	523
NF < Pnom	174	NF < Pnom	363	P > Vfin	481
NN < Pnom	153	ADJiku > N-VN	330	CNJ > N-VN	441
CNJ > PVfin	151	Pgen > NF	321	NN > Vfin	422
VSte < VAUXfin	141	NN > Vfin	281	PVfin < PSSa	378

(b) Under-generation

Type on JA10	Freq	Type on JA15	Freq	Type on JA20	Freq
PVfin > PSSb	524	PVfin > PSSb	1110	PVfin > PSSb	1649
Vte > VAUXfin	430	PQ > Vfin	795	PQ > Vfin	1205
PQ > Vfin	357	Vte > VAUXfin	792	Vte > VAUXfin	1079
ADJiku > VSfin	316	NN > Pnom	387	NN > Pnom	606
Pnom > ADJifin	232	Vfin > PSSa	376	Vfin > PSSa	548
P > ADJifin	200	Pnom > ADJifin	368	NF > Pnom	540
CNJ > VSfin	199	NF > Pnom	366	PVfin > PSSa	522
Vfin > PSSa	176	PVfin > PSSa	352	Pnom > Vfin	519
NF > Pnom	175	ADJiku > VSfin	327	Pnom > ADJifin	482
PVfin > PSSa	160	Pnom > Vfin	325	P > N-VN	387

ones. The order of performance also shows that the lexicon and headword generation probabilities (shared by Models 3 and 4) are sources of error.

We investigate the performance improvement of Model 0 by plotting the Zipfian distribution of dependency types as illustrated in Figure 6.10(c). It can be seen that the first three parameters are able to partially or fully capture frequent dependency types such as art > n, prp < n, n < prp, and v-fin < n. The capture is further improved by putting more language parameters such as n < prp, v-fin < v-inf, and v-fin < v-fin. The dependency types v-inf < prp and v-fin > v-fin are particularly not captured in the full set of parameters despite being previously captured by the first 16 parameters.

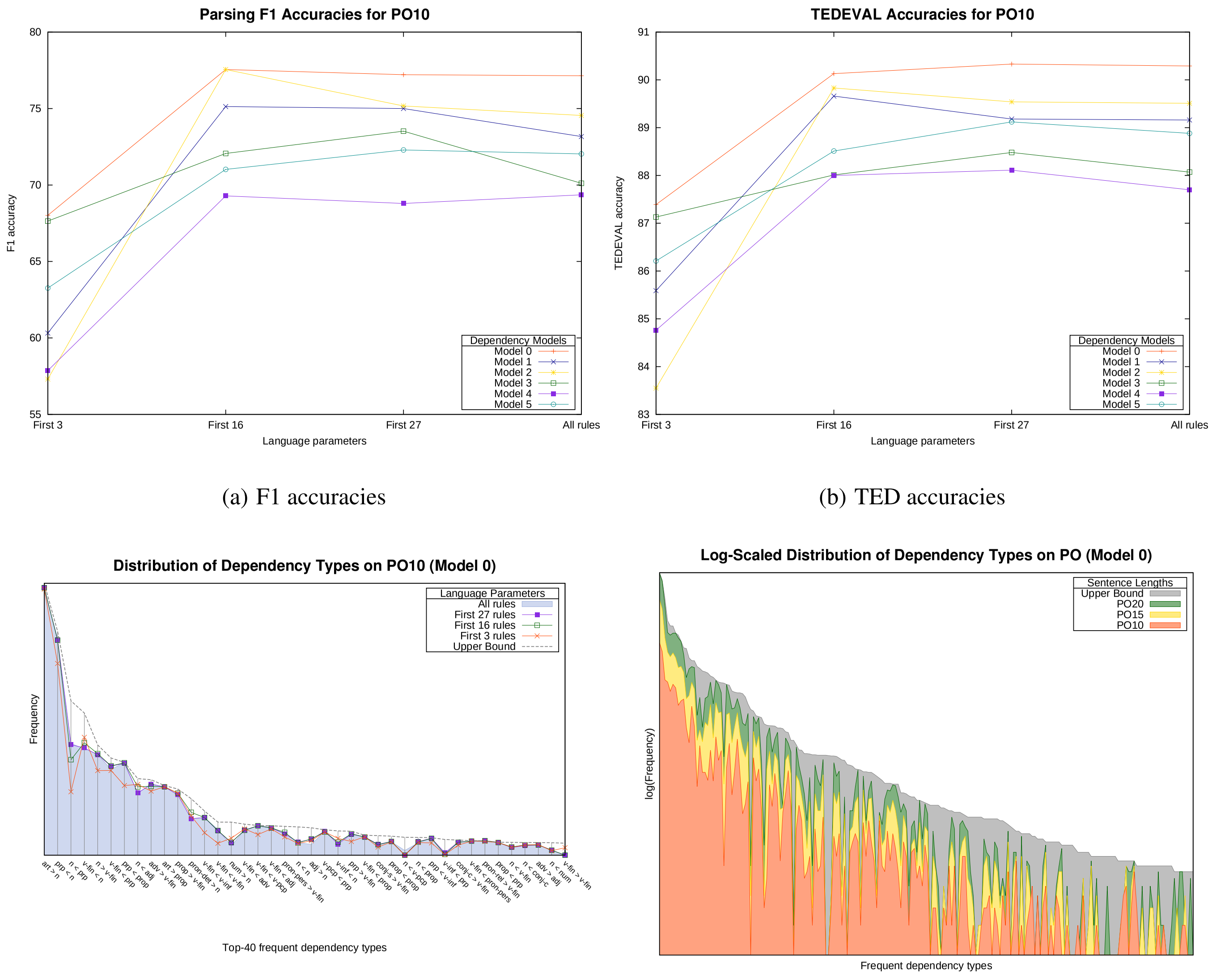
Figure 6.10(d) shows that long-tail dependency types can be coped with by our language parameters. Additional dependency types are captured from lengths 15 and 20, while the neglected dependency types are still not captured in the longer lengths.

We finally investigate the over- and under-generation of Model 0 in Portuguese. We list the top-10 over- and under-generation of dependency types in Portuguese of all lengths in Table 6.10. There are, in general, three categories of problems: PP attachment, NP dependency annotation, and VP dependency annotation. On length 10, PP attachment is predominant such as v-fin < prp, n < prp, and v-inf < prp. We found that NP dependency annotation is also an issue such as num < n v.s. num > n, adj < n v.s. adj > n, n < n, and n < adj. On lengths 15 and 20, these errors are found throughout the observation. Errors from NP dependency annotation also account for the finding of pron-det < n v.s. pron-det > n in lengths 15 and 20. This suggests that the annotation scheme for VP should be corrected to improve the directed dependency accuracy.

6.11 Slovene

Akin to Czech, Slovene is an inflectional language with a rather free word order and is thus challenging for the grammar induction task. We evaluate our models on Slovene and plot their accuracies in Figures 6.11(a) and 6.11(b). Model 1 is shown to outperform the others in terms of directed dependency accuracy. The trend of Model 1 is different from the hypothetical one in that it marginally decays while Model 1's TEDEVAL score marginally improves, as we introduce more language parameters into the syntactic prototype.

Model 1 wins the competition while the rest seem to cluster below it. The order of performance is as follows: Model 1 > Model 2 > Model 3 > Model 5 > Model 4. It suggests that complex models do not always increase the coverage of long-tail



(a) F1 accuracies

(b) TED accuracies

(c) Improvement of dependency type coverage

(d) Coverage of dependency types w.r.t. sentence lengths

Figure 6.10: Parsing accuracies of Portuguese

Table 6.10: Top-10 errors in Portuguese

(a) Over-generation

Type on PO10	Freq	Type on PO15	Freq	Type on PO20	Freq
v-fin < prp	245	v-fin < prp	690	v-fin < prp	920
num < n	142	pron-det < n	397	pron-det < n	729
v-fin < adj	141	num < n	360	num < n	689
pron-det < n	102	v-fin < adj	263	adj < prp	493
v-fin < num	95	v-fin < n	212	adj < n	370
v-fin < n	94	v-fin < num	211	prp > v-fin	349
adj < n	94	adj < n	208	v-fin < num	344
v-fin < pron-det	93	v-fin < v-fin	183	n < prp	341
adv > v-fin	86	adv > v-fin	179	v-fin < n	338
n < prp	77	adv > v-fin	177	conj-c < n	329

(b) Under-generation

Type on PO10	Freq	Type on PO15	Freq	Type on PO20	Freq
n < prp	234	n < prp	511	n < prp	661
v-fin < n	197	v-fin < n	379	pron-det > n	628
num > n	131	pron-det > n	362	v-fin < n	591
pron-det > n	109	num > n	326	num > n	570
adj > n	92	n < n	262	n < n	553
n < n	85	v-inf < prp	231	v-inf < prp	467
n < adj	81	v-fin < prp	206	v-fin < prp	420
v-inf < prp	79	v-pcp < prp	199	v-pcp < prp	396
n < v-pcp	72	adj > n	180	v-inf < n	342
v-fin > v-fin	66	v-inf < n	172	n > v-fin	333

dependencies. In this case, simpler models seem to outperform the more complex ones. The order of performance also shows that the lexicon and headword generation probabilities (shared by Models 3, 4, and 5) are sources of error.

There are three groups of accuracy trend: (1) Model 0, Model 1, Model 2, and Model 5's F_1 and TEDEVAL saturate or slightly decay. (2) Model 3's F_1 accuracy gradually increases and saturates and its TEDEVAL rises, drops, and saturates. (3) Model 4's F_1 accuracy is quite stable but its TEDEVAL scores rises and saturates. This suggests us that simpler Model 0, Model 1, and Model 2 are less sensitive to data sparsity in SL10.

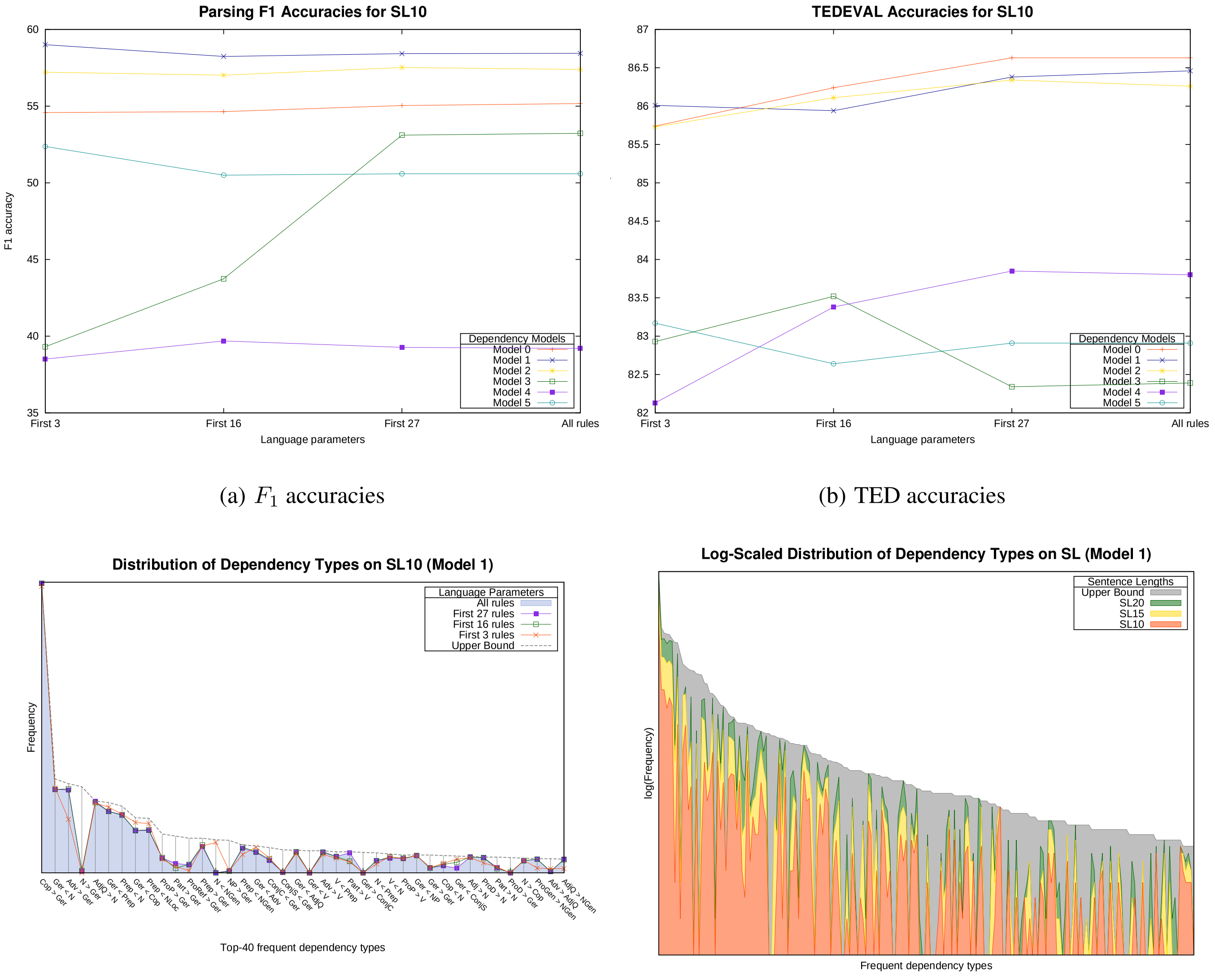
Performance improvement of Model 1 is shown in Figure 6.11(c). The first three language parameters partially capture frequent dependency types such as Cop > Ger, Ger < N, Adv > Ger, and AdjQ > N. As we introduce more language parameters into the syntactic prototype, more dependency types are captured while a few dependency types become less captured such as N < NGen and Ger < ConjS.

From Figure 6.11(d), we found that long-tail dependencies are still captured on longer lengths 15 and 20. Previously captured dependency types are still captured while additional dependency types are captured. Some neglected dependency types are always not captured. We hypothesize that the discrepancy of annotation schemes accounts for this issue.

We finally investigate the over- and under-generation of Model 1 in Slovene. We list the top-10 over- and under-generation of dependency types in Slovene of all lengths in Table 6.11. There are, in general, three categories of problems: gerundial dependency annotation, NP dependency annotation, and PP attachment. On length 10, errors from gerundial dependency annotation are predominant such as N > Cop and Ger > Cop v.s. N > Ger and Cop > Ger. Errors from NP dependency annotation are also found e.g. N > NGen v.s. N < NGen, while PP attachment becomes minor in Slovene. On lengths 15 and 20, these errors are found throughout the observation. This suggests that the gerundial structure annotation should be corrected to improve the directed dependency accuracy.

