Computational Learning of Probabilistic Grammars in the Unsupervised Setting

Shay Cohen

CMU-LTI-11-016

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
5000 Forbes Ave., Pittsburgh, PA 15213
www.lti.cs.cmu.edu

Thesis Committee:

Noah A. Smith (chair), Carnegie Mellon University
John Lafferty, Carnegie Mellon University
Eric P. Xing, Carnegie Mellon University
Mark Johnson, Macquarie University

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
In Language and Information Technologies

Abstract

With the rising amount of available multilingual text data, computational linguistics faces an opportunity and a challenge. This text can enrich the domains of NLP applications and improve their performance. Traditional supervised learning for this kind of data would require annotation of part of this text for induction of natural language structure. For these large amounts of rich text, such an annotation task can be daunting and expensive. Unsupervised learning of natural language structure can compensate for the need for such annotation.

Natural language structure can be modeled through the use of probabilistic grammars, generative statistical models which are useful for compositional and sequential structures. Probabilistic grammars are widely used in natural language processing, but they are also used in other fields as well, such as computer vision, computational biology and cognitive science. This dissertation focuses on presenting a theoretical and an empirical analysis for the learning of these widely used grammars in the unsupervised setting.

We analyze computational properties involved in estimation of probabilistic grammars: the computational complexity of the inference problem and the sample complexity of the learning problem. We show that the common inference problems for probabilistic grammars are computationally hard, even though a polynomial sample is sufficient for accurate estimation. We also give a variational inference framework for estimation of probabilistic grammars in the empirical Bayesian setting, which permits the use of non-conjugate priors with probabilistic grammars as well as parallelizable inference. The estimation techniques we use include two types of priors on probabilistic grammars: logistic normal priors and adaptor grammars. We further extend the logistic normal priors to shared logistic normal priors, which define a distribution over a collection of multinomials that represent a probabilistic grammar.

We test our estimation techniques on treebanks in eleven languages. Our empirical evaluation shows that our estimation techniques are useful and perform better than several Bayesian and non-Bayesian baselines.

Acknowledgements

I would first like to thank my thesis advisor, Noah Smith, for his constant support and encouragement, knowledge and wisdom and for the many useful discussions we have shared, as well as the freedom and independence he has given me to also pursue what I have a passion for. I will always feel fortunate to be one of his first students.

I am also especially grateful to my committee member, Mark Johnson, whose work inspired some of the work in this thesis. His feedback and support through my graduate studies have been incredibly helpful.

I want to thank the remaining members of my thesis committee: John Lafferty and Eric Xing, for taking the time and effort to provide useful feedback about my work.

I am also grateful for all the great time that I have been fortunate to spend with our group members, past and present: Desai Chen, Dipanjan Das, Chris Dyer, Philip Gianfortoni, Kevin Gimpel, Mohammad Haque, Michael Heilman, André Martins, Zack McCord, Daniel Mills, Brendan O'Connor, Naomi Saphra, Nathan Schneider, Cari Sisson, Dan Tasse, Mengqiu Wang, Tae Yano and Dani Yogatama. Even those with whom I did not have the opportunity to actively collaborate on a project were always helpful and happy to discuss anything I had an interest in.

I was also fortunate to enjoy a summer internship in Princeton University. There, I had the opportunity to work closely with Dave Blei (part of the work from this internship actually appears in this thesis). There I also had a chance to work and converse with Jordan Boyd-Graber, Jonathan Chang, Sean Gerrish, and Chong Wang. I will always remember this summer fondly.

There have been other researchers in the community with whom I have enjoyed talking and meeting in conferences, and on occasion, collaborating with as well. This includes Regina Barzilay, Michael Collins, Jason Eisner (who is my grand-advisor), Sharon Goldwater, Carlos Gómez-Rodríguez, Rebecca Hwa, Adam Lopez, Dan Roth, Giorgio Satta (from whom I learned a lot about parsing via intensive Skype meetings), and David Smith.

I want to also thank Steve Hanneke for being there whenever I wanted to discuss an idea and get feedback. His support also inspired some of the work in this thesis.

I also have been fortunate to meet other people in CMU who have been there for me when I needed them, whether just to catch a conversation, work on a project, or discuss an idea, and with whom I truly enjoyed interacting: Alon Lavie, Lori Levin, Jason Adams, Amr Ahmed, Jacob Eisenstein, Anagha Kulkarni, Kriti

Puniyani, Rob Simmons, Babis Tsourakakis, Reza Zadeh and Andreas Zollmann.

On more of a personal note, I want to thank my wife, Sylvia, who has been there for me almost from the beginning of this journey, witnessing almost all of my “triumphs and tribulations”. Her warm and welcoming family has made my stay at CMU more pleasant, while my family was far away in Israel.

Lastly, I want to thank my family, especially my parents, my sister and my brother. They have given me all the support I could ever hope for, through all the years, unconditionally.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

With the rising amounts of multilingual text data becoming available, computational linguistics faces an opportunity and a challenge. This text can enrich the domains of NLP applications and improve their performance. Traditional supervised learning for this kind of data would require annotating part of this text. For these large amounts of rich text, such an annotation task can be daunting and expensive.

From a practical point of view, unsupervised language learning has the potential to compensate for such missing annotations, or even save the effort required to create annotations to begin with.[1]

Using unsupervised learning, however, requires giving more attention to ensuring that the model used during learning accurately depicts the properties of language. With supervised learning, we can fit even a "bad" model to annotated data and achieve reasonable performance. However, with unsupervised learning, success hinges on an accurate model design with reasonable modeling assumptions. Probabilistic grammars offer a flexible way of designing such models and making the modeling assumptions required for modeling language.

It is for this reason that the use of probabilistic grammars and unsupervised learning together has led to a fruitful line of research. Like symbolic grammars,

[1] This argument, which is the most widely accepted argument motivating unsupervised learning in NLP, should be stated carefully. More specifically, much of the NLP evaluation methodology, such as the one used in this thesis, is based on comparison of the output of an unsupervised learner to gold-standard annotated data, and this does not eliminate the use of annotated data altogether. Although we use annotated data for unsupervised learning evaluation, the ultimate goal, of an unsupervised learner is for it to be used as a building block in a larger system without the need of annotated data. For example, an unsupervised dependency parser could be used in a machine translation system. This can happen as the field of unsupervised learning matures.

probabilistic grammars are amenable to human inspection, making it relatively easy to understand the tendencies captured by the model, given that the underlying rules are understandable (Johnson, 1998); unlike purely symbolic grammars, probabilistic grammars model frequency and provide a mechanism for reasoning in the face of ambiguity, which is ubiquitous in natural language data (Manning, 2003).

Probabilistic grammars are valuable in various settings in computational linguistics. Applications of probabilistic grammars include multilingual (mostly English) parsing (Collins, 2003; Klein and Manning, 2003b; Charniak and Johnson, 2005; Cohen and Smith, 2007), machine translation (Wu, 1997; Ding and Palmer, 2005; Chiang, 2005) and question answering (Wang et al., 2007). The models for these applications contain probabilistic grammars, either as a supporting structure with additional features or as the main structure of the model. Examples for such probabilistic grammars include context-free grammars (Charniak, 1996), tree adjoining grammars (Joshi, 2003) with an added probabilistic interpretation, and combinatory categorial grammar (Hockenmaier and Steedman, 2002; Clark and Curran, 2007).

In spite of their advantages, naïvely using probabilistic grammars for unsupervised learning results in low performance (Klein and Manning, 2004). However, as statistical models, probabilistic grammars lend themselves well to augmentation by other statistical models. For example, probabilistic grammars have been used with parametric Bayesian priors (Johnson et al., 2007; Post and Gildea, 2009), as well as nonparametric ones (Johnson et al., 2006; Liang et al., 2007; Finkel et al., 2007). This line of research, where the performance of a probabilistic grammar is improved through the use of more advanced statistical modeling techniques, is far from being exhausted, as new advances show that better results can be achieved through the improvement of estimation and modeling techniques, without necessarily changing the underlying grammar. This is indeed the main goal of this thesis: to advance state of the art in estimating probabilistic grammars in the unsupervised setting and provide theoretical insight into the properties of grammar estimation.

1.1 Contributions and Thesis Statement

In this thesis, we present analysis and methodology of the estimation of probabilistic grammars in the unsupervised setting. We take an approach in which we assume that a family of probabilistic grammars is given (where a probabilistic

grammar corresponds to a grammar originating in a grammar formalism coupled with a set of parameters to make it stochastic) and we need to identify a set of parameters that fit a collection of sentences that we receive as an input to the estimation procedure.

Estimating the parameters of the model can lead to a lightweight model that enables fast processing of unseen sentences using a parsing algorithm for the corresponding probabilistic grammar. This is especially useful for natural language applications, in which we are required to use a parser as a preprocessing step, and we would rather it be fast and flexible.

The principle that we follow in our estimation procedures is that of the *maximum likelihood principle* or one of its variants. In the first part of the thesis, we analyze some of the computational challenges that we have in following the MLE principle in the unsupervised setting. Our results show that these challenges are manifested both through algorithmic issues as well as issues relating to the learning-theoretic complexity of a family of probabilistic grammars in the unsupervised setting. These results align well with the empirical evidence showing that plain likelihood maximization does not necesasrily lead to well-performing models.