6.12 Spanish

We evaluate the performance of our models in parsing Spanish, a language with verb inflection and a rigid word order. In Figures 6.12(a) and 6.12(b), Model 0 yields the best directed dependency accuracy and TEDEVAL scores. All models follow the hy-



(a) F_1 accuracies

(b) TED accuracies

(c) Improvement of dependency type coverage

(d) Coverage of dependency types w.r.t. sentence lengths

Figure 6.11: Parsing accuracies of Slovene

Table 6.11: Top-10 errors in Slovene

(a) Over-generation

Type on SL10	Freq	Type on SL15	Freq	Type on SL20	Freq
N > Cop	139	N > Cop	218	Ger > Cop	302
Ger > Cop	69	N > NGen	162	N > Cop	298
ProRef > Cop	67	Ger > V	160	N > NGen	250
N > NGen	56	Ger > Cop	148	Ger > V	231
Ger > V	56	ProRef > Cop	136	ConjS > Cop	224
ProP > Cop	51	Ger > Ger	113	ProRef > Cop	211
ConjS > Cop	48	ProP > Cop	96	Ger < Prep	191
NP > Cop	47	Ger < Prep	95	Ger < ConjC	177
Part > Cop	45	ConjS > Cop	92	ProP > Cop	135
NP > Cop	45	NP > Cop	89	Ger < NGen	124

(b) Under-generation

Type on SL10	Freq	Type on SL15	Freq	Type on SL20	Freq
N > Ger	161	N > Ger	251	N > Ger	338
N < NGen	63	ConjS < Ger	171	ConjS < Ger	312
NP > Ger	58	N < NGen	149	N < NGen	228
Part > Ger	52	Ger > ConjC	123	ProRef > Ger	226
ProRef > Ger	51	ProRef > Ger	120	Ger > ConjC	213
ProP > Ger	46	ProP > Ger	101	ProP > Ger	143
ConjS < Ger	43	Ger < V	91	N < Prep	140
Ger < V	42	NP > Ger	90	Ger < V	138
Ger > ConjC	39	Part > Ger	86	Part > Ger	132
ProD > Ger	29	ConjC < Ger	83	Prep > Ger	128

pothetical trend in which the accuracy rapidly rises and starts to saturate or slightly decay as we incorporate more language parameters in the syntactic prototype.

Model 0 wins the competition while the rest seem to cluster below it. The order of performance is as follows: Models 0 > Model 2 > Model 1 > Models 3, 4 > Model 5. It suggests that complex models do not always increase the coverage of long-tail dependencies. In this case, simpler models seem to outperform the more complex ones. The order of performance also shows that the lexicon and headword generation probabilities (shared by Models 3, 4, and 5) are sources of error.

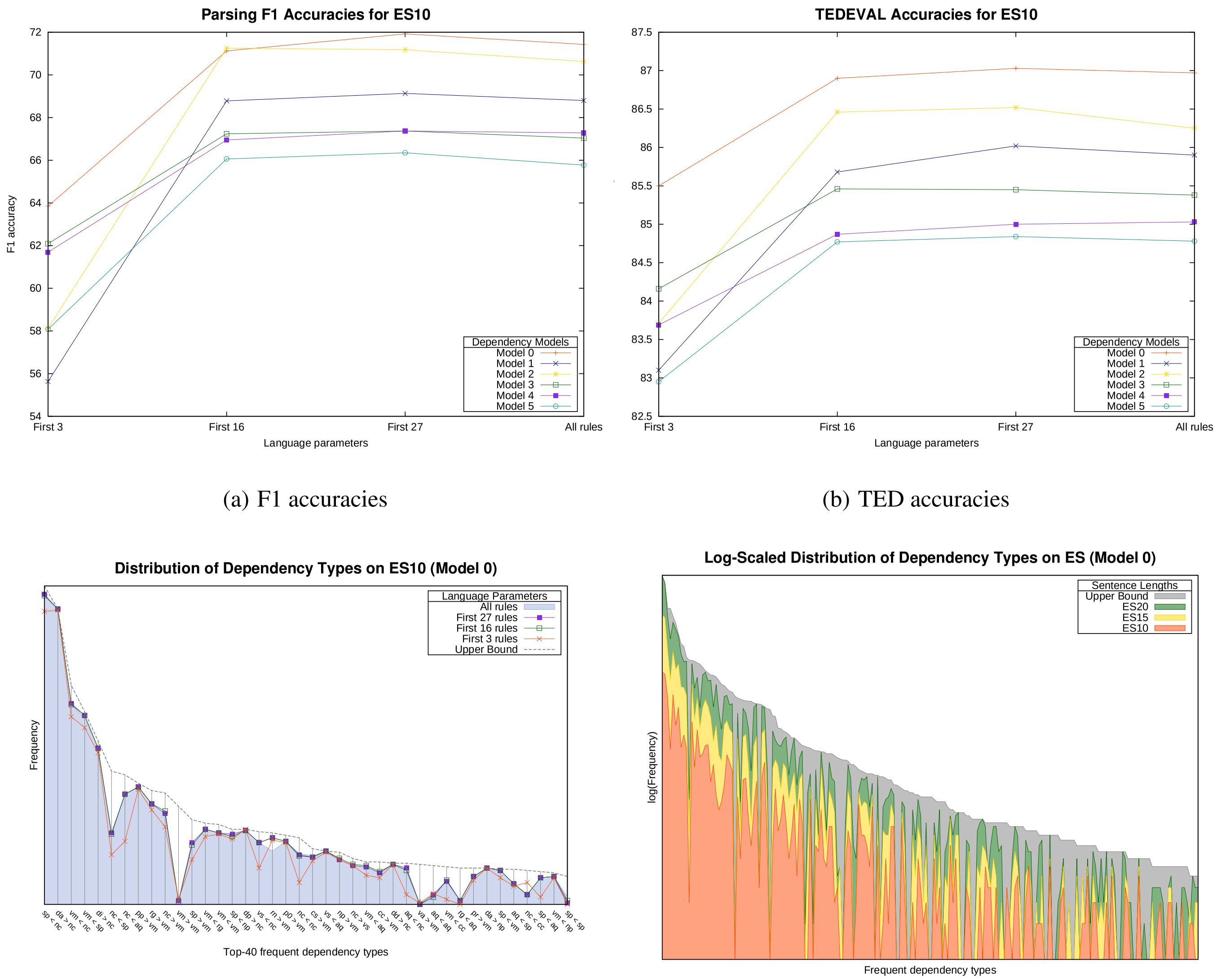
The performance improvement from our language parameters are obvious in Figure 6.12(c). Frequent dependency types are partially or even fully captured by only the first three language parameters, i.e. sp < nc, da > nc, vm < nc, and vm < sp. As more parameters are used, frequent dependency types are more captured such as sp < nc, nc < sp, and nc < aq. Nevertheless a few dependency types such as vm > vm, va > vm, rg < aq, and sp < sp are never captured by the full set of language parameters.

Our language parameters are capable of capturing long-tail dependencies. In Figure 6.12(d), long-tail dependency types can be found in longer lengths 15 and 20. Previously captured dependency types are still captured while additional dependency types are captured. Some neglected dependency types are always not captured.

We finally investigate the over- and under-generation of Model 0 in Spanish. We list the top-10 over- and under-generation of dependency types in Spanish of all lengths in Table 6.12. There are, in general, three categories of problems: serial verb dependency annotation, PP attachment, and adverb/auxiliary dependency annotation. On length 10, errors from serial verb annotation are more frequent than the others, i.e. vm < vm v.s. vm > vm. PP attachment can also be observed such as vm < sp and nc < sp, while errors from adverb/auxiliary dependency annotation are obvious: va < vm v.s. va > vm, and rg > aq v.s. rg < aq. On lengths 15 and 20, the serial verb annotation errors are still predominant although the other errors are found throughout the observation. This suggests that the serial verb annotation errors should be corrected to improve the directed dependency accuracy.

6.13 Swedish

We conduct across-the-board experiments to evaluate the performance of our technique on Swedish, a rather rigid word-order language. From Figures 6.13(a) and 6.13(b), Model 2 yields the best F_1 and TEDEVAL accuracies when the full set of language



(a) F1 accuracies

(b) TED accuracies

(c) Improvement of dependency type coverage

(d) Coverage of dependency types w.r.t. sentence lengths

Figure 6.12: Parsing accuracies of Spanish

Table 6.12: Top-10 errors in Spanish

(a) Over-generation

Type on ES10	Freq	Type on ES15	Freq	Type on ES20	Freq
vm < vm	62	vm < vm	161	vm < vm	239
vm < sp	57	vm < sp	134	vm < sp	212
va < vm	39	sp < vm	120	va < vm	200
cc > nc	35	va < vm	103	sp < vm	193
vs < aq	29	vm < cs	77	cc > vm	142
sp < vm	28	cc > nc	75	va < sp	140
vm < nc	23	nc > vm	72	vm < cs	133
rg > aq	19	va < sp	70	cc > nc	123
nc > vm	19	vs < aq	67	vs < aq	111
vm < cs	18	rg > aq	52	nc > vm	102

(b) Under-generation

Type on ES10	Freq	Type on ES15	Freq	Type on ES20	Freq
vm > vm	79	vm > vm	202	nc < sp	388
nc < sp	53	nc < sp	168	vm > vm	300
va > vm	33	sp > vm	91	sp > vm	169
rg < aq	27	va > vm	82	va > vm	160
aq < aq	24	rg < aq	77	vm < sp	135
sp > vm	21	vm < sp	66	rg < aq	134
nc < cc	20	vm > sp	65	sp < sp	131
sp < sp	17	cs > vm	64	cs > vm	128
nc > vm	17	nc < cc	62	vm > sp	124
vm > sp	16	sp < sp	59	vm < cc	117

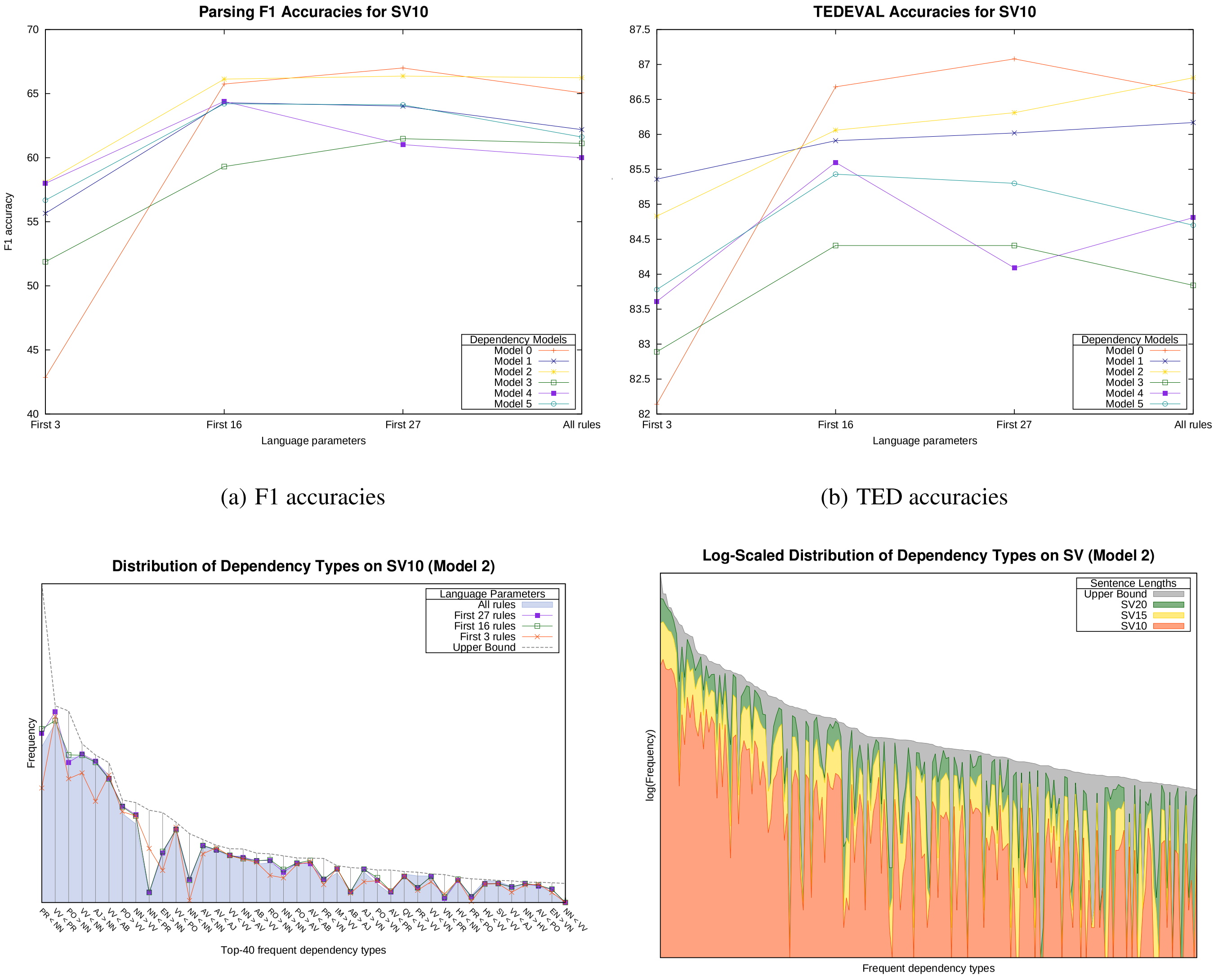
parameters are used. The majority of our models follow the hypothetical trend where the accuracy rapidly rises and starts to saturate when more languege parameters are introduced to the syntactic prototype. Model 4, in contrast, manifests an other trend in which the accuracy slightly drops after using the first 27 parameters. Although their F_1 scores are moderate, these TEDEVAL scores are quite impressive, suggesting us that our parsed trees are closely related to the gold standard ones.

Model 2 wins the competition while the rest seem to cluster below it. The order of performance is as follows: Models 0, 2 > Models 1, 3, 5 > Model 4. It suggests that complex models do not always increase the coverage of long-tail dependencies. In this case, simpler models seem to outperform the more complex ones. The order of performance also shows that the headword generation probability (shared by Models 4 and 5) is a source of error.

In Figure 6.13(c), Model 2 is shown to improve the parsing performance. Frequent dependency types, such as PR < NN, VV < PR, PO > NN, and VV < NN, are partially or fully captured by our language parameters. When we incorporate more language parameters into the syntactic prototype, rapid improvement becomes obvious as more of these dependency types are captured, while only a few language parameters become less captured such as PR > NN, NN > VV and NN < PR. It is noticeable that the dependency type NN < NN, which is not captured when the first three parameters are used, later becomes captured as we increase the number of parameters.

In Figure 6.13(d), long-tail dependencies are shown to be captured by our language parameters. Previously captured dependency types are to be found further in lengths 15 and 20, while most of new dependency types are captured in length 15. There are, of course, some dependency types that are always not captured by our language parameters, due to the discrepancy of annotation schemes.

We finally investigate the over- and under-generation of Model 2 in Swedish. We list the top-10 over- and under-generation of dependency types in Swedish of all lengths in Table 6.13. There are, in general, three categories of problems: PP dependency annotation, PP attachment, and NP dependency annotation. On length 10, errors from PP dependency annotation are predominant among the others. Instead of assigning the preposition as the head of PP, the system sometimes mistakenly identifies the complement as the head, such as PR > NN v.s. PR < NN, and VV < PR, causing the emergence of the error VV < NN instead of VV < PR, making the PP attachment errors structurally latent in the experiment results. Errors from NP dependency annotation are also found in the observation as seen in NN > NN v.s. NN < NN. On lengths 15 and 20, errors



(a) F1 accuracies

(b) TED accuracies

(c) Improvement of dependency type coverage

(d) Coverage of dependency types w.r.t. sentence lengths

Figure 6.13: Parsing accuracies of Swedish

from serial verb annotation are still predominant although the other errors are found throughout the observation. This suggests that the PP dependency annotation errors should be corrected to improve the directed dependency accuracy.