To better estimate grammatical models, in face of these challenges, we suggest several estimation procedures. Our procedures are rooted in an empirical Bayesian approach, i.e., we use and support the basic principles of Bayesian data analysis, but we do it in a way which is still reminiscent of the principle of maximizing likelihood, as well as yielding an estimation procedure. The Bayesian approach requires defining a distribution over the set of parameters, a prior, which, as we demonstrate, can have a tremendous effect on the performance of the resulting probabilistic grammar. We present a novel set of priors for probabilistic grammars, where our goal is to incorporate well-motivated features of natural language into the estimation procedure.

Thesis Statement We claim that probabilistic grammars are a useful modeling tool in unsupervised natural language processing. Vanilla maximum likelihood estimation of probabilistic grammars exposes interesting and challenging computational and learning-theoretic properties of the estimation of probabilistic grammars. However, vanilla maximum likelihood estimation is not sufficient to obtain accurate probabilistic grammars. An empirical Bayesian approach to this estimation problem, both in the parametric setting and in the nonparametric setting, can improve performance considerably.

1.2 Thesis Overview

This thesis includes two main components: a theoretical and methodological presentation and an empirical study. We next describe the organization of these parts.

1.2.1 Theory and Methodology

The first part of this thesis gives a framework for analyzing computational issues with the estimation of probabilistic grammars:

- Chapter 3 covers an analysis of the computational complexity of the estimation of grammars using algorithms such as expectation-maximization (EM), Viterbi EM and conditional Viterbi EM. This chapter is an extension of Cohen and Smith (2010c).
- Chapter 4 covers an analysis of learning-theoretic properties of the estimation of probabilistic grammars. Here, the objective function of expectation-maximization is analyzed in the empirical risk minimization framework with the log-loss. This chapter is an extension of Cohen and Smith (2010b).
- Chapter 5 covers a novel estimation procedure for probabilistic grammars in the empirical Bayesian setting. This procedure uses a prior which is based on the logistic normal prior. This chapter is an extension of Cohen et al. (2008) and Cohen and Smith (2009).
- Chapter 6 covers another estimation technique for probabilistic grammars in the empirical Bayesian setting. This procedure is based on a nonparametric model called adaptor grammars, introduced in Johnson et al. (2006). This chapter is an extension of Cohen et al. (2010).

1.2.2 Empirical Study

The second part of this thesis uses the estimation techniques covered in the first part:

- Chapter 7 describes the main application in this thesis, dependency grammar induction, as well as related work on this application.
- Chapter 8 describes multilingual experiments of applying the techniques from Chapter 5 and Chapter 6 for dependency grammar induction.

We then summarize and discuss future work in Chapter 9.

Chapter 2

Preliminaries

When designing a statistical inference procedure, there are two main approaches that modern statistics offers: frequentist and Bayesian. The frequentist approach is associated with the frequency interpretation of probability, which means that the outcome of an experiment, or a sample, should be viewed as one possible outcome in a sequence of repetitions of the same experiment. With the frequentist approach, the parameters of a model are often treated as fixed and unknown quantities. The end-result of the frequentist method, for example, can often be confidence intervals on these unknown parameters.

The Bayesian approach, on the other hand, offers a subjective interpretation of probability. This approach introduces a *prior*, a distribution over the possible models, which encodes various prior beliefs (possibly subjective) about these models. The introduction of a prior over the space of models makes the use of Bayes' rule readily available to invert the relationship which describes the probability of the data given the model, to a description of the probability of the model given the data:

$$P(\text{model} \mid \text{data}) = \frac{P(\text{data} \mid \text{model})P(\text{model})}{P(\text{data})}$$

Here, $P(\text{data})$ is a normalization constant, which denotes the marginal distribution of the data, and which can be computed as:

$$P(\text{data}) = \sum_{\text{model}} P(\text{data} \mid \text{model})P(\text{model})$$

The question of whether the Bayesian approach or the frequentist approach should serve as the foundation of statistics is subject to debate (Berger, 2010). Any attempt to summarize this debate will not do justice to this complicated issue. However, it is important to note some of the main criticisms that Bayesians and frequentists have about each other's approaches to explain our statistical approach in this thesis.

The main criticism that frequentists have about Bayesians is that they usually give treatment to very basic examples and fail to handle more complicated examples; they focus on priors which are computationally convenient; and that their methods are brittle in settings where there is a disagreement between the prior and the data, because the Bayesian approach depends on very specific priors. Bayesians criticise aspects of the frequentist method as well: they often claim that the frequentist methods do not offer a flexible way of incorporating prior knowledge; and that frequentists lack a principled procedure which can be systematically used for performing statistical inference (Carlin and Louis, 2000).

We believe that criticism from both sides reveals important concerns. In the context of computational linguistics and machine learning, frequentist methods do not permit the incorporation of prior knowledge in an easy or systematic way. There have been some recent attempts in the context of computational linguistics and machine learning to present frameworks which make it possible to encode such prior knowledge (Ganchev, 2010; McCallum et al., 2007). Posterior regularization (Ganchev, 2010) is one such framework, in which the distribution over inferred variables is constrained to fit various pieces of prior knowledge during the inference process. Another example of such a framework is that of the generalized expectation (McCallum et al., 2007), in which preferences about model parameters are incorporated into the objective function which is used for learning. The generalized expectation criterion is quite like the idea of eliciting priors in the Bayesian approach, in which experts help construct a specific prior distribution over the parameters through elicitation.

Yet, most of the frameworks suggested do not offer a solution which can be contained within probability theory, like the Bayesian framework is able to. Indeed, the elegance of the Bayesian approach directly lies in the fact that, from a theoretical point of view, we can systematically apply inference in the presence of prior knowledge by identifying the posterior.

We believe that some criticism towards the Bayesian approach warrants our attention, specifically that it currently relies too heavily on simple priors. If we again consider computational linguistics, where the multinomial family is the main building block of a typical statistical model for language, there is a widespread

use of the Dirichlet prior for computational convenience. This is the result of the Dirichlet prior being *conjugate* to the multinomial family (Section 5.1.1), which enables closed-form solutions for statistical inference in the Bayesian setting.

It is for this reason that we choose to combine these two approaches, that of the frequentists and the Bayesians, in an empirical Bayesian approach (Herbert, 1956; Carlin and Louis, 2000; Berger, 2010). The empirical Bayesian approach assumes that the prior itself is parametrized using *hyperparameters* and our goal is to estimate these parameters from data. We propose a procedure that uses the estimation of these hyperparameters to assist us in finding a point estimate for the model at hand. Bayesians typically manage uncertainty by computing the posterior, which is a distribution over the parameters. After estimating the hyperparameters of the prior, and computing the posterior, we *summarize* this information in a point-estimate such as the posterior mean or mode. This is another divergence from the typical Bayesian approach, which we justify from a practical point of view: a natural language model is usually used in a pipeline of building blocks, and requires a fast inference procedure. Running a full Bayesian procedure for inference on unseen data would lead to significant practical runtime limitations in such a pipeline.

In the rest of this chapter, we detail the two foundational concepts on which this thesis stands. The first is that of *probabilistic grammars*, which is the family of models for which we focus on presenting the estimation procedures, as described in Chapter 1. We then turn to describing the maximum likelihood principle and maximum marginal likelihood estimation in the Bayesian setting.

2.1 Probabilistic Grammars

Probabilistic grammars define a probability distribution over a structured object (a derivation of underlying symbolic grammar) step-by-step as a stochastic process. Hidden Markov models (HMMs), for example, can be understood as a random walk through a probabilistic finite-state network, with an output symbol sampled at each state. Probabilistic context-free grammars (PCFGs) generate phrase-structure trees by recursively rewriting nonterminal symbols as sequences of "child" symbols (each itself a nonterminal symbol or a terminal symbol analogous to an HMM's emissions).

Each step or emission of an HMM and each rewriting operation of a PCFG is conditionally independent of the others given a single structural element (one HMM or PCFG state); this Markov property permits efficient inference over deriva-

tions given a string.

In general, a probabilistic grammar is a pair $\langle \mathbf{G}, \boldsymbol{\theta} \rangle$, where $\mathbf{G}$ is a grammar originating in some grammar formalism, such as a context-free grammar or a linear context-free rewriting system, and $\boldsymbol{\theta}$ is the set of parameters for the probabilistic grammar. The probabilistic grammar $\langle \mathbf{G}, \boldsymbol{\theta} \rangle$ defines the joint probability of a string $\boldsymbol{x}$ and a grammatical derivation $\mathbf{y}$:

$$p(\boldsymbol{x}, \mathbf{y} \mid \boldsymbol{\theta}, \mathbf{G}) = \prod_{k=1}^{K} \prod_{i=1}^{N_k} \theta_{k,i}^{\psi_{k,i}(\boldsymbol{x},\mathbf{y})} = \exp \sum_{k=1}^{K} \sum_{i=1}^{N_k} \psi_{k,i}(\boldsymbol{x}, \mathbf{y}) \log \theta_{k,i} \tag{2.1}$$

where $\psi_{k,i}$ is a function "counting" the frequency of a kth distribution's ith event in a derivation. Here, $\mathbf{G}$ dictates permitted derivations, hence dictating the non-zero probability frequency vectors. The parameters $\boldsymbol{\theta}$ are a group of K multinomials $\langle \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K \rangle$, the kth of which includes N_k competing events. Where $\boldsymbol{\theta}_k = \langle \theta_{k,1}, \ldots, \theta_{k,N_k} \rangle$, each $\theta_{k,i}$ is a probability, such that

$$\forall k, \forall i, \qquad \theta_{k,i} \geq 0 \tag{2.2}$$

$$\forall k, \qquad \sum_{i=1}^{N_k} \theta_{k,i} = 1 \tag{2.3}$$

We denote by $\Theta_{\mathbf{G}}$ this parameter space for $\boldsymbol{\theta}$. We use $\deg(\mathbf{G})$ denote the "degree" of $\mathbf{G}$, i.e., $\deg(\mathbf{G}) = \max_k N_k$. We let $|\boldsymbol{x}|$ denote the length of the string $\boldsymbol{x}$, and $|\mathbf{y}| = \sum_{k=1}^{K} \sum_{i=1}^{N_k} \psi_{k,i}(\mathbf{y})$ denote the "length" (number of event tokens) of the derivation $\mathbf{y}$.