6.14 Turkish

Turkish is considered challenging for the task of grammar induction due to its rich morphology, free word order, and non-projective dependency. Across-the-board experiments are pursued for better understanding of the language. Figures 6.14(a) and 6.14(b) show that Model 2 outperforms the other models when evaluated its directed dependency accuracy and TEDEVAL.

Model 2 wins the competition while the rest seem to cluster below it. The order of performance is as follows: Models 2 > Model 0 > Model 1 > Model 3 > Models 4, 5. It suggests that complex models do not always increase the coverage of long-tail dependencies. In this case, simpler models seem to outperform the more complex ones. The order of performance also shows that the headword generation probability (shared by Models 4 and 5) is a source of error.

The majority of the models follow the hypothetical trend in which the accuracy rapidly rises and starts to saturate as we introduce more language paramters to the syntactic prototype. Model 4's F_1 and TEDEVAL, however, slightly drop when using the first 27 language parameters then rise again when the full set of language parameters are used.

In Figure 6.14(c), performance improvement in Model 2 is obvious. Frequent dependency types are partially or fully captured, such as Noun > Verb, Verb > Verb, Noun > Noun, and Adj > N by the first three parameters. When we incorporate more language parameters, the capture of these dependency types is improved. Only the dependency type Noun < Conj deteriorates as we increase the number of language parameters. There are some dependency types that are not captured by our parameters such as Verb > Noun-NInf, Noun-NInf > V, and Noun < Adj.

Long-tail dependency can also be captured as shown in Figure 6.14(d). Previously captured dependency types are to be found further in length 15. There are quite a number of dependency types that are always not captured by our language parameters, due to the discrepancy of annotation schemes.

We finally investigate the over- and under-generation of Model 2 in Turkish. We list the top-10 over- and under-generation of dependency types in Turkish of all lengths

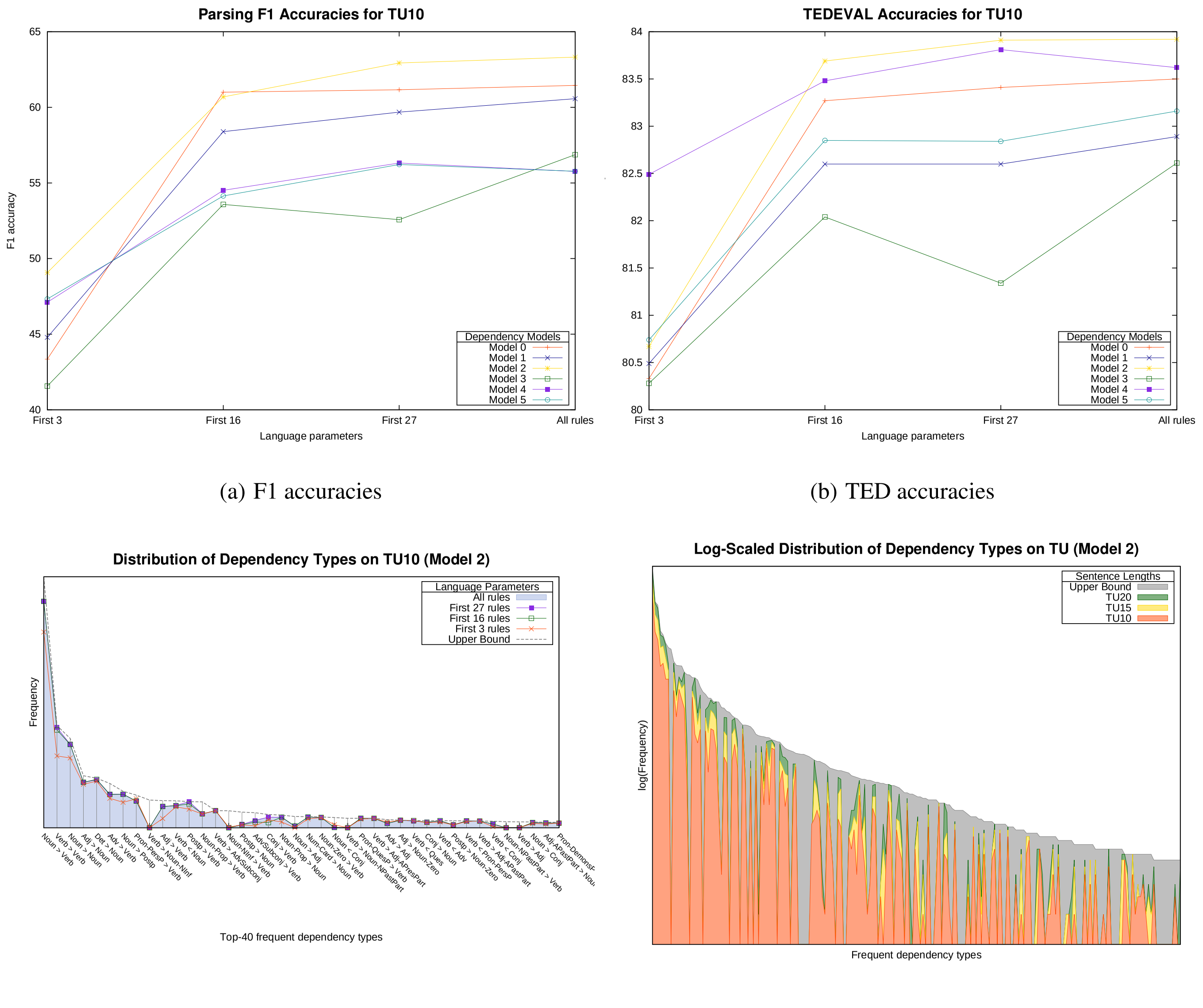
Table 6.13: Top-10 errors in Swedish

(a) Over-generation

Type on SV10	Freq	Type on SV15	Freq	Type on SV20	Freq
VV < NN	452	VV < NN	1159	VV < NN	1613
PR > NN	336	VV < PO	837	VV < PO	1333
VV < PO	335	PR > NN	832	NN > NN	1204
NN > NN	242	NN > NN	753	VV < UK	1097
PR > PO	190	VV < UK	719	PR > NN	1023
VV < PR	165	PR > PO	612	PR > PO	976
VV < UK	162	VV < PR	468	NN > PO	903
TP < NN	146	NN > PO	443	VV < PR	814
VV < EN	130	TP < NN	391	PO > PO	729
VV < AB	111	PO > PO	386	TP < NN	638

(b) Under-generation

Type on SV10	Freq	Type on SV15	Freq	Type on SV20	Freq
PR < NN	895	PR < NN	2261	PR < NN	3245
NN < PR	460	NN < PR	1284	NN < PR	1646
PO > NN	256	NN < NN	807	NN < NN	1183
NN < NN	246	PO > NN	664	PO > NN	1032
EN > NN	224	EN > NN	559	EN > NN	845
VN < PR	137	PR < VN	407	PO > VV	735
AB > AJ	133	NN < VV	391	VN < PR	661
AV < PR	132	VN < PR	384	NN < VV	630
PR < VN	123	VV < PR	377	PR < VN	623
NN > VV	117	PO > VV	349	UK > VV	608



(a) F1 accuracies

(b) TED accuracies

(c) Improvement of dependency type coverage

(d) Coverage of dependency types w.r.t. sentence lengths

Figure 6.14: Parsing accuracies of Turkish

in Table 6.14. There are, in general, three categories of problems: PP attachment, coordinate structure dependency annotation, and gerundial attachment. On length 10, gerundial attachment is the prominent error, such as Verb < Noun-NInf v.s. Verb > Noun-NInf. This error is caused by our permissive free word order, resulting in the SVO order allowed to occur in a SOV language like Turkish. PP attachment is also observable such as Postp > Verb and Postp > Noun. The discrepancy of coordinate structure dependency annotation is found such as Conj < Noun and Noun > Noun v.s. Noun < Conj and Noun > Noun. On lengths 15 and 20, the serial verb annotation errors are still predominant although the other errors are found throughout the observation. This suggests to us that these errors should be corrected to improve the directed dependency accuracy.

6.15 Summary

To answer Research Questions 4 and 5: We have conducted across-the-board experiments to demonstrate the performance of our technique and parsing models and discovered parsing errors by means of over- and under-generation analysis. Summarized in Table 6.15, the errors found are classified into four broad categories: (1) prepositional phrase attachment (2) sentence-like noun phrase (SNP) (3) multiple-word unit (MWU) (4) annotation discrepancy which occurs in several places such as NP, VP, and PP. Among those, PP attachment is the most frequent error that takes place in every language and in any sentence lengths.

From the experimental results, we found that more complex models do not always necessarily outperform the less complex ones. Most languages suffer from the lexicon and headword generation probabilities. Among these Chinese solely suffers from the lexicon generation probability while Turkish solely suffers from the headword generation probability. Model 5 only outperforms in only two languages: Danish and English. On the other hand, languages with non-projective dependency structures; i.e. Czech, Dutch, and German, all suffer from the generative dependency probability.

We will discuss these problems in detail in the next chapter. We regroup the annotation discrepancy problems into simpler and broader subgroups: NP, VP, and coordinate structure, where the number category is combined with the NP and the copula, the auxiliary, and the gerund categories are combined with the VP.

Table 6.14: Top-10 errors in Turkish

(a) Over-generation

Type on TU10	Freq	Type on TU15	Freq	Type on TU20	Freq
Noun > Noun	544	Noun > Noun	716	Noun > Noun	964
Verb > Verb	394	Verb > Verb	577	Verb < Noun-NInf	800
Verb < Noun-NInf	335	Verb < Noun-NInf	563	Conj > Verb	719
Conj > Verb	326	Conj > Verb	560	Verb > Verb	698
Postp > Verb	238	Postp > Verb	409	Postp > Verb	540
Conj > Noun	149	Conj > Noun	260	Conj > Noun	325
Noun > Verb	145	Noun-Prop > Noun	187	Verb < Noun-NPastPart	282
Noun-Prop > Noun	136	Verb < Noun-NPastPart	180	Noun-Prop > Noun	260
Verb < Noun-NPastPart	106	Noun > Verb	179	Noun > Noun-Prop	231
Verb < Adj	103	Noun > Noun-Prop	160	Verb > Conj	205

(b) Under-generation

Type on TU10	Freq	Type on TU15	Freq	Type on TU20	Freq
Verb > Noun-NInf	371	Verb > Noun-NInf	635	Verb > Noun-NInf	903
Noun > Verb	320	Noun > Verb	546	Noun > Verb	723
Noun-NInf > Verb	227	Noun-NInf > Verb	366	Noun-NInf > Verb	497
Postp > Noun	171	Postp > Noun	295	Postp > Noun	401
Noun-Prop > Verb	157	Adv > Verb	240	Verb > Noun-NPastPart	337
Adv > Verb	140	Noun-Prop > Verb	230	Noun-Prop > Verb	312
Noun < Conj	133	Verb > Noun-NPastPart	223	Adv > Verb	298
Noun > Adj	131	Noun > Adj	197	Noun > Adj	243
Verb > Noun-NPastPart	127	Noun < Conj	190	AdvSubconj > Verb	238
AdvSubconj > Verb	98	AdvSubconj > Verb	178	Noun < Conj	230

Table 6.15: Types of frequent errors (over- and under-generation)

Language	**PP Attachment**	**SNP**	**MWU**	**Annotation Discrepancy**							
				NP	**VP**	**PP**	**Coor**	**Copula**	**Aux**	**Num**	**Gerund**
Arabic	X			X			X				
Bulgarian	X			X		X		X			
Chinese	X	X				X			X		
Czech	X			X			X				
Danish	X			X			X				
Dutch	X		X	X		X					
English	X			X						X	
German	X			X		X					
Japanese	X				X		X				
Portuguese	X			X							
Slovene	X			X							X
Spanish	X				X				X		
Swedish	X			X		X					
Turkish	X						X				X

Chapter 7

General Discussion

Outline

In this chapter we discuss three predominant problems that cause errors in the experiments: (1) PP attachment (2) discrepancy of dependency annotation scheme (3) rich morphology. When we incorporate our language parameters, the accuracy of the unsupervised parser starts to approach that of the supervised parser, resulting in PP attachment becoming more obvious. Dependency annotation schemes used in the corpora are thoroughly observed and categorized for ease of understanding. We also explain the criteria of languages whose directed dependency accuracy exceeds 70% tend to comply with. We also discuss how much our approximation of inflection system can improve the accuracy of grammar induction in morphologically rich languages. Finally we discuss the effects of model's expressiveness with respect to the accuracy.

7.1 Introduction

In the last chapter, we showed the accuracy improvement of grammar induction by our language parameters. On corpora of short sentences (up to 10 words), our method significantly outperforms the state-of-the-art techniques on 13 out of 14 languages, while the performance on 1 out of 14 languages is non-significantly different from the state of the art. When applied to corpora of longer sentences (up to 15 and 20 words, respectively) the accuracy gradually decreases within 10% range in all languages, showing that our language parameters scale to learning from long sentences. The language parameters also show a significant effect of frequency, such that the more frequent word order put into the syntactic prototype, the more accuracy we can attain.

However, erroneous dependency types are still generated despite the guidance of syntactic prototypes. For example, in Figure 6.7(d) we plot the Zipfian distribution of English to illustrate the coverage of linguistic constructions produced by the syntactic prototypes. It is noticeable that the more closely the distribution of dependency types in the model approaches that in the gold standard, the more accuracy the system achieves. The drops of coverage in English's Zipfian distribution show that there is still room for improvement in our syntactic prototypes. We analyze over- and under-generation of dependency types and effects of corpus annotation schemes and morphology as follows.

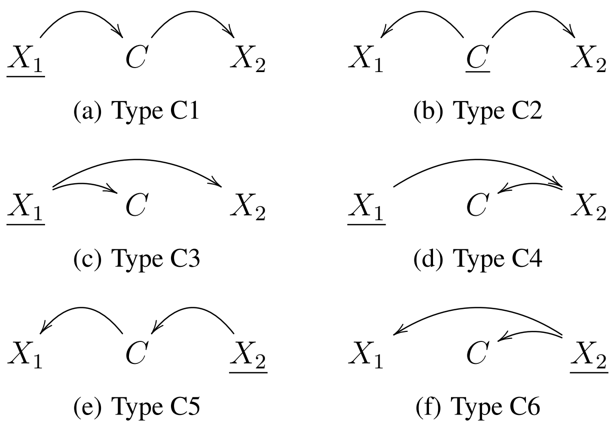
7.2 Modifier Attachment

In Chapter 6, we studied the top-ten erroneous dependency types in each language and we categorized them into over- and under-generation. For example, we list the top-ten erroneous dependency types generated in EN10, EN15, and EN20 in Table 6.7. We notice that frequent errors are caused by ambiguities of PP attachment (e.g. NN < IN and VB < IN). Meanwhile, other minor problems from modifier attachment are also found, such as NP bracketing (e.g. NNP > NNP and NN > NN), VP bracketing (e.g. RB > VB), and syntactically uninformative tags (i.e. tags that do not indicate the attachment direction, such as NNP < NNP vs. NNP > NNP). All of these problems are common in supervised parsing.

7.3 Effects of Dependency Annotation Schemes

We examine the causes of errors in our experiments by comparing the output trees with the gold standard ones. There are several discrepancies between the dependency annotation schemes used in some of the treebanks and those prescribed in our syntactic prototypes. Three types of annotation discrepancies were particularly frequent in the corpora: coordinate structures, NP structures, and VP structures.

As illustrated in Figure 7.1, there are six annotation schemes for coordinate structure. Type C1 assigns the first conjunct (X_1) as the head and the dependency go to the conjunction (C) and then to the second conjunct (X_2). Type C2 simply assigns the conjunction as the head of the coordinate structure. Type C3 assigns the first conjunct as the head which then generates the conjunction and the second conjunct. Type C4 assigns the first conjunct as the head and the dependency goes to the second conjunct



(a) Type C1

(b) Type C2

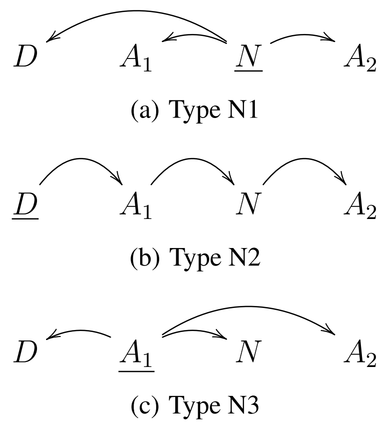
(c) Type C3

(d) Type C4

(e) Type C5

(f) Type C6

Figure 7.1: Discrepant annotation schemes of coordinate structures, where C is a conjunction, X_1 and X_2 are conjunctions, and the heads are underlined.



(a) Type N1

(b) Type N2

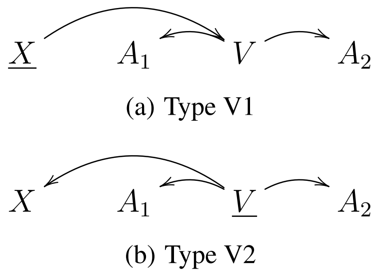
(c) Type N3

Figure 7.2: Discrepant annotation schemes of NP structures, where N is a noun, A_1 and A_2 are nominal modifiers, D is a determiner, and the heads are underlined.

and then to the conjunction. Type C5 assigns the second conjunct as the head and the dependency goes to the conjunction and then to the first conjunct. Finally type C6 assigns the second conjunct as the head which then generates the conjunction and the first conjunct.

There are three annotation schemes for NP structure as portrayed in Figure 7.2. Type N1 assigns the syntactic core noun as the head of the NP. Type N2 assigns the first word of the NP as the head. Finally type N3 assigns the first non-determiner word of the NP as the head.

There are yet two more structure annotation schemes for the VP as shown in Figure 7.3. Type V1 assigns the core verb as the head of the VP regardless of the existence of an auxiliary, while type V2 assigns the auxiliary as the head of the VP if it is present.



(a) Type V1

(b) Type V2

Figure 7.3: Discrepant annotation schemes of VP structures, where V is a verb, X is an auxiliary, A_1 and A_2 are adverbs, and the heads are underlined.

By thorough observation, each corpus is annotated with a different scheme as summarized in Table 7.1. With respect to coordinate structure, the majority of the corpora are annotated with coodinate structure type C2 (Figure 7.1(b)) that assigns the conjunction as the head of the coordinate structure. The majority of the corpora use the NP structure type 1 (Figure 7.2(a)) that assigns the true core noun as the head of the NP. The majority of the corpora use the VP structure type 1 (Figure 7.3(a)) that assigns the auxiliary as the head of the VP if it is present.

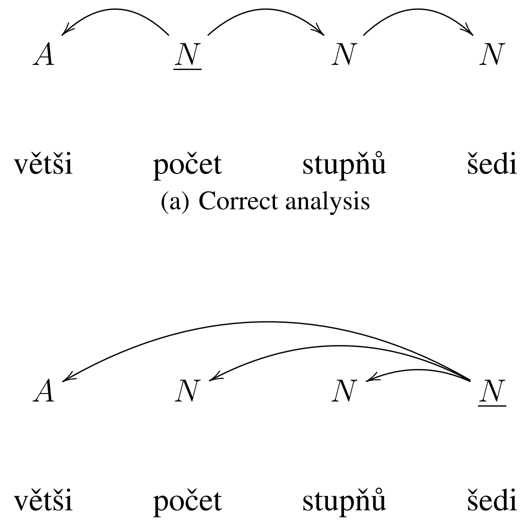
Not surprisingly, the accuracy of grammar induction depends on the dependency annotation schemes. First, there are some annotation schemes that our syntactic prototypes cannot produce (i.e. coordinate structure types 3-6 and NP structure types 2 and 3) because their existence is beyond our initial expectation. Correspondingly, the closer the corpus's annotation scheme is to the parsing model, the more accuracy we can achieve. It should be noted that five languages whose directed dependency accuracy exceeds 70% — Portuguese (C1, N1, V1), English (C1, N1, V1), Japanese (C6, N1, V1), Spanish (C3, N3, V2), and Swedish (C4, N1, V1) — tend to comply with the following creteria:

1. Its annotation schemes for coordinate structure tend to assign the true main conjunct as the head. In Portuguese, English, and Swedish, the first conjunct is the main conjunct, so type C1 is used. In Japanese, the second conjunct is the true head, so type C6 is used. Finally, in Spanish, the type C3 (a slight modification of type C1) still assigns the first conjunct as the head.

2. Its annotation schemes for noun phrase tend to assign the true core noun as the head. In every language except Spanish, type N1 is used. Although Spanish uses type N3, it still correctly assigns the head of the NP because most adjectives follow the core noun in the language.

Table 7.1: Dependency annotation schemes of the corpora

Languages	**Coordinate Structures**	**NP Structures**	**VP Structures**
AR	C2	N2	V2
BU	C2	N1	V2
CH	C2	N1	V2
CZ	C4	N1	V2
DA	C1	N3	V1
DU	C2	N1	V1
EN	C1	N1	V1
DE	C3	N1	V1
JA	C6	N1	V1
PO	C1	N1	V1
SL	C2	N1	V1
ES	C3	N3	V2
SV	C4	N1	V1
TU	C5	N1	V2



(a) Correct analysis

(b) Candidate analysis from the annotation scheme type 1 for NP structures

Figure 7.4: Ambiguous dependency analyses of a Czech NP “větši počet stupňů šedi” without morphological information

3. Its annotation schemes for verb phrase tend to assign the auxiliary as the head if it is present. In every language except Spanish, type V1 is used. In Spanish where type V2 is used, our system produces noticeable errors in this category.

7.4 Effects of Morphology

Morphology plays a crucial role in parsing morphologically rich languages. In Arabic, Bulgarian, Czech, and Slovene, long sequences of nouns-adjective mixture can regularly be found in the text causing combinatory explosion in parsing when non-projective dependencies are taken into account. For instance, dependency analyses of the Czech NP “větši$_{\text{A}}$ počet$_{\underline{\text{N}}}$ stupňů$_{\text{N}}$ šedi$_{\text{N}}$” (greater number of grayscale) are depicted in Figure 7.4. It is hard to generate the correct analysis in Figure 7.4(a) from any annotation schemes for NP structures.

To solve this problem, we minimally approximate the noun inflection system of these languages by roughly distinguish the noun tags with their grammatical cases. As aforementioned in Section 2.5.3, we distinguish Arabic nouns into Nnom, Nga, and Ngen, and Arabic pronouns into Snom, Sga, and Sgen. Since the Bulgarian tagset do not distinguish the grammatical cases, we neutralize the genders of nouns, adjectives, and hybrid tags. Czech nouns are distinguished by attaching the grammatical cases to the noun tag, i.e. N1 to N7, and NX for any cases. Finally, in Slovene, only nouns with genetive and locative cases are distinguished — NLoc and NGen.

Our tag distinction practice has been proven to improve the direct dependency accuracy. As seen in Table 5.1, our Czech accuracy ourperforms Boonkwan and Steedman's (2011) system by 9% because in this previous work, the Czech nouns are attached by all morphological attributes, allowing the data sparsity issue to take place. However, it has been shown in the error analysis that NP structures are still mistakenly produced in a certain degree. In future work, it is neccessary to include morphological information into the tagsets in order to improve the accuracy of grammar induction.

7.5 Effects of Model Expressiveness

From the experimental results, we found that more complex models do not always necessarily outperform the less complex ones. Most languages suffer from the lexicon and headword generation probabilities. Among these, Chinese solely suffers from the lexicon generation probability while Turkish solely suffers from the headword generation probability. The rationale behind the accuracy deterioration of the lexicon and headword generation probabilies is that they make the models more sensitive to data sparsity, especially when incorporated into Models 1 and 2.

Furthermore, languages with non-projective dependency structures; i.e. Czech, Dutch, and German, all suffer from the generative dependency probability. This is because we have to approximate their non-projectivity in the grammars, resulting in more unparsable sentences than the others and ultimately the reduction of accuracy.

7.6 Summary

To answer Research Question 6: We discussed the three problems that cause errors in the experiments: (1) modifier attachment (2) discrepancy of dependency annotation scheme (3) rich morphology. When we incorporate our language parameters, the accuracy of the unsupervised parser starts to approach that of the supervised parser, resulting in modifier attachment becoming predominant. Dependency annotation schemes used in the corpora are classified as follows: six groups for coordinate structure, three groups for NP, and two groups for VP. We found that the languages whose directed dependency accuracy exceeds 70% follow the following criteria: their annotation schemes (1) assign the main conjunct as the head of the coordinate structure (2) assign the true core noun as the head of the NP and (3) assign the auxiliary as the head of the VP if it is present. We also found that our approximation of inflection system in Arabic, Bulgarian, Czech, and Slovene seems to improve the accuracy of grammar induction in morphologically rich languages. Finally, we found that data sparsity still affects the accuracy of grammar induction with our parsing models and non-projectivity approximation in our grammar is also a source of error.

Part IV

Conclusion

Chapter 8

Conclusion

8.1 Concluding Remarks

We have presented an approach to prototype-driven grammar induction using language parameterization. A set of 33 parameters of basic word orders, which are easy to acquire from non-linguist informants, capture frequent grammar rules in the Zipfian distribution of natural languages, while the rest of the grammar can be automatically induced from unlabeled data. An initial grammar generated from such information is represented in terms of a lexicalized categorial grammar.

We have assessed the scalability of syntactic prototype in grammar induction on as many languages and as longest sentences as possible. Although our language parameters need slightly more labor to elicit than Naseem et al.'s (2010) head-dependent pairs do, the language parameters can be elicited by direct consultation with grammar compendiums and by interview with naïve language informants. Our language parameters shrink the search space of the grammar induction problem by exploiting both word order and predefined attachment directions.

The contribution of this thesis is three-fold. (1) We have shown that the language parameters are adequately generalizable cross-linguistically, as our grammar induction experiments were carried out on 14 languages by an unsupervised grammar induction system. (2) Our specification of language parameters improves the accuracy of unsupervised parsing even when the parser is exposed to much less frequent linguistic phenomena in longer sentences while the accuracy of directed dependency recovery decreases within 10% compared to that of the short sentences. (3) We have identified some predominant error-types in grammar induction which provide room for accuracy improvement in future work.

8.2 Performance

The proposed language parameters are capable of capturing most frequent grammar rules in natural languages. Our method outperforms or is comparable to the state-of-the-art techniques.

With only 10 man-hours for preparing syntactic prototypes, it improves the accuracy of directed dependency recovery over the state-of-the-art Gillenwater et al.'s (2010) completely unsupervised parser in: (1) Chinese by 30.32% (2) Swedish by 28.96% (3) Portuguese by 37.64% (4) Dutch by 15.17% (5) German by 14.21% (6) Spanish by 13.53% (7) Japanese by 13.13% (8) English by 12.41% (9) Czech by 9.16% (10) Slovene by 7.24% (11) Turkish by 6.72% and (12) Bulgarian by 5.96%.

We have noted that although the directed dependency accuracies of some languages are below 60%, their TEDEVAL scores are still satisfactory (approximately 80%). This suggests us that our parsed trees are, in fact, closely related to the gold-standard trees and that our dependency recovery scores are unduly depressed by the vagaries of the PASCAL annotation schemes across the subcorpora.

From the across-language average accuracies of each model, we found that the trade-off between the accuracies and the model expressivity due to the data sparsity issue is obvious. We found that Model 0 seems to outperform the others although it is the least expressive.

We have compared our results with PASCAL Challenge on linguistic structure induction. In five out of eight languages (Czech, Danish, Dutch, English, and Portuguese), our method significantly outperforms the others, while in the other languages (Arabic, Slovene, and Swedish) our results are not significantly inferior.

We have also evaluated our method on different sentence lengths — 10, 15, and 20 — to evaluate the scalability to long-tail dependencies of our language parameters. We found that, although the accuracy decreases as the sentence length increases to 15 and 20, the directed dependency accuracy decreases within the range of 10% in Bulgarian, Chinese, Danish, Dutch, Spanish, Portuguese, and Turkish.

Finally, we have studied the effects of language parameters towards the accuracy improvement. We found for almost all languages that the greater number of language parameters used in the syntactic prototype, the better the accuracy.

8.3 Error Analysis

Error analysis showed, not surprisingly, that syntactic ambiguity issues that are common in supervised parsing such as PP attachment also take place in grammar induction. However, the accuracy of grammar induction is more strongly limited by a number of discrepancies in the dependency annotation schemes used in the various gold standards used for evaluation. In particular, some annotation patterns are not generated by our language parameters because their existence is beyond our expectation.

We have conducted across-the-board experiments to demonstrate the performance of our technique and parsing models and discovered parsing errors by means of over- and under-generation analysis. The errors found are classified into four broad categories: (1) modifier attachment (2) quasi-sentential noun phrase (QSNP) (3) multiple-word unit (MWU) (4) annotation discrepancy which occurs in NP, VP, and coordinate structure.

We have discussed the three sources of error in the parser: (1) modifier attachment (2) discrepancies of dependency annotation schemes and (3) rich morphology. When we incorporate our language parameters, the accuracy of the unsupervised parser starts to approach that of the supervised parser, resulting in PP attachment becoming predominant. Dependency annotation schemes used in the corpora are classified as follows: six groups for coordinate structure, three groups for NP, and two groups for VP. We found that the languages whose directed dependency accuracy exceeds 70% follow the following criteria: their annotation schemes (1) assign the main conjunct as the head of the coordinate structure (2) assign the true core noun as the head of the NP and (3) assign the auxiliary as the head of the VP if it is present. We also found that our approximation of inflection system in Arabic, Bulgarian, Czech, and Slovene seems to improve the accuracy of grammar induction in morphologically rich languages. Finally, we found that data sparsity still affects the accuracy of grammar induction with our parsing models and non-projectivity approximation in our grammar is also a source of error.