As in many probabilistic models, the variables can be divided various ways. We can consider $\boldsymbol{x}$ and $\mathbf{y}$ correlated structure variables (often $\boldsymbol{x}$ is known if $\mathbf{y}$ is known), or derivation event counts $\mathbf{f}(\boldsymbol{x}, \mathbf{y}) = \langle f_{k,i}(\boldsymbol{x}, \mathbf{y}) \rangle_{1 \leq k \leq K, 1 \leq i \leq N_k}$ as an integer-vector random variable.

Note that there may be many derivations $\mathbf{y}$ for a given string $\boldsymbol{x}$—perhaps even infinitely many in some kinds of grammars. For HMMs, there are three kinds of multinomials: a starting state multinomial, a transition multinomial per state and an emission multinomial per state. With HMMs, $K = 2s + 1$, where s is the number of states. The value of N_k depends on whether the kth multinomial is the starting state multinomial (in which case $N_k = s$), transition multinomial ($N_k = s$) or emission multinomial ($N_k = t$, with t being the number of symbols in the HMM). For PCFGs, each multinomial among the K multinomials correspond

to a set of N_k context-free rules headed by the same nonterminal. $\theta_{k,i}$ is then the probability of the ith rule for the kth nonterminal.

Probabilistic grammars are an expressive family of models, that can represent models which do not seem, at first glance, like grammatical models.

Example 2.1 *Class-based unigram model (Brown et al., 1990) – Let the observed symbols in $\boldsymbol{x}$ range over words in some language's vocabulary $\mathcal{W}$. Let each word token x_i have an associated word class from a finite set Λ, denoted y_i; the y_i are all hidden. The derivation in this model is the sequence $\langle y_1, \ldots, y_n \rangle$. The probabilistic model consists of two parts:*

1. *For all $y \in \Lambda \cup \{\text{stop}\}$, $\theta_{\text{c}}(\mathbf{y})$ is the probability that the next word will be generated by class y. $\theta_{\text{c}}(\text{stop})$ is the stopping probability.*

2. *For all $y \in \Lambda$ and all $x \in \mathcal{W}$, $\theta_{\text{w}}(x \mid y)$ is the conditional probability that class y will generate word x.*

In this simple model, $K = 1 + |\Lambda|$, $N_1 = |\Lambda|$, and for $k > 1$, $N_k = |\mathcal{W}|$. This model can be thought of as an hidden Markov model with zero order, i.e., it has no dependencies between the different hidden states. In addition, if we place a Dirichlet prior on the grammar parameters $\boldsymbol{\theta}$ and sample them once per document, this model becomes equivalent to the latent Dirichlet allocation model (Blei et al., 2003). In this case, the derivation vector $\mathbf{y}$ corresponds to a set of topics selected for each word in the bag of words representing the document. This model (latent Dirichlet allocation) can also be formulated as a context-free grammar. See Johnson (2010) for details.

We use the following notation for $\mathbf{G}$:

- $L(\mathbf{G})$ is the set of all strings (sentences) $\boldsymbol{x}$ that can be generated using the grammar $\mathbf{G}$ (the "language of $\mathbf{G}$").

- $D(\mathbf{G})$ is the set of all possible derivations $\mathbf{y}$ that can be generated using the grammar $\mathbf{G}$.

- $D_{\boldsymbol{x}}(\mathbf{G})$ is the set of all possible derivations $\mathbf{y}$ that can be generated using the grammar $\mathbf{G}$ and have the yield $\boldsymbol{x}$.

We turn now to a more detailed explanation and definition of probabilistic context-free grammars, which are of special interest in this disseration. A PCFG $\langle \mathbf{G}, \boldsymbol{\theta} \rangle$ consists of:

- A finite set of nonterminal symbols $\mathcal{N} = \mathcal{N}(\mathbf{G})$;
- A finite set of terminal symbols $\mathcal{W}$;
- For each $A \in \mathcal{N}$, a set of rewrite rules $\boldsymbol{R}_A$ of the form $A \to \alpha$, where $\alpha \in (\mathcal{N} \cup \mathcal{W})^*$, and $\mathcal{R} = \mathcal{R}(\mathbf{G}) = \cup_{A \in \mathcal{N}} \boldsymbol{R}_A$;
- For each rule $A \to \alpha$, a probability $\theta_{A \to \alpha}$. The collection of probabilities is denoted $\boldsymbol{\theta}$, and they are constrained such that:

$$\begin{aligned} \forall (A \to \alpha) \in \boldsymbol{R}_A, \quad \theta_{A \to \alpha} &\geq 0 \\ \forall A \in \mathcal{N}, \quad \sum_{\alpha : (A \to \alpha) \in \boldsymbol{R}_A} \theta_{A \to \alpha} &= 1 \end{aligned}$$

  That is, $\boldsymbol{\theta}$ is grouped into $|\mathcal{N}|$ multinomial distributions.

Under the PCFG, the joint probability of a string $x \in \mathcal{W}^*$ and a grammatical derivation y is[1]

$$p(\boldsymbol{x}, \mathbf{y} \mid \boldsymbol{\theta}, \mathbf{G}) = \prod_{(A \to \alpha) \in \mathcal{R}} (\theta_{A \to \alpha})^{\psi_{A \to \alpha}(\mathbf{y})} = \exp \sum_{(A \to \alpha) \in \mathcal{R}} \psi_{A \to \alpha}(\mathbf{y}) \log \theta_{A \to \alpha}$$

where $\psi_{A \to \alpha}(\mathbf{y})$ is a function that "counts" the number of times the rule $A \to \alpha$ appears in the derivation $\mathbf{y}$. $\psi_A(\mathbf{y})$ will similarly denote the number of times that nonterminal A appears in y. Given a sample of derivations $\mathbf{y} = \langle \mathbf{y}_1, \ldots, \mathbf{y}_n \rangle$, we denote:

$$\begin{aligned} F_{A \to \alpha}(\mathbf{y}) &= \sum_{i=1}^{n} \psi_{A \to \alpha}(\mathbf{y}_i) \\ F_A(\mathbf{y}) &= \sum_{i=1}^{n} \psi_A(\mathbf{y}_i). \end{aligned}$$

2.2 Maximum Likelihood Principle

Roughly speaking, the maximum likelihood principle states that in order to identify the parameters of the model, we need to maximize the likelihood function of

[1] Note that $x = \text{yield}(\mathbf{y})$; if the derivation is known, the string is also known. On the other hand, there may be many derivations with the same yield, perhaps even infinitely many.

the data with respect to the parameters. In the case of probabilistic grammars, the data can be complete (which means that the data includes sentences and grammatical derivations for these sentences), or it can be incomplete (which means that the data includes sentences only). Parameter estimation in the latter case is also referred to as "unsupervised learning."

In the case of complete data, the MLE principle can be formalized as follows. Given $(\boldsymbol{x}_1, \mathbf{y}_1), \ldots, (\boldsymbol{x}_n, \mathbf{y}_n)$, where x_i represent strings and $\mathbf{y}_i$ represent derivations for these strings under some grammar $\mathbf{G}$, we are interested in identifying parameters $\boldsymbol{\theta}^*$:

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \frac{1}{n} \sum_{i=1}^{n} \log p(\boldsymbol{x}_i, \mathbf{y}_i \mid \boldsymbol{\theta})$$

In the case of incomplete data, we can no longer include $\mathbf{y}_i$ in the specification of the log-likelihood, as only $\boldsymbol{x}_i$ are available. Instead of maximizing the complete likelihood in this case, we focus on identifying the $\boldsymbol{\theta}^*$ that maximize the *marginal* log-likelihood:

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \frac{1}{n} \sum_{i=1}^{n} \log \sum_{\mathbf{y}} p(\boldsymbol{x}_i, \mathbf{y} \mid \boldsymbol{\theta})$$

There are a few problems with the latter setting. Specifically:

- The objective function of the marginal log-likelihood, for which we are trying to identify a maximizer, has many local maxima in many cases. As a consequence, it is not trivial to find the global maximizer, unlike the supervised case, in which there is a closed-form solution for the log-likelihood's maximizer. We address this issue in Chapter 3, and describe some hardness results for maximization of the marginal log-likelihood as well as closely related objectives.

- The marginalized distribution over strings, with the derivations being marginalized out, is *non-identifiable*. This means that there can be sets of parameters, all different from each other, that lead to the same marginalized distribution over strings. We address this issue in Chapter 4.