8.4 Future Work

Our future work in prototype-driven grammar induction will be as follows. First we will examine the effectiveness of language parameter elicitation by comparing between the parameters elicited from humans and those elicited from machine translation. Second, as our accuracy approaches that of supervised parsers, we need to improve attachment accuracy. Third, we need to speed up the process of language parameter elicitation. Fourth, we have to improve the parsing models to cope with data sparsity better and to expand our linguistic prototype's coverage and expressiveness to nonprojective dependencies. Finally, the use of the directed dependency accuracy starts to be questionable because the discrepancies of dependency annotation become significant.

8.4.1 Language Parameter Elicitation by Machine Translation

One interesting question regarding this thesis is the accuracy of the language parameters we elicited from machine translation compared with those attained from human interview. It is interesting to conduct grammar induction for less-privileged languages whose parallel corpora are voluminous and the word orders of the other language, e.g. English, has been analyzed.

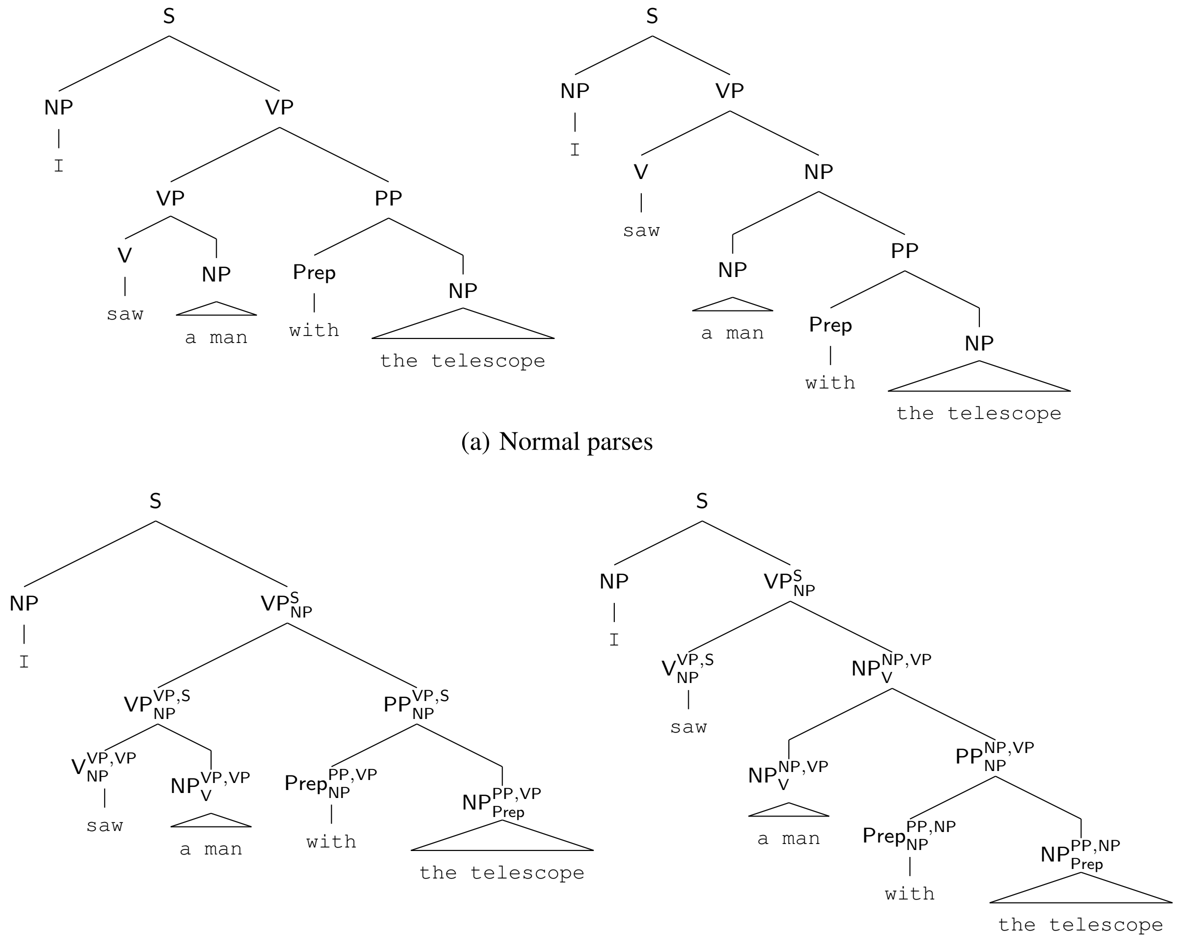
At the first step, we plan to redo the experiments on the languages we already had interview results (i.e. Arabic, Chinese, Japanese, and German) using Google Translate to elicit the language parameters. We will compare both sets of language parameters and analyze their linguistic differences. We will also compare errors produced from both syntactic prototypes to analyze the effects to the system.

Next, we plan to incorporate our language parameters to existing bilingual parsing algorithms such as (Snyder et al., 2009) and induce word orders from the resulting parallel treebanks.

8.4.2 Syntactic Ambiguity

Syntactic ambiguity was shown to be predominant among the produced errors, including (1) PP attachment that occurs in every language and (2) quasi-sentential noun phrase that is frequent in Chinese. We plan to reduce this issue by incorporating more information to each node of the tree, such as Tree Markovization (Klein and Manning, 2003).

In this technique, a limited number of vertically and horizontally ancestor tags are added to the current node's tag, helping distinguish different kinds of phrase attachment with context. m, n-Markovization means that each node is annotated with m horizontal ancestors and n vertical ancestors. For example, two analyses of the sentence "`I`



(a) Normal parses

(b) Corresponding 1,2-Markovized trees

Figure 8.1: Tree Markovization

saw a man with the telescope" are 1,2-Markovized as illustrated in Figure 8.1. The preposition 'with' becomes differentiated into $\mathsf{Prep}_{\mathsf{NP}}^{\mathsf{PP,VP}} \rightarrow \texttt{with}$ and $\mathsf{Prep}_{\mathsf{NP}}^{\mathsf{PP,NP}} \rightarrow \texttt{with}$, which ultimately help disambiguate PP attachment.

Tree Markovization however requires some modification when applied to grammar induction whose search space is very large. Straightforward context annotation to the tags makes structure sharing impossible as all tags are differentiated. One way to overcome this hindrance is to replace the Inside-Outside Algorithm with Bayesian inference based on Markov-Chain Monte Carlo methods such as (Johnson et al., 2007b). In their MCMC method, each tree is sampled from the posterior distribution of trees, whereby for each node, we sample a sequence of daughters from the distribution of their inside scores.

We can approximate Bayesian inference of Markovized PCFGs by employing the Metropolis-Hastings Algorithm. We set the proposal distribution $Q(\cdot|s, \Theta')$ to be the posterior distribution of un-Markovized trees, and the true distribution $P(\cdot|s,\Theta)$ to be the posterior distribution of Markovized trees.

As explained in Algorithm 8.1, for each iteration, we sample a tree from the proposal distribution Q closely following Johnson et al.'s (2007b) joint distribution for PCFGs and Markovize the tree. If the Markovized tree is accepted, i.e. $a \geq 1$, we replace the old trees in the tree sequences with the new trees. The algorithm iterates until the distribution of Markovized trees converges.

On the other hand, it is worth noticing that we need not Markovize every node of the tree as we can focus on specific types of syntactic ambiguity, e.g. PP attachment or quasi-sentential noun phrase. In this case, partial Markovization is feasible as we can annotate context to specific tags, such as prepositions, relative pronouns, and verbs, allowing structure sharing in the charts.

Algorithm 8.1 `mhtrain`$(\mathcal{D}, P, Q)$: Metropolis-Hastings algorithm, where B is the number of burn-in iterations

```
let counter $i \leftarrow 0$
Initialize the un-Markovized tree sequence $\mathbf{t}'$
Initialize the Markovized tree sequence $\mathbf{t}$ with respect to $\mathbf{t}'$
repeat
    let input sentence $s \leftarrow \mathcal{D}[i \mod N+1]$
    Sample an un-Markovized tree $t' \sim Q(\cdot|s,\Theta')$
    let a Markovized tree $t \leftarrow \texttt{markovize}(t', s)$
    let $a \leftarrow \frac{P(t)}{P(\mathbf{t}[i])}\frac{Q(\mathbf{t}'[i];t')}{Q(t';\mathbf{t}'[i])}$    ▷ acceptance ratio
    if $a \geq 1$ or $i < B$ then
        update $\mathbf{t}'[i] \leftarrow t'$
        update $\mathbf{t}[i] \leftarrow t$
    end if
    update $i \leftarrow i+1$
until $P(\mathcal{D})$ converges
return $\mathbf{t}'$
```

8.4.3 Speed-up of Language Parameter Elicitation

Although our language parameters have been shown to capture frequent word orders, there are two remaining issues which have to be solve in the future. The first one is the preparation process for language parameter elicitation is relatively time-consuming.

The other one is we have to seek for additional language parameters which can improve the accuracy in morphologically rich languages.

In the first issue, the preparation process takes up to ten hours — two to four hours for parameter elicitation and four to six more hours for mapping the corpus-specific tagset to the cross-linguistic tagset. As the most time-consuming process, the process of mapping the corpus-specific POS tagset to the cross-linguistic tagset can be abridged by roughly preparing the mapping table according to the treebank annotation manuals. Initially, the lexicon inventory needs not be complete; i.e. the POS tagset do not have to be completely mapped to the cross-linguistic tagset. We plan to extend the lexicon inventory by existing algorithms, such as (Thomforde and Steedman, 2011; Bisk and Hockenmaier, 2012a; Bisk and Hockenmaier, 2012b), but we further constrain the search space with the elicited language parameters. Furthermore, when bitexts are available, bilingual parser induction such as (Snyder et al., 2009) can also be useful in automatic elicitation of language parameters.

In the second issue, we plan to incorporate morphological information to the language parameters. At this stage, we take into account only grammatical cases. In Arabic, Bulgarian, Czech, and Slovene, these grammatical cases can be generally classified into four groups: (1) nominal cases (2) adjectival cases (3) adverbial cases (4) prepositional object cases. If we can map corpus-specific cases into these generalized cases, we can specialize syntactic categories for nouns and adjectives with different cases.

8.4.4 Data Sparsity and Nonprojective Dependency

There are two problems in our parsing models: (1) data sparsity in the lexicon and headword emission probabilities and (2) ineffective non-projectivity approximation.

For the first problem, we suggest use posterior regularization techniques such as (Gillenwater et al., 2010; Ganchev et al., 2010) to regularize the probability distributions of the lexicon and headword emission. In this method, we can put constraints on the posteriors (in this case, the lexicon and headword emission probabilities) to learn them efficiently. We believe that such posterior constraint can be automatically generated from the language parameters already elicited.

To resolve the problems in our non-projectivity approximation, we plan to extend the expressive power of CDG to that of Combinatory Categorial Grammar in order to cope with nonprojective dependency. Statistical CCG parsing (Hockenmaier, 2003b; Hockenmaier, 2003a) shows that nonprojective dependency, such as coordinate structures, dative shifts, and *Wh*-movement in English, and serial verb construction in Dutch, can be effectively handled. By all means, CCG parsing is more complex than CDG parsing because it enlarges the search space by introducing extra derivation rules. Our language parameters can be employed to control the extension of lexicon inventory.

8.4.5 Evaluation

From the error analysis, we discovered that there are discrepancies of dependency annotation schemes, because there is a big gap between the directed dependency accuracy and TEDEVAL scores. We questioned whether the directed dependency accuracy is appropriate for accuracy assessment. In completely unsupervised grammar induction, we attempt to simulate the annotation scheme of a particular corpus, justifying the use of directed dependency accuracy as an evaluation metric. However, this differs from our approach: we attempt to capture frequent word orders with the syntactic prototype following our annotation scheme and extend them with less frequent word orders in the corpus. It is unfair to evaluate prototype-driven grammar induction in the same fashion as completely unsupervised grammar induction.

We suggest two additional evaluation practices for prototype-driven grammar induction. First, tree similarity scores such as TEDEVAL and NED should be reported alongside the directed dependency accuracy, reflecting how similar the parsed dependency structures are to the gold standard ones regardless of annotation schemes. Second, task-based evaluation should also be conducted to demonstrate the usefulness of the produced dependency structures in a particular task, for instance, machine translation and information extraction.

8.5 Online Resources

We established a project named *FUNGI* (Fast Unsupervised Grammar Inducer) which is hosted on SourceForge.net. The source code, the language parameters for all 14 languages in the experiments, and the corpus preparation script can be downloaded from the following URL:

`http://fungi.sourceforge.net/`

The code was written in OCaml (`http://www.ocaml.org`), and compilation with version 4.00 or above is highly recommended. It also requires the library Functory (`http://functory.lri.fr`) for multicore processing.

Part V

Appendices

Appendix A

Syntactic Prototype Questionnaire

Preliminaries

Question I

Which language are you speculating its linguistic typology?

Answer:

(Example: English)

Question II

How much do you know about that language?

□ I am a native speaker of it.

□ I am a (computational) linguist, a typologist, or a syntactician.

□ Other reasons.

Please specify here:

Canonical Word Orders

Question 1: Sentence

What are *canonical* word-ordering patterns of the subject (S), the verb (V), the direct object (O), and the indirect object (I) in your language?

□ Tick here if you consider that your language rather has fixed word orders.

□ Tick here if there exists a notion of ditransitive verb in your language. Also tick the dominant word orders in the following table. (For example, English's word order is SVIO.)

□ SVOI
□ SVIO
□ SIVO
□ ISVO
□ VSOI
□ VSIO
□ VISO
□ IVSO
□ SOVI
□ SOIV
□ SIOV
□ ISOV
□ OVSI
□ OVIS
□ OIVS
□ IOVS
□ VOSI
□ VOIS
□ VIOS
□ IVOS
□ OSVI
□ OSIV
□ OISV
□ IOSV

□ Otherwise, tick here if there doesn't exist a notion of ditransitive verb in your language. Also tick the dominant word orders in the following table.

□ SOV
□ OSV
□ SVO
□ OVS
□ VSO
□ VOS

□ Otherwise, tick here if you consider that your language strictly has free word order. That means *all* the word orders in the above table are allowed.

Question 2: Simple Modifiers

Question 2.1: Adjectives and Nouns

What is the word order of the adjectives when they combine with a noun?

□ Tick here if you consider that your language allows the adjectives to combine with nouns. Also tick the allowable word orders in the following table. (For example, English allows Adj+N.)

□ Adj+N □ N+Adj

□ Otherwise, tick here if you consider that your language does not allow the adjectives to combine with the nouns.

Question 2.2: Adverbs and Verb Phrases

What is the word order of the adverbs when they combine with a verb phrase?

□ Tick here if you consider that your language allows the adverbs to combine with verb phrases. Also tick the allowable word orders in the following table. (For example, English allows both Adv+VP and VP+Adv.)

□ Adv+VP □ VP+Adv

□ Otherwise, tick here if you consider that your language does not allow the adverbs to combine with the verb phrases.

Question 2.3: Adverbs and Adjectives

What is the word order of the adverbs when they combine with an adjective?

□ Tick here if you consider that your language allows the adverbs to combine with adjectives. Also tick the allowable word orders in the following table. (For example, English allows Adv+Adj.)

□ Adv+Adj □ Adj+Adv

□ Otherwise, tick here if you consider that your language does not allow the adverbs to combine with the adjectives.

Question 2.4: Negators

What is the word order of the negators (Neg) when they combine with a verb (V), an adjective (Adj), and an adverb (Adv)?

□ Tick here if there exists a notion of negators in your language. Also tick the allowable word orders in the following table. (For example, English allows Neg+V, Neg+Adj, and Neg+Adv.)

□ Neg+V □ V+Neg
□ Neg+Adj □ Adj+Neg
□ Neg+Adv □ Adv+Neg

□ Otherwise, tick here if there does not exist the notion of negators in your language.

Question 3: Complex Verbs

Question 3.1: Copulae

Does there exist a notion of copulae in your language?

□ Tick here if it does.

□ Otherwise, tick here if it does not.

Question 3.2: Modal Verbs

What is the word order of the modal verbs when they combine with a verb phrase?

□ Tick here if you consider that there is the notion of modal verbs in your language. Also tick the allowable word orders in the following table. (For example, English allows Modal+VP.)

□ Modal+VP   □ VP+Modal

□ Otherwise, tick here if the modal verbs don't exist in your language.

Question 3.3: Intransitive Complex Verbs

What are canonical word orders of the subject (S), the intransitive complex verb (V), and the complementing verb phrase (C)?

□ Tick here if there exists the notion of intransitive complex verbs in your language. Also tick the allowable word orders in the following table. (For example, English allows SVC.)[1]

□ SVC
□ SCV
□ CSV
□ VSC
□ VCS
□ CVS

□ Otherwise, tick here if there doesn't exist the notion of intransitive complex verbs in your language.

Question 3.4: Transitive Complex Verbs

What are canonical word orders of the subject (S), the transitive complex verb (V), the object (O), and the complementing verb phrase (C)?

□ Tick here if there exists the notion of transitive complex verbs in your language. Also tick the allowable word orders in the following table. (For example, English allows SVOC.)[2]

□ SVOC
□ SVCO
□ SCVO
□ CSVO
□ VSOC
□ VSCO
□ VCSO
□ CVSO
□ SOVC
□ SOCV
□ SCOV
□ CSOV
□ OVSC
□ OVCS
□ OCVS
□ COVS
□ VOSC
□ VOCS
□ VCOS
□ CVOS
□ OSVC
□ OSCV
□ OCSV
□ COSV

[1] You can also treat the serial verb construction as this complex verb. For example, Thai allows SVC.

[2] You can also treat the serial verb construction as this complex verb. For example, Thai allows SVOC.

□ Otherwise, tick here if there doesn't exist the notion of transitive complex verbs in your language.

Question 4: Complex Modifiers

Question 4.1: Prepositions/Postpositions

What is the word order of the prepositions/postpositions in your language?

□ Tick here if there exists the notion of prepositions/postpositions in your language. Tick the allowable word orders in the following table. (For example, English allows Prep+NP.)

□ Prep+NP	□ NP+Post
(preposition)	(postposition)

□ Otherwise, tick here if there doesn't exist the notion of prepositions/postpositions in your language.

Question 4.2: Possessivizers

What is the word order of the owner (Owner), the possessivizer (Poss), and the ownee (Ownee) in your language?

□ Tick here if there exists the notion of prepositions/postpositions in your language. Also tick the allowable word orders in the following table. (For example, English allows the pattern Owner+Poss+Ownee.)

□ Owner+Ownee+Poss
□ Owner+Poss+Ownee
□ Poss+Owner+Ownee
□ Ownee+Owner+Poss
□ Ownee+Poss+Owner
□ Poss+Ownee+Owner

□ Otherwise, tick here if there doesn't exist the notion of possessivizers in your language.

Question 4.3: Relative Pronouns

What is the word order of the relative pronoun (Relpro) and the complementing verb phrase (VP), and that of the relative clause (Relcls) and the core noun phrase (NP) in your language?

□ Tick here if there exists the notion of relative pronouns in your language. Tick the allowable word orders in the following table. (For example, English allows Relpro+VP.)

□ Relpro+VP □ VP+Relpro

□ Otherwise, tick here if there doesn't exist the notion of relative pronouns in your language.

Question 4.4: Modifiers

What is the word order of the modifiers (*Mod) in your language?

□ Tick here if there exists the notion of adjectival modifiers in your language. Tick the allowable word orders in the following table.

□ NP+NMod □ NMod+NP

□ Tick here if there exists the notion of adverbial modifiers in your language. Tick the allowable word orders in the following table.

□ VP+VMod □ VMod+VP

□ Tick here if there exists the notion of gerund modifiers in your language. Tick the allowable word orders in the following table.

□ Gerund+GMod □ GMod+Gerund

□ Tick here if there exists the notion of sentential modifiers in your language. Tick the allowable word orders in the following table.

□ Sent+SMod □ SMod+Sent

□ Otherwise, tick here if there doesn't exist the notion of these modifiers in your language.

Question 4.5: Sentential Particles

What is the word order of the sentence (Sent) and the sentential particle (Part) in your language?

□ Tick here if there exists the notion of sentential particles in your language. Tick the allowable word orders in the following table. (For example, English allows Part+Sent.)

□ Sent+Part □ Part+Sent

□ Otherwise, tick here if there doesn't exist the notion of sentential particles in your language.

Question 4.6: Noun Classifiers

Do you use noun classifiers in your language?

□ Tick here if you use noun classifiers to count things in your language. Tick the allowable word orders in the following table. (CL = noun classifiers)

□ Num+CL   □ CL+Num

And what is it used as in your language?

□ Adjective   □ Adverb   □ Noun modifier   □ VP modifier

□ Otherwise, tick here if you don't use noun classifiers in your language.

Question 5: Gerunds

Can a gerund, a transformation of a verb phrase, perform the following functions?

□ A noun phrase.

□ A noun modifier. Also tick the allowable word orders in the following table. (For example, English allows NP+Gerund.)

□ NP+Gerund   □ Gerund+NP

□ A predicative adverbial. Also tick the allowable word orders in the following table. (For example, English allows VP+Gerund.)

□ VP+Gerund   □ Gerund+VP

□ Otherwise, tick here if there doesn't exist the notion of gerunds in your language.

Question 6: Subordinate Conjunctions

What is the word order for the main clause (Main), the subordinate conjunction (Conj), and the subordinate clause (Subcls) in your language?

□ Tick here if there exists the notion of subordinate conjunctions in your language. Also tick the allowable word orders in the following table. (For example, English allows Main+Conj+Subcls and Conj+Subcls+Main.)

□ Main+Subcls+Conj	□ Subcls+Main+Conj
□ Main+Conj+Subcls	□ Subcls+Conj+Main
□ Conj+Main+Subcls	□ Conj+Subcls+Main

□ Otherwise, tick here if there doesn't exist the notion of subordinate conjunctions in your language.

Question 7: Transformational Affixes

Question 7.1: Infinitive Markers

What is the word order for the infinitive marker (Inf) and the verb phrase (VP) in your language?

□ Tick here if there exists the notion of infinitive markers in your language. Also tick the allowable word orders in the following table. (For example, English allows Inf+VP.)

□ Inf+VP □ VP+Inf

□ Otherwise, tick here if there doesn't exist the notion of infinitive markers in your language.

Question 7.2: Nominalizing Affixes

What is the word order for the nominalizing affixes?

□ Tick here if the nominalizing affixes (Nom) can combine with noun phrases (NP). Also tick the allowable word orders in the following table. (For example, Thai allows Nom+NP.)

□ Nom+NP □ NP+Nom

□ Tick here if the nominalizing affixes (Nom) can combine with verb phrases (VP). Also tick the allowable word orders in the following table. (For example, Thai allows Nom+VP.)

□ Nom+VP □ VP+Nom

□ Otherwise, tick here if there doesn't exist the notion of nomializing affixes in your language.

Question 8: Relocation and Dropping

Question 8.1: Dative Shift

Is dative shift allowed in your language?

□ Yes.
□ No.
□ I don't know.

Question 8.2: Dropping

Can you drop out the following parts of the sentence if the context is clear enough?

□ Subject.

□ Object.

□ Indirect object.

□ None of these.

Corpus-Specific Information

Which corpus are you inducing its grammar? (Example: PTB)

Please match the POS of the corpus to the generalized tagset provided in the following table.

Generalized Tagset	**Corpus-specific Tags**
Noun (`n`)	
Adjective (`adj`)	
Nominal modifier (`nmod`)	
Verb (`v`)	
Intransitive verb (`vi`)	
Transitive verb (`vt`)	
Ditransitive verb (`vd`)	
Complex verb (`vcomp`)	
Complex intransitive verb (`vicomp`)	
Complex transitive verb (`vtcomp`)	
Modal verb (`modal`)	
Copula (`copula`)	
Gerund (`gerund`)	
Adverb (`adv`)	
Particle (`part`)	
Adverbial modifier (`vmod`)	
Sentential modifier (`smod`)	
Gerund's modifier (`gmod`)	
Preposition/postposition (`adposition`)	
Relative pronoun (`relpro`)	
Conjunction (`conj`)	
Subordinate conjunction (`subconj`)	
Classifier (`cl`)	
substituting adjective (`adjcl`)	
substituting adverb (`advcl`)	
substituting nominal modifier (`nmodcl`)	
substituting adverbial modifier (`vmodcl`)	
Possessive marker (`poss`)	
Infinitive marker (`inf`)	
NP nominalizer (`npnom`)	
VP nominalizer (`vpnom`)	
Negator (`neg`)	
Verb phrase (`vp`)	
Adpositional phrase (`pp`)	

Appendix B

Dialog for Language Parameter Elicitation

Question 1: Sentence Structure

[Question 1] Translate the sentence *Mary gives John a flower* (pattern: [$_\text{S}$ Mary] [$_\text{V}$ gives] [$_\text{I}$ John] [$_\text{O}$ a flower]). Does he have to rephrase it as *Mary gives a flower to John* (or something equivalent) instead?

Question 2: Simple Modifiers

[Question 2.1] Translate the phrase *small kittens* (pattern: [$_\text{Adj}$ small] [$_\text{N}$ kittens]).

[Question 2.2] Translate the sentence *Mary sits quietly* (pattern: [$_\text{V}$ sits] [$_\text{Adv}$ quietly]).

[Question 2.3] Translate the phrase *strongly bitter tea* (pattern: [$_\text{Adv}$ strongly] [$_\text{Adj}$ bitter]).

[Question 2.4] Translate the following phrases/sentences: (1) *The car does not work* (pattern: [$_\text{Neg}$ not] [$_\text{V}$ work]); (2) *a not complex exercise* (pattern: [$_\text{Neg}$ not] [$_\text{Adj}$ complex]); (3) *not strongly bitter tea* (pattern: [$_\text{Neg}$ not] [$_\text{Adv}$ strongly]).

Question 3: Complex Verbs

[Question 3.1] Translate the sentences: (1) *John is a student*; (2) *John is tall*; (3) *John is in the classroom*. Is there anything equivalent to the verb *to be*?

[Question 3.2] Translate the sentence *Mary can swim* (pattern: [$_\text{Modal}$ can] [$_\text{V}$ swim]).

[Question 3.3] Translate the sentence *Mary wants to swim* (pattern: [$_{\text{V}}$ want] [$_{\text{C}}$ swim]).

[Question 3.4] Translate the sentence *John asks Mary to hold the door for him* (pattern: [$_{\text{V}}$ ask] [$_{\text{O}}$ Mary] [$_{\text{C}}$ hold the door]).

Question 4: Complex Modifiers

[Question 4.1] Translate the following phrases/sentences: (1) *a gift in the box* (pattern: [$_{\text{Prep}}$ in] [$_{\text{NP}}$ the box]); (2) *Mary walks into the classroom* (pattern: [$_{\text{Prep}}$ into] [$_{\text{NP}}$ the room]).

[Question 4.2] Translate the phrase *John's car* (pattern: [$_{\text{Owner}}$ John] [$_{\text{Poss}}$'s] [$_{\text{Ownee}}$ car]). Also ask the informant if he can directly say that or he has to rephrase it as *a car of John's* (or something equivalent) instead.

[Question 4.3] Translate the sentence *John lifts the box that contains many books* (pattern: [$_{\text{NP}}$ box] [$_{\text{Relpro}}$ that] [$_{\text{VP}}$ contains many books]).

[Question 4.4] Translate the sentences: (1) *John is the man on the bench* (pattern: [$_{\text{NP}}$ man] [$_{\text{NMod}}$ on the bench]); (2) John walks on the shore (pattern: [$_{\text{VP}}$ walk] [$_{\text{VMod}}$ on the shore]); (3) John is the man running on the shore (pattern: [$_{\text{Gerund}}$ running] [$_{\text{GMod}}$ on the shore]); (4) *On Monday, John will hand in his homework* (pattern: [$_{\text{SMod}}$ On Monday] [$_{\text{S}}$ John will hand in his homework]).

[Question 4.5] Ask the informant if there are any adverb-like words which seem to modify the verb, as in *over* in *Mary starts the process over* (pattern: [$_{\text{VP}}$ start the process] [$_{\text{Part}}$ over]).

[Question 4.5] Ask the informant if there are any adverb-like words which seem to modify the verb, as in *over* in *Mary starts the process over* (pattern: [$_{\text{VP}}$ start the process] [$_{\text{Part}}$ over]).

[Question 4.6] Translate the phrase *three cars*. Does he have to rephrase it as *three bodies of car* (pattern: [$_{\text{Num}}$ three] [$_{\text{CL}}$ bodies] [$_{\text{NP}}$ car])?

Question 5: Gerunds

[Question 5] Translate the following phrases/sentences: (1) *Running is good* ([$_{\text{Gerund}}$ running] as a noun phrase); (2) *a running man* ([$_{\text{Gerund}}$ running] as an adjectival);

(3) *John is running* ([$_{\text{Gerund}}$ running] as a non-finite verb). Check if any of these is grammatical in the language.

Question 6: Subordinate Conjunctions

[Question 6] Translate the following sentences: (1) *If you press this button, the door will open* (pattern: [$_{\text{Conj}}$ if] [$_{\text{Subcls}}$ you press this button] [$_{\text{Main}}$ the door will open]); (2) *The door will open if you press this button* (pattern: [$_{\text{Main}}$ the door will open] [$_{\text{Conj}}$ if] [$_{\text{Subcls}}$ you press this button]).

Question 7: Transformational Affixes

[Question 7.1] Translate the sentence *Mary carefully reads her draft to identify the inconsistency* (pattern: [$_{\text{Inf}}$ to] [$_{\text{VP}}$ identify the inconsistency]).

[Question 7.2] Ask the informant if: (1) there are any bound morphemes that transform a verb into a noun phrase such as *travel* > *traveler* (pattern: [$_{\text{VP}}$ travel] [$_{\text{Nom}}$ -er]); (2) there are any bound morphemes that augment the meaning of a noun, such as the Thai bound morpheme *nák* in *tennis* 'tennis' > *nák tennis* 'tennis player' (pattern: [$_{\text{Nom}}$ nák] [$_{\text{NP}}$ tennis]). Note that each bound morpheme does not have any meaning on its own.