The algorithm that is most commonly used to maximize the objective function in Equation 2.2 is the expectation-maximization (EM) algorithm (Dempster et al.,

1977). The EM algorithm iterates between two steps, updating at iteration t a set of parameters, $\boldsymbol{\theta}_t$:

- E-step. Compute the function

$$Q(\boldsymbol{\theta}) = \sum_{\mathbf{y}} p(\mathbf{y} \mid \boldsymbol{x}, \boldsymbol{\theta}_{t-1}) \log p(\boldsymbol{x}, \mathbf{y} \mid \boldsymbol{\theta}).$$

- M-step. Update

$$\boldsymbol{\theta}_t = \arg\max_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}).$$

EM and its variants have been used in many cases for the estimation of probabilistic grammars (Pereira and Schabes, 1992; Carroll and Charniak, 1992; Chen, 1995; Klein and Manning, 2002, 2004). In our empirical study (Chapter 8), we show that using EM for the estimation of probabilistic grammars is not sufficient to obtain good performance. We suggest an alternative, in the form of a variational EM algorithm, which sets a Bayesian prior on the grammar parameters.

Maximum Marginal Likelihood Estimation As mentioned above, for a large part of this thesis, we choose to work in an empirical Bayesian setting. Bayesian analysis depends on a prior to obtain model parameters, $p(\boldsymbol{\theta})$. This means that we now have a joint model over both parameters and variables in the model:

$$p(\boldsymbol{\theta}, \boldsymbol{x}, \mathbf{y}) = p(\boldsymbol{\theta}) p(\mathbf{y} \mid \boldsymbol{\theta}) p(\boldsymbol{x} \mid \boldsymbol{\theta}, \boldsymbol{x}).$$

The prior in this thesis will either be parametric (Chapter 5) or nonparametric (Chapter 6). Priors (both parametric and nonparametric) often depend on unknown hyperparameters $\boldsymbol{\alpha}$. We can have a hyperprior over $\boldsymbol{\alpha}$ as well, but we instead choose to stop the hierarchy at the level of the hyperparameters, and *estimate* these hyperparameters (as we discuss earlier in this chapter). Given a set of hyperparameters $\boldsymbol{\alpha}$ which parametrize $p(\boldsymbol{\theta} \mid \boldsymbol{\alpha})$, we show that the marginal distribution of the data, when observing only strings (i.e. in the unsupervised setting), is:

$$p(\boldsymbol{x} \mid \boldsymbol{\alpha}) = \int_{\boldsymbol{\theta}} \sum_{\mathbf{y}} p(\boldsymbol{x}, \mathbf{y} \mid \boldsymbol{\theta}) p(\boldsymbol{\theta} \mid \boldsymbol{\alpha}) d\boldsymbol{\theta} \tag{2.4}$$

Empirical Bayesian estimation uses the marginal distribution in Equation 2.4 to estimate $\hat{\boldsymbol{\alpha}}$, and then sets the prior to be $p(\boldsymbol{\theta} \mid \hat{\boldsymbol{\alpha}})$. A common approach for estimating $\hat{\boldsymbol{\alpha}}$ is to again use the maximum likelihood principle. Indeed, this kind of estimation is also known as "maximum marginal likelihood estimation."

We see in Chapters 5 and 6 that it is not trivial to maximize the marginal likelihood in our case. To overcome these difficulties, we use variational inference within a variational EM algorithm (Wainwright and Jordan, 2008).

Part I

Theory and Methodology

Chapter 3

Computational Complexity of Estimation of Probabilistic Grammars

We presented in Section 2.2 the maximum likelihood principle and its application to the estimation of probabilistic grammars in the unsupervised setting. In Section 2.2 we reviewed the expectation-maximization algorithm, which attempts to maximize the marginalized likelihood of a given set of strings and a probabilistic grammar. The EM algorithm is a coordinate ascent algorithm, and therefore iterative in nature. There are other iterative algorithms, variants to the EM algorithm, most notably the Viterbi EM algorithm (or "hard" EM), on which we focus in the first part of this chapter. The goal of this chapter is to describe computational complexity results for Viterbi EM and other algorithms from the same family, including the EM algorithm described in Section 2.2.

For the most part, we restrict our discussion in this chapter to probabilistic context-free grammars. We argue that most grammar formalisms in computational linguistics which are used for modeling natural language syntax include at least the expressivity which exists in context-free grammars, and therefore the results we present in this chapter, which are hardness results, are generally applicable to any type of probabilistic grammar.

The main results described in this chapter are:

- NP-hardness of optimizing the objective function of Viterbi EM, conditional Viterbi EM and EM (when the grammar is not fixed, i.e., when it is given as an *input* to the learner).

- NP-hardness of finding an approximate solution to the maximizer of the objective function of Viterbi EM (when the grammar is not fixed).
- #P-hardness result of counting the number of local maxima the objective function of Viterbi EM has.
- A polynomial time algorithm for maximizing the objective function of Viterbi EM when the grammar is fixed.

Some of the work in this chapter has been described in Cohen and Smith (2010c).

3.1 Viterbi Training

Viterbi EM is an unsupervised learning algorithm, used in NLP in various settings (Choi and Cardie, 2007; Wang et al., 2007; Goldwater and Johnson, 2005; DeNero and Klein, 2008; Spitkovsky et al., 2010b). In the context of PCFGs, it aims to select parameters $\boldsymbol{\theta}$ and phrase-structure trees $\mathbf{y}$ jointly. It does so by iteratively updating a state consisting of $(\boldsymbol{\theta}, \mathbf{y})$. The state is initialized with some value, then the algorithm alternates between (i) a "hard" E-step, where the strings $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ are parsed according to a current, fixed $\boldsymbol{\theta}$, giving new values for $\mathbf{y}$, and (ii) an M-step, where the $\boldsymbol{\theta}$ are selected to maximize likelihood, with $\mathbf{y}$ fixed.

With PCFGs, the E-step requires running an algorithm such as (probabilistic) CKY or Earley's algorithm, while the M-step normalizes frequency counts $F_{A\rightarrow\alpha}(\mathbf{y})$ to obtain the maximum likelihood estimate's closed-form solution.

We can understand Viterbi EM as a coordinate ascent procedure that approximates the solution to the following declarative problem:

Problem 3.1 ViterbiTrain
Input: $\mathbf{G}$ *context-free grammar,* $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ *training instances from* $L(\mathbf{G})$
Output: $\boldsymbol{\theta}$ *and* $\mathbf{y}_1, \ldots, \mathbf{y}_n$ *such that*

$$(\boldsymbol{\theta}, \mathbf{y}_1, \ldots, \mathbf{y}_n) = \operatorname*{argmax}_{\boldsymbol{\theta}, \mathbf{y}} \prod_{i=1}^{n} p(\boldsymbol{x}_i, \mathbf{y}_i \mid \boldsymbol{\theta}) \tag{3.1}$$

The optimization problem in Equation 3.1 is non-convex (with potentially many local optima) and, as we will show in Section 3.2, hard to optimize. Therefore it is necessary to resort to approximate algorithms like Viterbi EM.

Neal and Hinton (1998) use the term "sparse EM" to refer to a version of the EM algorithm where the E-step finds the modes of hidden variables (rather than marginals as in standard EM). Viterbi EM is an example of this, where the E-step finds the mode for each $\boldsymbol{x}_i$'s derivation, $\operatorname{argmax}_{\mathbf{y} \in D_{\boldsymbol{x}_i}(\mathbf{G})} p(\boldsymbol{x}_i, \mathbf{y} \mid \boldsymbol{\theta})$.

We will refer to

$$\mathcal{L}(\boldsymbol{\theta}, \mathbf{y}) \quad = \quad \prod_{i=1}^{n} p(\boldsymbol{x}_i, \mathbf{y}_i \mid \boldsymbol{\theta})$$

as "the objective function of ViterbiTrain," or "the Viterbi likelihood" for short.

Viterbi training and Viterbi EM are closely related to **self-training**, an important concept in semi-supervised NLP (Charniak, 1997; McClosky et al., 2006a,b). With self-training, the model is learned with some seed annotated data, and then iterates by labeling new, unannotated data and adding it to the original annotated training set. McClosky et al. consider self-training to be "one round of Viterbi EM" with supervised initialization using labeled seed data. We refer the reader to Abney (2007) for more details.

Viterbi training is attractive because it essentially requires only a decoding algorithm to run. Viterbi EM for PCFGs, for example, requires parsing a corpus during the E-step. While computing feature expectations, which would be required in EM, can be done using a relatively small choice for algorithms, there is a larger selection of decoding algorithms, all of them can be used with Viterbi training. These algorithms can also be specialized to run much faster using heuristics such as A^*-search (Klein and Manning, 2003a).

3.2 Hardness of Viterbi Training

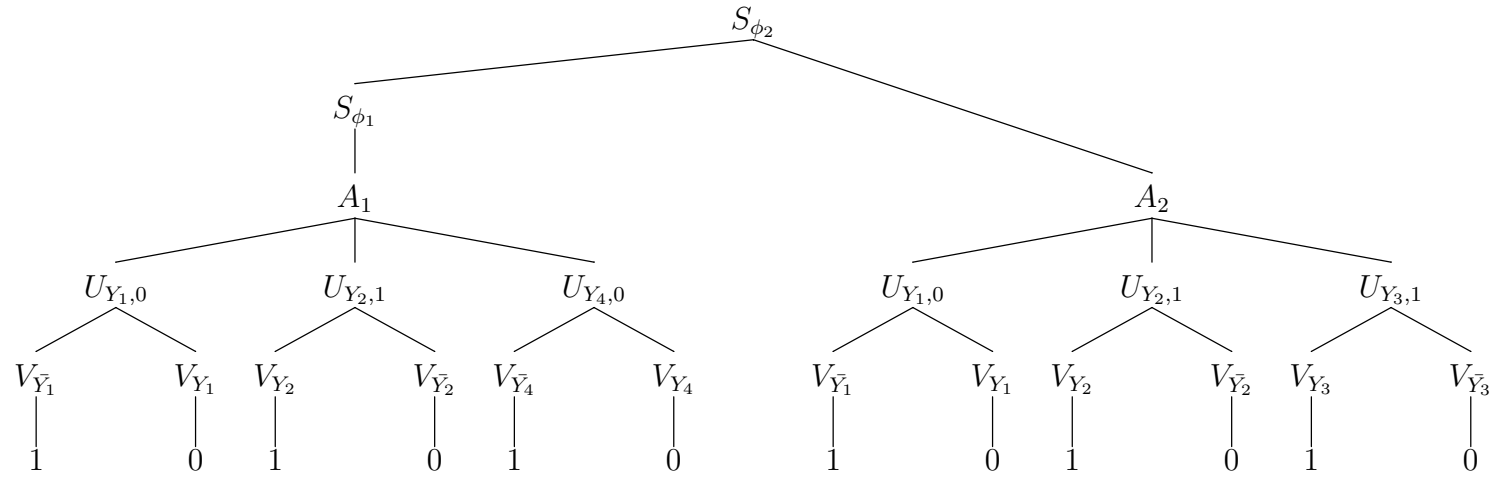


Figure 3.1: An example of a Viterbi parse tree which represents a satisfying assignment for $\phi = (Y_1 \vee Y_2 \vee \bar{Y_4}) \wedge (\bar{Y_1} \vee \bar{Y_2} \vee Y_3)$. In $\boldsymbol{\theta}_\phi$, all rules appearing in the parse tree have probability 1. The extracted assignment would be $Y_1 = 0, Y_2 = 1, Y_3 = 1, Y_4 = 0$. Note that there is no usage of two different rules for a single nonterminal.

We now describe hardness results for Problem 3.1. We first note that the following problem is known to be NP-hard, and in fact, NP-complete (Sipser, 2006):

Problem 3.2 3-SAT
Input: *A formula* $\phi = \bigwedge_{i=1}^{m} (a_i \vee b_i \vee c_i)$ *in conjunctive normal form, such that each clause has* 3 *literals.*
Output: *1 if there is a satisfying assignment for* ϕ *and 0 otherwise.*

We now describe a reduction of 3-SAT to Problem 3.1. Given an instance of the 3-SAT problem, the reduction will, in polynomial time, create a grammar and a single string such that solving the ViterbiTrain problem for this grammar and string will yield a solution for the instance of the 3-SAT problem.

Let $\phi = \bigwedge_{i=1}^{m} (a_i \vee b_i \vee c_i)$ be an instance of the 3-SAT problem, where a_i, b_i and c_i are literals over the set of variables $\{Y_1, \ldots, Y_N\}$ (a literal refers to a variable Y_j or its negation, $\bar{Y_j}$). Let C_j be the jth clause in ϕ, such that $C_j = a_j \vee b_j \vee c_j$. We define the following context-free grammar $\mathbf{G}_\phi$ and string to parse s_ϕ:

3. The terminals of $\mathbf{G}_\phi$ are the binary digits $\Sigma = \{0, 1\}$.
4. We create N nonterminals V_{Y_r}, $r \in \{1, \ldots, N\}$ and rules $V_{Y_r} \to 0$ and $V_{Y_r} \to 1$.
5. We create N nonterminals $V_{\bar{Y_r}}$, $r \in \{1, \ldots, N\}$ and rules $V_{\bar{Y_r}} \to 0$ and $V_{\bar{Y_r}} \to 1$.
6. We create $U_{Y_r,1} \to V_{Y_r} V_{\bar{Y_r}}$ and $U_{Y_r,0} \to V_{\bar{Y_r}} V_{Y_r}$.
7. We create the rule $S_{\phi_1} \to A_1$. For each $j \in \{2, \ldots, m\}$, we create a rule $S_{\phi_j} \to S_{\phi_{j-1}} A_j$ where S_{ϕ_j} is a new nonterminal indexed by $\phi_j \triangleq \bigwedge_{i=1}^{j} C_i$ and A_j is also a new nonterminal indexed by $j \in \{1, \ldots, m\}$.
8. Let $C_j = a_j \vee b_j \vee c_j$ be clause j in ϕ. Let $Y(a_j)$ be the variable that a_j mentions. Let (y_1, y_2, y_3) be a satisfying assignment for C_j where $y_k \in \{0, 1\}$ and is the value of $Y(a_j)$, $Y(b_j)$ and $Y(c_j)$ respectively for $k \in \{1, 2, 3\}$. For each such clause-satisfying assignment, we add the rule:

$$A_j \to U_{Y(a_j),y_1} U_{Y(b_j),y_2} U_{Y(c_j),y_3}$$

   For each A_j, we would have at most 7 rules of that form, since one rule will be logically inconsistent with $a_j \vee b_j \vee c_j$.
9. The grammar's start symbol is S_{ϕ_n}.

10. The string to parse is $s_\phi = (10)^{3m}$, i.e. $3m$ consecutive occurrences of the string 10.

A parse of the string s_ϕ using $\mathbf{G}_\phi$ will be used to get an assignment by setting $Y_r = 0$ if the rule $V_{Y_r} \to 0$ or $V_{\bar{Y_r}} \to 1$ are used in the derivation of the parse tree, and 1 otherwise. Notice that at this point we do not exclude "contradictions" coming from the parse tree, such as $V_{Y_3} \to 0$ used in the tree together with $V_{Y_3} \to 1$ or $V_{\bar{Y_3}} \to 0$. The following lemma gives a condition under which the assignment is consistent (so contradictions do not occur in the parse tree):

Lemma 3.1 *Let ϕ be an instance of the* 3-SAT *problem, and let $\mathbf{G}_\phi$ be a probabilistic CFG based on the above grammar with weights $\boldsymbol{\theta}_\phi$. If the (multiplicative) weight of the Viterbi parse of s_ϕ is 1, then the assignment extracted from the parse tree is consistent.*

Proof Since the probability of the Viterbi parse is 1, all rules of the form $\{V_{Y_r}, V_{\bar{Y_r}}\} \to \{0, 1\}$ which appear in the parse tree have probability 1 as well. There are two possible types of inconsistencies. We show that neither exists in the Viterbi parse:

1. For any r, an appearance of both rules of the form $V_{Y_r} \to 0$ and $V_{Y_r} \to 1$ cannot occur because all rules that appear in the Viterbi parse tree have probability 1.
2. For any r, an appearance of rules of the form $V_{Y_r} \to 1$ and $V_{\bar{Y_r}} \to 1$ cannot occur, because whenever we have an appearance of the rule $V_{Y_r} \to 0$, we have an adjacent appearance of the rule $V_{\bar{Y_r}} \to 1$ (because we parse substrings of the form 10), and then again we use the fact that all rules in the parse tree have probability 1. The case of $V_{Y_r} \to 0$ and $V_{\bar{Y_r}} \to 0$ is handled analogously.

Thus, both possible inconsistencies are ruled out, resulting in a consistent assignment. □

Figure 3.1 gives an example of an application of the reduction.

Lemma 3.2 *Define ϕ, $\mathbf{G}_\phi$ as before. There exists $\boldsymbol{\theta}_\phi$ such that the Viterbi parse of s_ϕ is 1 if and only if ϕ is satisfiable. Moreover, the satisfying assignment is the one extracted from the parse tree with weight 1 of s_ϕ under $\boldsymbol{\theta}_\phi$.*

Proof ($\Longrightarrow$) Assume that there is a satisfying assignment. Each clause $C_j = a_j \vee b_j \vee c_j$ is satisfied using a tuple (y_1, y_2, y_3) which assigns value for $Y(a_j)$, $Y(b_j)$ and $Y(c_j)$. This assignment corresponds the following rule

$$A_j \to U_{Y(a_j),y_1} U_{Y(b_j),y_2} U_{Y(c_j),y_3}$$

Set its probability to 1, and set all other rules of A_j to 0. In addition, for each r, if $Y_r = y$, set the probabilities of the rules $V_{Y_r} \to y$ and $V_{\bar{Y_r}} \to 1 - y$ to 1 and $V_{\bar{Y_r}} \to y$ and $V_{Y_r} \to 1 - y$ to 0. The rest of the weights for $S_{\phi_j} \to S_{\phi_{j-1}} A_j$ are set to 1. This assignment of rule probabilities results in a Viterbi parse of weight 1.