Question 8: Relocation and Dropping

[Question 8.1] Translate the sentence *John introduces to Mary his long-time friends from high school* (pattern: [$_{\text{Dative}}$ to Mary] [$_{\text{O}}$ his long-time friends from high school]). Also ask the informant if he has to relocate the dative part to a particular position if the direct object is elongated.

[Question 8.2] Translate the sentence *Mary gives John a flower* (pattern: pattern: [$_{\text{S}}$ Mary] [$_{\text{V}}$ gives] [$_{\text{I}}$ John] [$_{\text{O}}$ a flower]) and consider the grammaticality of the following omissions: (1) (*She*) *gives John a flower*; (2) *Mary gives* (*him*) *a flower*; (3) *Mary gives John* (*it*).

Bibliography

[Adriaans, 1992] P. W. Adriaans. 1992. *Language Learning from a Categorial Perspective*. Ph.D. thesis, Universiteit van Amsterdam.

[Adriaans, 1999] P. W. Adriaans. 1999. Learning shallow context-free languages under simple distributions. Technical Report ILLC Report PP-1999-13, Institute for Logic, Language, and Computation, Amsterdam, the Netherland.

[Afonso et al., 2002] S. Afonso, E. Bick, R. Haber, and D. Santos. 2002. Floresta Sinta(c)tica: a treebank for Portuguese. In *Proceedings of LREC*.

[Ajdukiewicz, 1935] Kazimierz Ajdukiewicz. 1935. Die Syntaktische Konnexität. *Polish Logic*, pages 207–231.

[Attias, 2000] Hagai Attias. 2000. A variational Bayesian framework for graphical models. In *Advances in Neural Information Processing Systems (NIPS 2000)*.

[Baker, 1979] J. K. Baker. 1979. Trainable grammars for speech recognition. In D. H. Klatt and J. J. Wolf, editors, *Speech Communication Papers for the 97th Meeting of the Acoustical Society of America*, pages 547–550.

[Baldridge and Kruijff, 2003] Jason Baldridge and Geert-Jan M. Kruijff. 2003. Multimodal combinatory categorial grammar. In *Proceedings of the 10th Conference of the European Chapter of the ACL 2003*, pages 211–218, Budapest, Hungary.

[Bar-Hillel, 1953] Yehoshua Bar-Hillel. 1953. A Quasi-Arithmetical Notation for Syntactic Description. *Language*, 29:47–58.

[Baum et al., 1970] L. Baum, T. Petrie, G. Soules, and N. Weiss. 1970. A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *The Annals of Mathematical Statistics*, 41(1):164–171, February.

[Beal, 2003] Matthew J. Beal. 2003. *Variational Algorithms for Approximate Bayesian Inference*. Ph.D. thesis, Gatsby Computational Neuroscience Unit, University of London.

[Bisk and Hockenmaier, 2012a] Yonatan Bisk and Julia Hockenmaier. 2012a. Induction of linguistic structure with combinatory categorial grammars. In *Proceedings of the NAACL-HLT Workshop on the Induction of Linguistic Structure*, pages 90–95. ACL.

[Bisk and Hockenmaier, 2012b] Yonatan Bisk and Julia Hockenmaier. 2012b. Simple robust grammar induction with combinatory categorial grammar. In *Proceedings of the 26th National Conference on Artificial Intelligence (AAAI)*, pages 1643–1648. AAAI.

[Black et al., 1992] Erza Black, John Lafferty, and Salim Roukos. 1992. Development and evaluation of a broad-coverage probabilistic grammar of English-language computer manuals. In *Proceedings of 30th Annual Meeting on Association for Computational Linguistics*.

[Bohomovà et al., 2001] A. Bohomovà, J. Hajic, E. Hajicova, and B. Hladka. 2001. The Prague dependency treebank: Three-level annotation scenario. In Anne Abeillé, editor, *Treebanks: Building and Using Syntactically Annotated Corpora*.

[Boonkwan and Steedman, 2011] Prachya Boonkwan and Mark Steedman. 2011. Grammar induction from text using small syntactic prototypes. In *Proceedings of the 5th IJCNLP*, pages 438–446.

[Boyer and Moore, 1972] R. S. Boyer and J. S. Moore. 1972. The sharing of structure in theorem-proving programs. In J. Bresnan, editor, *Machine Intelligence*, volume 7, pages 101–116. Edinburgh University Press.

[Brants et al., 2002] S. Brants, S. Dipper, S. Hansen, W. Lezius, and G. Smith. 2002. The TIGER treebank. In *Proceedings Workshop on Treebanks and Linguistic Theories*.

[Buchholz and Marsi, 2006] Sabine Buchholz and Erwin Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of CoNLL-2006*, pages 149–164.

[Carroll and Charniak, 1992] Glenn Carroll and Eugene Charniak. 1992. Two experiments on learning probabilistic dependency grammars from corpora. In C. Weir, S. Abney, R. Grishman, and R. Weischedel, editors, *Working Notes of the Workshop Statistically-Based NLP Techniques*, pages 1–13. AAAI Press, Menlo Park, CA.

[Charniak, 1997] Eugene Charniak. 1997. Statistical parsing with a context-free grammar and word statistics. In *Proceedings of the 14th National Conference on Artificial Intelligence*, Menlo Park, CA. AAAI Press/MIT Press.

[Chen, 1995] S. F. Chen. 1995. Bayesian grammar induction for language modeling. In *Proceedings of the 33rd Meeting of the Association for Computational Linguistics*, pages 228–235.

[Chomsky, 1964] Noam Chomsky. 1964. Current issues in linguistic theory. *Janua Linguarum*, (38).

[Chomsky, 1965] Noam Chomsky. 1965. *Aspects of the Theory of Syntax*. MIT Press.

[Civit and Martí, 2004] M. Civit and M. A. Martí. 2004. Bulding Cast3lb: A Spanish treebank. In *Research on Language & Computation*.

[Clark and Curran, 2007] Stephen Clark and James R. Curran. 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33(4):493–552, December.

[Clark and Lappin, 2010] Alexander Clark and Shalom Lappin. 2010. Unsupervised learning and grammar induction. In Alexander Clark, Chris Fox, and Shalom Lappin, editors, *Computational Linguistics and Natural Language Processing Handbook*. Wiley-Blackwell, Oxford.

[Clark and Lappin, 2011] Alexander Clark and Shalom Lappin. 2011. Computational learning theory and language acquisition. In Ruth Kempson, Nicholas Asher, and Tim Fernando, editors, *Handbook of Philosophy of Linguistics*. Elsevier/MIT Press.

[Clark, 2000] Alexander Clark. 2000. Inducing syntactic categories by context distribution clustering. In *Proceedings of CoNLL-2000 and LLL-2000*, pages 91–94.

[Clark, 2001] Alexander Clark. 2001. Unsupervised induction of stochastic context-free grammars using distributional clustering. In *Proceedings of the 5th Conference on Natural Language Learning*.

[Cocke and Schwartz, 1970] John Cocke and Jacob T. Schwartz. 1970. Programming languages and their compilers: Preliminary notes. Technical report, Courant Institute of Mathematical Sciences, New York University.

[Cohen et al., 2008] Shay B. Cohen, Kevin Gimpel, and Noah A. Smith. 2008. Logistic normal priors for unsupervised probabilistic grammar induction. In *Advances in Neural Information Processing Systems 21*.

[Cohen et al., 2010] Shay B. Cohen, David M. Blei, and Noah A. Smith. 2010. Variational inference for adaptor grammars. In *Proceedings of Human Language Technologies: 2010 Annual Conference of NAACL*, pages 564–572.

[Cohn et al., 2009] Trevor Cohn, Sharon Goldwater, and Phil Blunsom. 2009. Inducing compact but accurate tree-substitution grammars. In *Proceedings of Human Language Technologies: 2009 Annual Conference of NAACL*, pages 548–556.

[Cohn et al., 2010] Trevor Cohn, Phil Blunsom, and Sharon Goldwater. 2010. Inducing tree-substitution grammars. *The Journal of Machine Learning Research*, 9999:3053–3096.

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

[Cristofaro, 2011a] Sonia Cristofaro. 2011a. Purpose clauses. In Matthew S. Dryer and Martin Haspelmath, editors, *The World Atlas of Language Structures Online*. Max Planck Digital Library, Munich.

[Cristofaro, 2011b] Sonia Cristofaro. 2011b. Reason clauses. In Matthew S. Dryer and Martin Haspelmath, editors, *The World Atlas of Language Structures Online*. Max Planck Digital Library, Munich.

[Cristofaro, 2011c] Sonia Cristofaro. 2011c. Utterance complement clauses. In Matthew S. Dryer and Martin Haspelmath, editors, *The World Atlas of Language Structures Online*. Max Planck Digital Library, Munich.

[Cristofaro, 2011d] Sonia Cristofaro. 2011d. 'when' clauses. In Matthew S. Dryer and Martin Haspelmath, editors, *The World Atlas of Language Structures Online*. Max Planck Digital Library, Munich.

[Dell'Orletta et al., 2011] Felice Dell'Orletta, Giulia Venturi, and Simonetta Montemagni. 2011. ULISSE: an unsupervised algorithm for detecting reliable dependency parses. In *Proceedings of the 15th Conference on CoNLL 2011*, pages 115–124.

[Dempster et al., 1977] A. P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society*, 39(1):1–38.

[Dodd, 1988] L. Dodd. 1988. Grammatical inference for automatic speech recognition: an application of the inside/outside algorithm and the spelling of English words. In *Proceedings of the 7th FASE Symposium*, pages 1061–1068, Edinburgh.

[Druck et al., 2009] Gregory Druck, Gideon Mann, and Andrew McCallum. 2009. Semi-supervised learning of dependency parsers using generalized expectation criteria. In *Proceedings of 47th Annual Meeting of the Association of Computational Linguistics and the 4th IJCNLP of the AFNLP*, pages 360–368, Suntec, Singapore, August.

[Dryer, 2011a] Matthew S. Dryer. 2011a. Order of adjective and noun. In Matthew S. Dryer and Martin Haspelmath, editors, *The World Atlas of Language Structures Online*. Max Planck Digital Library, Munich.

[Dryer, 2011b] Matthew S. Dryer. 2011b. Order of adposition and noun phrase. In Matthew S. Dryer and Martin Haspelmath, editors, *The World Atlas of Language Structures Online*. Max Planck Digital Library, Munich.

[Dryer, 2011c] Matthew S. Dryer. 2011c. Order of adverbial subordinator and clause. In Matthew S. Dryer and Martin Haspelmath, editors, *The World Atlas of Language Structures Online*. Max Planck Digital Library, Munich.

[Dryer, 2011d] Matthew S. Dryer. 2011d. Order of degree word and adjective. In Matthew S. Dryer and Martin Haspelmath, editors, *The World Atlas of Language Structures Online*. Max Planck Digital Library, Munich.

[Dryer, 2011e] Matthew S. Dryer. 2011e. Order of demonstrative and noun. In Matthew S. Dryer and Martin Haspelmath, editors, *The World Atlas of Language Structures Online*. Max Planck Digital Library, Munich.

[Dryer, 2011f] Matthew S. Dryer. 2011f. Order of genitive and noun. In Matthew S. Dryer and Martin Haspelmath, editors, *The World Atlas of Language Structures Online*. Max Planck Digital Library, Munich.

[Dryer, 2011g] Matthew S. Dryer. 2011g. Order of object and verb. In Matthew S. Dryer and Martin Haspelmath, editors, *The World Atlas of Language Structures Online*. Max Planck Digital Library, Munich.

[Dryer, 2011h] Matthew S. Dryer. 2011h. Order of relative clause and noun. In Matthew S. Dryer and Martin Haspelmath, editors, *The World Atlas of Language Structures Online*. Max Planck Digital Library, Munich.

[Dryer, 2011i] Matthew S. Dryer. 2011i. Order of subject and verb. In Matthew S. Dryer and Martin Haspelmath, editors, *The World Atlas of Language Structures Online*. Max Planck Digital Library, Munich.

[Dryer, 2011j] Matthew S. Dryer. 2011j. Order of subject, object and verb. In Matthew S. Dryer and Martin Haspelmath, editors, *The World Atlas of Language Structures Online*. Max Planck Digital Library, Munich.

[Dryer, 2011k] Matthew S. Dryer. 2011k. Polar questions. In Matthew S. Dryer and Martin Haspelmath, editors, *The World Atlas of Language Structures Online*. Max Planck Digital Library, Munich.

[Dryer, 2011l] Matthew S. Dryer. 2011l. Position of polar question particles. In Matthew S. Dryer and Martin Haspelmath, editors, *The World Atlas of Language Structures Online*. Max Planck Digital Library, Munich.

[Dryer, 2011m] Matthew S. Dryer. 2011m. Relationship between the order of object and verb and the order of adjective and noun. In Matthew S. Dryer and Martin Haspelmath, editors, *The World Atlas of Language Structures Online*. Max Planck Digital Library, Munich.

[Dryer, 2011n] Matthew S. Dryer. 2011n. Relationship between the order of object and verb and the order of adposition and noun phrase. In Matthew S. Dryer and Martin Haspelmath, editors, *The World Atlas of Language Structures Online*. Max Planck Digital Library, Munich.

[Dryer, 2011o] Matthew S. Dryer. 2011o. Relationship between the order of object and verb and the order of relative clause and noun. In Matthew S. Dryer and Martin

Haspelmath, editors, *The World Atlas of Language Structures Online*. Max Planck Digital Library, Munich.

[Džeroski et al., 2006] S. Džeroski, T. Erjavec, N. Ledinek, P. Pajas, Z. Žabokrtsky, and A. Žele. 2006. Towards a Slovene dependency treebank. In *Proceedings of LREC*.

[Elman et al., 1996] J. Elman, E. Bates, M. H. Johnson, A. Karmiloff-Smith, D. Parisi, and K. Plunkett. 1996. *Rethinking Innateness: A Connectionist Perspective on Development*. MIT Press/Bradford Books, Cambridge, Massachusetts.

[Forney, 1973] George David Forney. 1973. The Viterbi algorithm. In *Proceedings of the IEEE*, volume 61, pages 268–278.

[Ganchev et al., 2010] Kuzman Ganchev, João Graça, Jennifer Gillenwater, and Ben Taskar. 2010. Posterior regularization for structured latent variable models. *Journal of Machine Learning Research*, 11:2001–2049.

[Gelling and Cohn, 2012] Douwe Gelling and Trevor Cohn. 2012. The PASCAL Challenge on grammar induction. In *NAACL-HLT Workshop on the Induction of Linguistic Structure*, pages 64–80.

[Geman and Geman, 1984] Stuart Geman and Donald Geman. 1984. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741.

[Ghahramani and Beal, 2000] Zoubin Ghahramani and Matthew J. Beal. 2000. Variational inference for Bayesian mixtures of factor analyses. In *Advances in Neural Information Processing Systems (NIPS 2000)*.

[Gil, 2011] David Gil. 2011. Numeral classifiers. In Matthew S. Dryer and Martin Haspelmath, editors, *The World Atlas of Language Structures Online*. Max Planck Digital Library, Munich.

[Gillenwater et al., 2010] Jennifer Gillenwater, Kuzman Ganchev, João Graça, Fernando Pereira, and Ben Taskar. 2010. Sparsity in dependency grammar induction. In *Proceedings of ACL-2010 Short Papers*, pages 194–199.