($\Longleftarrow$) Assume that the Viterbi parse has probability 1. From Lemma 3.1, we know that we can extract a consistent assignment from the Viterbi parse. In addition, for each clause C_j we have a rule

$$A_j \to U_{Y(a_j),y_1} U_{Y(b_j),y_2} U_{Y(c_j),y_3}$$

that is assigned probability 1, for some (y_1, y_2, y_3). One can verify that (y_1, y_2, y_3) are the values of the assignment for the corresponding variables in clause C_j, and that they satisfy this clause. This means that each clause is satisfied by the assignment we extracted. □

In order to show an NP-hardness result, we need to "convert" ViterbiTrain to a decision problem. The natural way to do it, following Lemmas 3.1 and 3.2, is to state the decision problem for ViterbiTrain as "given $\mathbf{G}$ and $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ and $\alpha \geq 0$, is the optimized value of the objective function $\mathcal{L}(\boldsymbol{\theta}, \mathbf{y}) \geq \alpha$?" and use $\alpha = 1$ together with Lemmas 3.1 and 3.2. (Naturally, an algorithm for solving ViterbiTrain can easily be used to solve its decision problem.) Following this argument, we conclude with the main theorem statement for this section:

Theorem 3.3 *The decision version of the* ViterbiTrain *problem is NP-hard.*

3.3 Hardness of Approximation

A natural path of exploration following the hardness result we showed is determining whether an *approximation* of ViterbiTrain is also hard. Perhaps there is an efficient approximation algorithm for ViterbiTrain we could use instead of coordinate ascent algorithms such as Viterbi EM. Recall that such algorithms' main guarantee is identifying a local maximum; we know nothing about how far it will be from the global maximum.

We next show that approximating the objective function of ViterbiTrain with a constant factor of ρ is hard for any $\rho \in (\frac{1}{2}, 1]$ (i.e., $1/2 + \epsilon$ approximation is hard for any $\epsilon \leq 1/2$). This means that, under the P $\neq$ NP assumption, there is no efficient algorithm that, given a grammar $\mathbf{G}$ and a sample of sentences $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$,

returns $\boldsymbol{\theta}'$ and $\mathbf{y}'$ such that:

$$\mathcal{L}(\boldsymbol{\theta}', \mathbf{y}') \quad \geq \quad \rho \cdot \max_{\boldsymbol{\theta}, \mathbf{y}} \prod_{i=1}^{n} p(\boldsymbol{x}_i, \mathbf{y}_i \mid \boldsymbol{\theta})$$

We will continue to use the same reduction from Section 3.2. Let s_ϕ be the string from that reduction, and let $(\boldsymbol{\theta}, \mathbf{y})$ be the optimal solution for ViterbiTrain given $\mathbf{G}_\phi$ and s_ϕ. We first note that if $p(s_\phi, y \mid \boldsymbol{\theta}) < 1$ (implying that there is no satisfying assignment), then there must be a nonterminal which appears along with two different rules in $\mathbf{y}$.

This means that we have a nonterminal $B \in \mathcal{N}$ with some rule $B \to \alpha$ that appears k times, while the nonterminal appears in the parse $r \geq k+1$ times. Given the tree $\mathbf{y}$, the $\boldsymbol{\theta}$ that maximizes the objective function is the maximum likelihood estimate (MLE) for $\mathbf{y}$ (counting and normalizing the rules).[1] We therefore know that the ViterbiTrain objective function, $\mathcal{L}(\boldsymbol{\theta}, y)$, is at most $\left(\frac{k}{r}\right)^k$, because it includes a factor equal to $\left(\frac{\psi_{B\to\alpha}(y)}{\psi_B(y)}\right)^{\psi_{B\to\alpha}(y)}$, where $\psi_B(y)$ is the number of times nonterminal B appears in $\mathbf{y}$ (hence $\psi_B(y) = r$) and $\psi_{B\to\alpha}(y)$ is the number of times $B \to \alpha$ appears in $\mathbf{y}$ (hence $\psi_{B\to\alpha}(y) = k$). For any $k \geq 1$, $r \geq k+1$:

$$\left(\frac{k}{r}\right)^k \leq \left(\frac{k}{k+1}\right)^k \leq \frac{1}{2} \tag{3.4}$$

This means that if the value of the objective function of ViterbiTrain is not 1 using the reduction from Section 3.2, then it is at most $\frac{1}{2}$. If we had an efficient approximate algorithm with approximation coefficient $\rho > \frac{1}{2}$ (Equation 3.3 holds), then in order to solve 3-SAT for formula ϕ, we could run the algorithm on $\mathbf{G}_\phi$ and s_ϕ and check whether the assignment to $(\boldsymbol{\theta}, \mathbf{y})$ that the algorithm returns satisfies ϕ or not, and return our response accordingly.

If ϕ were satisfiable, then the true maximal value of $\mathcal{L}$ would be 1, and the approximation algorithm would return $(\boldsymbol{\theta}, \mathbf{y})$ such that $\mathcal{L}(\boldsymbol{\theta}, \mathbf{y}) \geq \rho > \frac{1}{2}$. $\mathbf{y}$ would have to correspond to a satisfying assignment, and in fact $p(\mathbf{y} \mid \boldsymbol{\theta}) = 1$, because in any other case, the probability of a derivation which does not represent a satisfying assignment is smaller than $\frac{1}{2}$. If ϕ were not satisfiable, then the

[1] Note that we can only make $p(\mathbf{y} \mid \boldsymbol{\theta}, x)$ greater by using $\boldsymbol{\theta}$ to be the MLE for the derivation $\mathbf{y}$.

approximation algorithm would never return a $(\boldsymbol{\theta}, \mathbf{y})$ that results in a satisfying assignment (because such a $(\boldsymbol{\theta}, \mathbf{y})$ does not exist).

The conclusion is that an efficient algorithm for approximating the objective function of ViterbiTrain (Equation 3.1) within a factor of $\frac{1}{2} + \epsilon$ is unlikely to exist. If there were such an algorithm, we could use it to solve 3-SAT using the reduction from Section 3.2.

We note that the hardness of approximation result can be improved to a constant of $1/4+\epsilon$ instead of a constant of $1/2+\epsilon$. The reason for that is that whenever we have a rule, for nonterminal A, firing in derivation less than the total number of times that nonterminal A appears, there must be at least *two* rules like that. Each of the rules, having counts k_1 and k_2, will contribute to the likelihood a value of $\left(\frac{k_i}{r}\right)^{k_i} \leq 1/4$, which means that the total likelihood we obtain will be smaller or equal to $1/4$. (See Equation 3.4.) Then we can follow the same line of argumentation we followed in this section to tighten the hardness of approximation factor from $1/2$ to $1/4$, showing that the problem of approximation is harder.

3.4 Extensions of the Hardness Result

We include in this section several extensions to the hardness result of ViterbiTrain.

3.4.1 Hardness Results for Other Objectives

An alternative problem to Problem 3.1, a variant of Viterbi training, is the following (see, for example, Klein and Manning, 2001):

Problem 3.3 ConditionalViterbiTrain
Input: $\mathbf{G}$ *context-free grammar,* $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ *training instances from* $L(\mathbf{G})$
Output: $\boldsymbol{\theta}$ *and* $\mathbf{y}_1, \ldots, \mathbf{y}_n$ *such that*

$$(\boldsymbol{\theta}, \mathbf{y}_1, \ldots, \mathbf{y}_n) = \operatorname*{argmax}_{\boldsymbol{\theta}, \mathbf{y}} \prod_{i=1}^{n} p(\mathbf{y}_i \mid \boldsymbol{\theta}, \boldsymbol{x}_i)$$

Here, instead of maximizing the likelihood, we maximize the *conditional* likelihood. Note that there is a hidden assumption in this problem definition, that $\boldsymbol{x}_i$ can be parsed using the grammar $\mathbf{G}$. Otherwise, the quantity $p(\mathbf{y}_i \mid \boldsymbol{\theta}, \boldsymbol{x}_i)$ is not well-defined. We can extend ConditionalViterbiTrain to return $\perp$ in the case of

not having a parse for one of the $\boldsymbol{x}_i$—this can be efficiently checked using a run of a cubic-time parser on each of the strings $\boldsymbol{x}_i$ with the grammar $\mathbf{G}$.

An approximate technique for this problem is similar to Viterbi EM, only modifying the M-step to maximize the conditional, rather than joint, likelihood. This new M-step will not have a closed form and may require auxiliary optimization techniques like gradient ascent.

Our hardness result for ViterbiTrain applies to ConditionalViterbiTrain as well. The reason is that if $p(\mathbf{y}, s_\phi \mid \boldsymbol{\theta}_\phi) = 1$ for a ϕ with a satisfying assignment, then $L(\mathbf{G}) = \{s_\phi\}$ and $D(\mathbf{G}) = \{y\}$. This implies that $p(\mathbf{y} \mid \boldsymbol{\theta}_\phi, s_\phi) = 1$. If ϕ is unsatisfiable, then for the optimal $\boldsymbol{\theta}$ of ViterbiTrain we have $\mathbf{y}$ and $\mathbf{y}'$ such that $0 < p(\mathbf{y}, s_\phi \mid \boldsymbol{\theta}_\phi) < 1$ and $0 < p(\mathbf{y}', s_\phi \mid \boldsymbol{\theta}_\phi) < 1$, and therefore $p(\mathbf{y} \mid \boldsymbol{\theta}_\phi, s_\phi) < 1$, which means the conditional objective function will not obtain the value 1. (Note that there always exist some parameters $\boldsymbol{\theta}_\phi$ that generate s_ϕ.) So, again, given an algorithm for ConditionalViterbiTrain, we can discern between a satisfiable formula and an unsatisfiable formula, using the reduction from Section 3.2 with the given algorithm, and identify whether the value of the objective function is 1 or strictly less than 1. We get the result that:

Theorem 3.4 *The decision problem of* ConditionalViterbiTrain *problem is NP-hard.*

where the decision problem of ConditionalViterbiTrain is defined analogously to the decision problem of ViterbiTrain.

We can similarly show that finding the global maximum of the marginalized likelihood:

$$\max_{\boldsymbol{\theta}} \frac{1}{n} \sum_{i=1}^{n} \log \sum_{y} p(\boldsymbol{x}_i, \mathbf{y} \mid \boldsymbol{\theta}) \tag{3.5}$$

is NP-hard. The reasoning follows. Using the reduction from before, if ϕ is satisfiable, then Equation 3.5 gets value 0. If ϕ is unsatisfiable, then we would still get value 0 only if $L(\mathbf{G}) = \{s_\phi\}$. If $\mathbf{G}_\phi$ generates a single derivation for $(10)^{3m}$, then we actually do have a satisfying assignment from Lemma 3.1. Otherwise (more than a single derivation), the optimal $\boldsymbol{\theta}$ would have to give fractional probabilities to rules of the form $V_{Y_r} \to \{0, 1\}$ (or $V_{\bar{Y}_r} \to \{0, 1\}$). In that case, it is no longer true that $(10)^{3m}$ is the only generated sentence, which is a contradiction.

The quantity in Equation 3.5 can be maximized approximately using algorithms like EM, so this gives a hardness result for optimizing the objective function of EM for PCFGs. Day (1983) previously showed that maximizing the marginalized likelihood for hidden Markov models is NP-hard.

We note that the grammar we use for all of our results is not recursive. Therefore, we can encode this grammar as a hidden Markov model, strengthening our result from PCFGs to HMMs.

3.4.2 #P-hardness of Viterbi Training

We conclude the extensions to the hardness result with a note about the #P-hardness of the counting problem of Viterbi training. The counting problem we consider is:

Problem 3.4 ViterbiTrainCount
Input: $\mathbf{G}$ *context-free grammar,* $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ *training instances from* $L(\mathbf{G})$*i,* $\alpha \in [0, 1]$
Output: *The count* m *of* $\boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_m$ *and* $\mathbf{y}_{j1}, \ldots, \mathbf{y}_{jn}$ *for* $j \leq m$*, such that for every* $j \leq m$ *we have:*

$$\prod_{i=1}^{n} p(\boldsymbol{x}_i, \mathbf{y}_{ji} \mid \boldsymbol{\theta}_j) \leq \alpha$$

It is known that the problem of counting the number of assignments that satisfy a 3-SAT formula is #P-complete (in fact, this is true even for a 2-SAT formula). If we had an algorithm for solving ViterbiTrainCount, we could have done the following to count the number of assignments satisfying a 3-SAT formula:

- Use the reduction described in Section 3.2 to convert a 3-SAT formula ϕ to a grammar $\mathbf{G}_\phi$ and a string $\boldsymbol{s}$.
- Return the result of ViterbiTrainCount on these inputs ($\mathbf{G}$ and $\boldsymbol{s}_\phi$), together with $\alpha = 1$.

Indeed, this algorithm is correct, because of Lemma 3.2: the 3-SAT formula is satisfied using an assignment if and only if the corresponding assignment parse tree is realizable with probability 1 using $\mathbf{G}$. The conclusion is that ViterbiTrainCount is #P-hard.

3.5 Polynomial Time Algorithm for Solving Viterbi Training

We now show that the complexity of the Viterbi training algorithm comes from the arbitrariness in the selection of a grammar as an input. More specifically, we show that if the grammar is fixed (i.e. not given as part of the input), then there is a polynomial time algorithm that solves Viterbi training, where polynomial here is with respect to the total length of the corpus being input to the Viterbi training problem. We note that our algorithm cannot be used in practice. Even though it is a polynomial time algorithm, the exponent depends on the size of the grammar. There are other cases in which related problems, such as parsing in linear context-free rewriting systems, are polynomial when the grammar is fixed, with exponent depending on the grammar (Satta, 1992; Kaji et al., 1992).

We next describe the setting in which our algorithm works. Consider a context-free grammar $\mathbf{G}$, as specified in Section 2.1. We assume that there are no unary rules possible in this grammar and also no ϵ rules. The immediate consequence of this assumption, is that for any sentence of length r, the number of rules that can be used in a derivation in this grammar is upper-bounded by $2r$: starting bottom up, the number of rules used at depth d of the tree has to be, at least, halved when moving to depth $d-1$. Therefore the total number of rules firing in a derivation is upper bounded by $r + r/2 + r/4 + ... = 2r$.

We now turn to describe a polynomial time algorithm for solving the Viterbi training problem with a context-free grammar of the kind specified above. In Section 3.5.1 we describe a semiring which will be used with a parsing algorithm, such as CKY or Earley's algorithms, during the execution of the algorithm for solving Viterbi training. In Section 3.5.2 we describe the algorithm itself and the way it uses the semiring from Section 3.5.1.

3.5.1 The Minkowski Semiring

The polynomial algorithm that we present for solving the Viterbi training problem (for a fixed grammar) requires us to use, as a sub-routine, a parsing algorithm over a semiring that we call "the Minkowski semiring," since the operation we use for summation with this semiring is Minkowski sum. A similar semiring has been used by Dyer (2010) for describing the computations done by a machine translation training algorithm. Coupling a semiring parsing algorithm (Goodman, 1998) with this semiring yields an enhanced recognition algorithm for parsing.

This algorithm takes as an input a string, a grammar and a vector over the natural numbers of the size of the number of rules in the grammar. The algorithm will determine whether there exists a derivation that uses each rule precisely the number of times specified in the input vector.

For brevity of notation, we denote by $|\mathbf{G}|$ the quantity $|\mathcal{R}(\mathbf{G})|$, the total number of rules in the context-free grammar $\mathbf{G}$. The Minkowski semiring $\mathcal{S}(\mathbf{G}) = \langle R, \oplus, \otimes \rangle$ is then defined as follows:

1. R is defined to be the power set of vectors over the natural numbers of length $|\mathbf{G}|$: $R = 2^{(\mathbb{N} \cup \{0\})^{|\mathbf{G}|}}$.

2. The addition operation $a \oplus b$ is defined to be the union of a and b: $a \oplus b \triangleq a \cup b$. This is also called "Minkowski sum."

3. The multiplication operation $a \otimes b$ is defined to be:

$$a \otimes b \quad \triangleq \quad \{v_1 + v_2 \mid v_1 \in a, v_2 \in b\},$$

   i.e. the set of the sum of all pairs of vectors from a and b.

Lemma 3.5 *For any context-free grammar* $\mathbf{G}$*,* $\mathcal{S}(\mathbf{G})$ *is a semiring.*

Proof The additive identity element for this semiring, $\overline{0}$, would be the empty set, $\emptyset$. The multiplicative identity element for this semiring, $\overline{1}$, would be $\{a\}$ where a is a natural number vector of length $|\mathbf{G}|$ containing only zeros. It is easy to verify distributivity of $\otimes$ over $\oplus$ and that $\overline{0}$ is an annihilator for $\mathcal{S}(\mathbf{G})$ with respect to $\otimes$. $\square$

For this semiring to be usable with a parsing algorithm, we are left to specify the weights in $\mathcal{S}(\mathbf{G})$ of the grammar rules. For the grammar rule indexed by i, we let its weight be $\{(0, \ldots, 0, 1, 0, \ldots, 0)\}$ where the 1 appears in the ith position of the vector.

In that case, whenever we use the Minkowski semiring to parse a string, the resulting weight for the parse would be a set of vectors, containing all possible counts for the grammar rules realizable by some derivation which uses the grammar. Now, we can construct an algorithm that takes as an input a string $\boldsymbol{s}$ and a grammar $\mathbf{G}$ and a vector of counts for the rules v, and outputs 1 if there exists a derivation that uses these counts, and 0 otherwise. This algorithm is the result of running the parsing algorithm with $\mathcal{S}(\mathbf{G})$ and then checking whether v

belongs to the final set returned by the parsing algorithm. We call this algorithm $\mathrm{CountRecognize}(\mathbf{G}, \boldsymbol{s}, v)$.

Let r be the length of $\boldsymbol{s}$. The complexity of $\mathrm{CountRecognize}$ takes $O(r^3)$ semiring operations using an algorithm such as CKY or Earley. Addition in this semiring takes linear time in the size of set operands (merging two sets). Multiplication of $a \otimes b$ takes $O(|a| \times |b|)$ to achieve. Note that at each point during the running of the parsing algorithm, the size of the set weight of each constituent is going to be polynomial in r, where the exponent of the polynomial is the number of rules in the grammar $|\mathbf{G}|$. This means that the total amount of time the algorithm takes to run is $O(r^{3+|\mathbf{G}|} + r^{|\mathbf{G}|}) = O(r^{3+|\mathbf{G}|})$.

3.5.2 A New Viterbi Training Algorithm

Equipped with $\mathrm{CountRecognize}$, we are ready to present the polynomial time algorithm for solving Viterbi training. The algorithm is based on the observation that there is a rather limited space of parameters that we need to explore in order to find the global maximum of the Viterbi likelihood.

For simplicity, and without loss of generality, we will assume that we have a single string $\boldsymbol{x}$ for which we are interested in finding the optimal Viterbi likelihood. The reason we do not lose generality is that we could always concatenate all data strings together with some new symbol σ separating each string, and then add a rule to the context-free grammar $S' \to S\sigma S'$ and $S' \to S\sigma$ where S' will function as a new start symbol in the CFG.