[Goldwater, 2007] Sharon Goldwater. 2007. *Nonparametric Bayesian Models of Lexical Acquisition*. Ph.D. thesis, Department of Cognitive and Linguistic Sciences, Brown University, Providence, Rhode Island, May.

[Haghighi and Klein, 2006] Aria Haghighi and Dan Klein. 2006. Prototype-driven grammar induction. In *Proceedings of 44th Annual Meeting of the Association for Computational Linguistics*, pages 881–888.

[Haspelmath et al., 2005] Martin Haspelmath, Matthew S. Dryer, David Gil, and Bernard Comrie. 2005. *The World Atlas of Language Structures*. Oxford University Press, `http://wals.info/`, July.

[Haspelmath, 2011a] Martin Haspelmath. 2011a. Ditransitive constructions: The verb 'give'. In Matthew S. Dryer and Martin Haspelmath, editors, *The World Atlas of Language Structures Online*. Max Planck Digital Library, Munich.

[Haspelmath, 2011b] Martin Haspelmath. 2011b. 'want' complement subjects. In Matthew S. Dryer and Martin Haspelmath, editors, *The World Atlas of Language Structures Online*. Max Planck Digital Library, Munich.

[Hastings, 1970] W. K. Hastings. 1970. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57:97–109.

[Headden III et al., 2009] William P. Headden III, Mark Johnson, and David McClosky. 2009. Improving unsupervised dependency parsing with richer contexts and smoothing. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics*, Boulder, Colorado, June.

[Henrichsen, 2002] P. J. Henrichsen. 2002. Grasp: Grammar learning from unlabeled speech corpora. In *Proceedings of CoNLL-2002*, pages 22–28, Taipei, Taiwan.

[Hockenmaier and Steedman, 2007] Julia Hockenmaier and Mark Steedman. 2007. CCGbank: A corpus of CCG derivations and dependency structures extracted from the Penn Treebank. *Computational Linguistics*, 33(3):355–396.

[Hockenmaier, 2003a] Julia Hockenmaier. 2003a. *Data and Models for Statistical Parsing with Combinatory Categorial Grammar*. Ph.D. thesis, University of Edinburgh.

[Hockenmaier, 2003b] Julia Hockenmaier. 2003b. Parsing with generative models of predicate-argument structure. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 359–366, Sapporo, Japan.

[Jelinek, 1985] F. Jelinek. 1985. Markov source modeling of text generation. In *Impact of Processing Techniques on Communication*, pages 569–598.

[Johansson and Nugues, 2007] Richard Johansson and Pierre Nugues. 2007. Extended constituent-to-dependency conversion for english. In *Proceedings of NODALIDA 2007*.

[Johnson and Demuth, 2010] Mark Johnson and Katherine Demuth. 2010. Unsupervised phonemic chinese word segmentation using adaptor grammars. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING 2010)*, pages 528–536.

[Johnson and Goldwater, 2009] Mark Johnson and Sharon Goldwater. 2009. Improving nonparametric bayesian inference: Experiments on unsupervised word segmentation with adaptor grammars. In *Proceedings of Human Language Technologies: 2009 Annual Conference of NAACL*, pages 317–325.

[Johnson et al., 2007a] Mark Johnson, Thomas L. Griffiths, and Sharon Goldwater. 2007a. Adaptor grammars: A framework for specifying compositional nonparametric bayesian models. *Advances in Neural Information Processing Systems*, 19.

[Johnson et al., 2007b] Mark Johnson, Thomas L. Griffiths, and Sharon Goldwater. 2007b. Bayesian inference for pcfgs via markov chain monte carlo. In *Proceedings of NAACL 2007*.

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

[Johnson, 2008] Mark Johnson. 2008. Using adaptor grammars to identify synergies in the unsupervised acquisition of linguistic structure. In *Proceedings of ACL-2008: HLT*, pages 398–406.

[Kasami, 1965] Tadao Kasami. 1965. An efficient recognition and syntax-analysis algorithm for context-free languages. Technical report, Air Force Cambridge Research Lab, Bedford, MA.

[Kawata and Bartels, 2000] Y. Kawata and J. Bartels. 2000. Stylebook for the Japanese treebank in VERBMOBIL. Technical report, Eberhard-Karls-Universität Tübingen.

[Keh-Liann and Hsieh, 2004] Chen Keh-Liann and Yu-Ming Hsieh. 2004. Chinese treebanks and grammar extraction. In *Proceedings of IJCNLP-2004*, pages 560–565.

[Klein and Manning, 2001a] Dan Klein and Christopher D. Manning. 2001a. Distributional phrase structure induction. In *Proceedings of the 5th Conference on Natural Language Learning*, pages 113–120.

[Klein and Manning, 2001b] Dan Klein and Christopher D. Manning. 2001b. Natural language grammar induction using a constituent-context model. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems (NIPS 2001)*, volume 1, pages 35–42. MIT Press.

[Klein and Manning, 2002] Dan Klein and Christopher D. Manning. 2002. A generative constituent-context model for improved grammar induction. In *Proceedings of the 40th Associations for Computational Linguistics*, pages 128–135.

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

[Klein and Manning, 2004] Dan Klein and Christopher D. Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*.

[Klein, 2005] Dan Klein. 2005. *The Unsupervised Learning of Natural Language Structure*. Ph.D. thesis, Stanford University, March.

[Koller and Kuhlmann, 2009] Alexander Koller and Marco Kuhlmann. 2009. Dependency trees and the strong generative capacity of ccg. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 460–468, April.

[Kromann et al., 2003] M. T. Kromann, L. Mikkelsen, and S. K. Lynge. 2003. Danish dependency treebank. In *Proceedings of TLT*.

[Kurihara and Sato, 2006] Kenichi Kurihara and Taisuke Sato. 2006. Variational Bayesian grammar induction for natural language. In *International Colloquium on Grammatical Inference*, pages 84–96.

[Lari and Young, 1990] K. Lari and S. J. Young. 1990. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech and Language*, 4:35–56.

[Liang et al., 2007] Percy Liang, Slav Petrov, Michael I. Jordan, and Dan Klein. 2007. The infinite pcfg using hierarchical dirichlet processes. In *Proceedings of the 2007 Joint Conference on EMNLP and CoNLL*, pages 688–697.

[Marcus et al., 1993] Mitchell P. Marcus, Beatrice Santorini, and Mary A. Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19:313–330.

[Metropolis et al., 1953] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. 1953. Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21(6):1087–1092.

[Miestamo, 2011a] Matti Miestamo. 2011a. Subtypes of asymmetric standard negation. In Matthew S. Dryer and Martin Haspelmath, editors, *The World Atlas of Language Structures Online*. Max Planck Digital Library, Munich.

[Miestamo, 2011b] Matti Miestamo. 2011b. Symmetric and asymmetric standard negation. In Matthew S. Dryer and Martin Haspelmath, editors, *The World Atlas of Language Structures Online*. Max Planck Digital Library, Munich.

[Moore, 1973] J. S. Moore. 1973. *Computational Logic: Structure Sharing and Proof of Program Properties*. Ph.D. thesis, Department of Computational Logic, University of Edinburgh.

[Naseem et al., 2010] Tahira Naseem, Harr Chen, Regina Barzilay, and Mark Johnson. 2010. Using universal linguistic knowledge to guide grammar induction. In *Proceedings of EMNLP-2010*.

[Nilsson et al., 2005] J. Nilsson, J. Hall, and J. Nivre. 2005. MAMBA meets TIGER: Reconstructing a Swedish treebank from antiquity. In *NODALIDA Special Session on Treebanks*.

[Oflazer et al., 2003] K. Oflazer, B. Say, D. Z. Hakkani-Tür, and G. Tür. 2003. Building a Turkish treebank. In *Treebanks: Building and Using Syntactically Annotated Corpora*.

[Osborne and Briscoe, 1997] Miles Osborne and Ted Briscoe. 1997. Learning stochastic categorial grammars. In *Proceedings of CoNLL-1997*, pages 80–87.

[Page and Brin, 1998] Larry Page and Sergey Brin. 1998. The anatomy of a large-scale hypertextual web search engine. *International Web Conference*.

[Reichart and Rappoport, 2009] Roi Reichart and Ari Rappoport. 2009. Automatic selection of high quality parses created by a fully unsupervised parser. In *Proceedings of 13th Conference on CoNLL*, pages 156–164.

[Rissanen, 1978] Jorma Rissanen. 1978. Modeling by the shortest data description. *Automatica*, 14:465–471.

[Rissanen, 1989] Jorma Rissanen. 1989. *Stochastic Complexity and Statistical Inquiry*. World Scientific Co., Singapore.

[Schabes et al., 1993] Yves Schabes, Michal Roth, and Randy Osborne. 1993. Parsing the Wall Street Journal with the inside-outside algorithm. In *Proceedings of the 6th Conference on European Chapter of the Association for Computational Linguistics*, pages 341–347.

[Schütze, 1995] Hinrich Schütze. 1995. Distributional part-of-speech tagging. In Morgan Kaufmann, editor, *Proceedings of the 7th Meeting of the European Chapter of the Association for Computational Linguistics*, pages 141–148, San Francisco, CA.

[Seginer, 2007] Yoav Seginer. 2007. Fast unsupervised incremental parsing. In *Proceedings of 45th Annual Meeting of ACL*, pages 384–391.

[Siewierska, 2011] Anna Siewierska. 2011. Third person zero of verbal person marking. In Matthew S. Dryer and Martin Haspelmath, editors, *The World Atlas of Language Structures Online*. Max Planck Digital Library, Munich.

[Simov et al., 2001] K. Simov, P. Osenova, M. Slavcheva, S. Kolkovska, E. Balabanova, D. Doikoff, K. Ivanova, A. Simov, E. Simov, and M. Kouylekov. 2001. Building a linguistically interpreted corpus of Bulgarian: The Bultreebank. In *Proceedings of LREC*.

[Smith, 2006] Noah A. Smith. 2006. *Novel Estimation Methods for Unsupervised Discovery of Latent Structure in Natural Language Text*. Ph.D. thesis, Department of Computer Science, John Hopkins University.

[Smrž et al., 2002] Otakar Smrž, Jan Šnaldauf, and Petr Zemánek. 2002. Prague dependency treebank for Arabic: Multi-level annotation of Arabic corpus. In *Proceedings of International Symposium on Processing of Arabic*, pages 147–155.

[Snyder et al., 2009] Benjamin Snyder, Tahira Naseem, and Regina Barzilay. 2009. Unsupervised multilingual grammar induction. In *Proceedings of the Joint Conference of the 47th ACL and the 4th IJCNLP*.

[Søgaard, 2011] Anders Søgaard. 2011. From ranked words to dependency trees: Two-staged unsupervised nonprojective dependency parsing. In *Proceedings of the TextGraphs-6 Workshop*, pages 60–68.

[Song, 2011a] Jae Jung Song. 2011a. Nonperiphrastic causative constructions. In Matthew S. Dryer and Martin Haspelmath, editors, *The World Atlas of Language Structures Online*. Max Planck Digital Library, Munich.

[Song, 2011b] Jae Jung Song. 2011b. Periphrastic causative constructions. In Matthew S. Dryer and Martin Haspelmath, editors, *The World Atlas of Language Structures Online*. Max Planck Digital Library, Munich.

[Spitkovsky and Alshawi, 2011a] Valentin I. Spitkovsky and Hiyan Alshawi. 2011a. Lateen EM: Unsupervised training with multiple objectives applied to dependency grammar induction. In *Proceedings of EMNLP-2011*, pages 1269–1280.

[Spitkovsky and Alshawi, 2011b] Valentin I. Spitkovsky and Hiyan Alshawi. 2011b. Punctuation: Making a point in unsupervised dependency parsing. In *Proceedings of the 15th Conference on CoNLL 2011*, pages 19–28.

[Spitkovsky et al., 2010] Valentin I. Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. 2010. From baby steps to leapfrog: How “less is more” in unsupervised dependency parsing. In *Proceedings of NAACL-HLT 2010*.

[Spitkovsky et al., 2011] Valentin I. Spitkovsky, Hiyan Alshawi, Angel X. Chang, and Daniel Jurafsky. 2011. Unsupervised dependency parsing without gold part-of-speech tags. In *Proceedings of the 2011 Conference on EMNLP*, pages 1281–1290.

[Steedman, 2000] Mark Steedman. 2000. *The Syntactic Process*. The MIT Press, Cambridge, Massachusetts.

[Stolcke and Omohundro, 1994] A. Stolcke and S. M. Omohundro. 1994. Inducing probabilistic grammars by Bayesian model merging. In *Proceedings of the 2nd International Colloquium on Grammatical Inference*. Springer-Verlag.

[Surdeanu et al., 2008] Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The CoNLL-2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proceedings of the 12th Conference on Computational Natural Language Learning (CoNLL).*

[Thomforde and Steedman, 2011] Emily Thomforde and Mark Steedman. 2011. Semi-supervised CCG lexicon extension. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1246–1256. ACL.

[Tomita, 1987] M. Tomita. 1987. An efficient augmented-context-free parsing algorithm. *Computational Linguistics*, 13(1–2):31–46, January–June.

[Tsarfaty et al., 2011] Reut Tsarfaty, Joakim Nivre, and Evelina Andersson. 2011. Evaluating dependency parsing: Robust and heuristics-free cross-annotation evaluation. In *Proceedings of 8th EMNLP*, Edinburgh, UK, 27–29 July 2011.

[Tsarfaty et al., 2012a] Reut Tsarfaty, Joakim Nivre, and Evelina Andersson. 2012a. Cross-framework evaluation for statistical parsing. In *Proceedings of EACL*, France.

[Tsarfaty et al., 2012b] Reut Tsarfaty, Joakim Nivre, and Evelina Andersson. 2012b. Joint evaluation of morphological segmentation and syntactic parsing. In *Proceedings of ACL 2012*, Korea.

[Tu, 2012] Kewei Tu. 2012. Combining the sparsity and unambiguity biases for grammar induction. In *Proceedings of the NAACL-HLT Workshop on the Induction of Linguistic Structure*.

[van der Beek et al., 2002] L. van der Beek, G. Bouma, R. Malouf, and G. van Noord. 2002. The Alpino dependency treebank. In *Language and Computers*.

[van Zaanen, 2002] M. van Zaanen. 2002. *Bootstrapping Structure into Language: Alignment-Based Learning*. Ph.D. thesis, School of Computing, University of Leeds.

[Villavicencio, 2002] Aline Villavicencio. 2002. *The Acquisition of a Unification-based Generalized Categorial Grammar*. Ph.D. thesis, University of Cambridge, April.

[Viterbi, 1967] Andrew J. Viterbi. 1967. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transaction: Information Theory*, 13(2):260–269.

[Watkinson and Manadhar, 2001] Stephen Watkinson and Suresh Manadhar. 2001. A psychologically plausible and computationally effective approach to learning syntax. In *The Workshop on Computational Natural Language Learning, ACL/EACL 2001*.

[Yang, 2008] Charles Yang. 2008. The great number crunch. *Journal of Linguistics*, 44(1):205–228.

[Younger, 1967] Daniel H. Younger. 1967. Recognition and parsing of context-free languages in time n^3. *Information and Control*, 10(2):189–208.