Now, consider the global maximum for the Viterbi likelihood, realized by $\mathbf{y}$. Clearly, the solution for the $\boldsymbol{\theta}$ would be the normalized counts of the rules from $\mathbf{y}$, since computing $\boldsymbol{\theta}$ this way can only increase the Viterbi likelihood – this solution for $\boldsymbol{\theta}$ is the maximum likelihood solution when observing $\mathbf{y}$. The consequence of this fact is simple. If r is the length of $\boldsymbol{x}$, then we know that $\mathbf{y}$ contains at most $2r$ rules. Therefore, we only need to consider, for $\boldsymbol{\theta}$, rational numbers with denominator smaller or equal to $2r$, if we are to find the global maximum. With this in mind, we can enumerate all possible vectors over natural numbers of length $|\mathbf{G}|$, where the sum of the elements in the vector is smaller or equal to $2r$.

A simple combinatorial fact is that the number of vectors over natural numbers (of length $|\mathbf{G}|$) such that the sum of the elements of each vector is exactly l for some $l \in \mathbb{N}$ is $\binom{l}{|\mathbf{G}|-1}$.[2] Therefore, the total number of vectors where the sum of elements is smaller than or equal to $2r$ would be:

[2]This is also called "the stars and bars problem."

$$\sum_{l=0}^{2r} \binom{l}{|\mathbf{G}|-1} = \sum_{l=0}^{2r} \binom{l+1}{|\mathbf{G}|} - \binom{l}{|\mathbf{G}|}$$

This is a telescopic sum that equals $\binom{2r}{|\mathbf{G}|}$, which is $O(r^{|\mathbf{G}|})$. For each vector v that we scan in this set of vectors of natural numbers with the sum being smaller or equal to to $2r$, we need to run $\mathrm{CountRecognize}(\boldsymbol{x}, \mathbf{G}, v)$. This means that the total running time of this algorithm is $O(r^{(3+2|\mathbf{G}|)})$.

3.5.3 Generalizing the Polynomial Time Algorithm

We conclude this section about the polynomial time algorithm for Viterbi training with two notes about generalizing it to other scenarios:

- We do not have to restrict ourselves necessarily to context-free grammars without ϵ-rules or unary rules. Any context-free grammar which entails a polynomial bound on the number of nodes in a parse tree for that grammar is sufficient. For example, instead of requiring that there are no ϵ-rules or unary rules, we can allow the grammar to be "lexicalized:" each rule that fires must lead to a terminal rewriting using constant number of rules. See discussion in Rambow and Satta (1994).

- As a matter of fact, we do not have to restrict ourselves to context-free grammars for this polynomial time algorithm to stay polynomial. We just require a polynomial time parsing algorithm. For example, synchronous grammars, which have a polynomial parsing time algorithm, could also be trained using Viterbi training in a polynomial time, assuming the number of rules firing in a derivation is bounded.

3.6 Discussion

Viterbi training is closely related to the k-means clustering problem, where the objective is to find k centroids for a given set of d-dimensional points such that the sum of distances between the points and the closest centroid is minimized. The analog for Viterbi EM for the k-means problem is the k-means clustering algorithm (Lloyd, 1982), a coordinate ascent algorithm for solving the k-means problem. It works by iterating between an E-like-step, in which each point is

assigned the closest centroid, and an M-like-step, in which the centroids are set to be the center of each cluster.

"k" in k-means corresponds, in a sense, to the size of our grammar. k-means has been shown to be NP-hard both when k varies and d is fixed and when d varies and k is fixed (Aloise et al., 2009; Mahajan et al., 2009), yet it is polynomial when both k and d are fixed. Analogously, we showed that Viterbi training is NP-hard when the grammar size varies, but that there is a polynomial time algorithm when the grammar is fixed.

Many combinatorial problems in NLP involving phrase-structure trees, alignments, and dependency graphs are hard (Sima'an, 1996; Goodman, 1998; Knight, 1999; Casacuberta and de la Higuera, 2000; Lyngsø and Pederson, 2002; Udupa and Maji, 2006; McDonald and Satta, 2007; DeNero and Klein, 2008, *inter alia*). Of special relevance to the results presented in this chapter is Abe and Warmuth (1992), who showed that the problem of finding maximum likelihood model of probabilistic automata is hard even for a single string and an automaton with two states.

3.7 Summary

In this chapter we analyzed the computational complexity of various learning problems for probabilistic grammars in the unsupervised setting. We found that learning problems, such as Viterbi training and log-likelihood maximization are NP-hard, and in fact even their approximation is NP-hard, when the grammar is part of the input to the learning algorithm. We also described an algorithm for solving Viteri training which is polynomial in the input sentences, but exponential in the grammar size.

Chapter 4

Learning-Theoretic Analysis of Probabilistic Grammars

In Chapter 3, we showed that the problem of maximizing likelihood (and problems similar to that) is computationally hard. In this chapter, we assume we are given a blackbox that maximizes likelihood, and turn to a learning-theoretic analysis that yields sample complexity results for the estimation of probabilistic grammars. Our results generalize to the case when an algorithm such as EM is used instead of a global likelihood maximizer.

Here, a sample complexity result quantifies the number of samples required to accurately learn a probabilistic grammar either in a supervised or in an unsupervised way. If bounds on the requisite number of samples are sufficiently tight, then they may offer guidance to learner performance, given various amounts of data and wide range of parametric families. Being able to analytically reason about the amount of data to annotate, and the relative gains in moving to a more restricted parametric family, could offer practical advantages to language engineers.

We develop a framework for deriving sample complexity bounds using the maximum likelihood principle for probabilistic grammars in a distribution-dependent setting. Distribution dependency is introduced here by making empirically justified assumptions about the distributions that generate the natural language data. Our framework uses and significantly extends ideas that have been introduced for deriving sample complexity bounds for probabilistic graphical models (Dasgupta, 1997). Maximum likelihood estimation is put in the empirical risk minimization framework (Vapnik, 1998) with the loss function being the log-loss. Following that, we develop a set of learning theoretic tools to explore rates of estimation convergence for probabilistic grammars. We also develop algorithms for perform-

ing empirical risk minimization. As a precursor to our analysis in the unsupervised setting, we also describe a learning-theoretic analysis for the *supervised* setting.

Much research has been devoted to the problem of learning finite state automata (which can be thought of as a class of grammars) in the PAC setting, leading to the conclusion that it is a very hard problem (Kearns and Valiant, 1989; Ron et al., 1995; Ron, 1995). Typically, the setting in these cases is different than our setting: error is measured as the probability mass of strings which are not identified correctly by the learned finite state automaton, instead of measuring KL divergence between the automaton and the true distribution. In addition, in many cases, there is also a focus on the distribution-free setting. To the best of our knowledge, it is still an open problem whether finite state automata are learnable in the distribution-dependent setting when measuring the error as the fraction of misidentified strings. More recent work (Clark and Thollard, 2004; Palmer and Goldberg, 2007) also gives treatment to probabilistic automata with an error measure which is more suitable for the probabilistic setting, such as KL divergence or variation distance. The work mentioned above also focuses on learning the structure of finite state machines. In our setting we assume that the grammar is fixed, and that our goal is to estimate its parameters.

We note an important connection to an earlier study about the learnability of probabilistic automata and hidden Markov models (Abe and Warmuth, 1992). In that earlier study, the authors provided positive results for the sample complexity for learning probabilistic automata—they showed that a polynomial sample is sufficient for maximum likelihood estimation. We demonstrate positive results for the more general class of probabilistic grammars which goes beyond probabilistic automata. Abe and Warmuth also showed that the problem of finding or even approximating the maximum likelihood solution for a two-state probabilistic automaton with an alphabet of an arbitrary size is hard. We extend our proofs from Chapter 3 to a proof that illustrates the NP-hardness of identifying the maximum likelihood solution for probabilistic grammars in the specific framework of "proper approximations" that we define in this chapter.

The main results described in this chapter are:

- A set of empirically motivated assumptions on distributions that generate natural language data. These assumptions make analysis of probabilistic grammars in the learning-theoretic setting manageable.

- A learning-theoretic analysis of the sample complexity of probabilistic grammars in the supervised setting.

- A similar analysis for the unsupervised setting.
- A description of a variant of the expectation-maximization algorithm that fits our learning-theoretic framework.

We also provide a description of a new normal form for probabilistic context-free grammars, which is easier to analyze in our learning framework (§4.2.2).

We note that our analysis of the supervised case is mostly described as a preparation for the unsupervised case. This analysis is mostly described as a preparation for the unsupervised case. In general, the families of probabilistic grammars we give a treatment to are parametric families, and the maximum likelihood estimator for these families is a consistent estimator in the supervised case. However, in the unsupervised case, lack of identifiability prevents us from getting these traditional consistency results. Also, the traditional results about the consistency of maximum likelihood estimation are based on the assumption that the sample is generated from the parametric family we are trying to estimate. This is not the case in our analysis, where the distribution that generates the data does not have to be a probabilistic grammar.

Some of the work in this chapter has been described in Cohen and Smith (2010b) and Cohen and Smith (2011).

4.1 Empirical Risk Minimization and Maximum Likelihood Estimation

We begin by introducing some notation and the general formulation of the framework of empirical risk minimization (Vapnik, 1998). In this general setting, we seek to construct a predictive model that maps inputs from space $\mathcal{X}$ to outputs from space $\mathcal{Z}$. In this chapter, $\mathcal{X}$ is a set of strings using some alphabet Σ ($\mathcal{X} \subseteq \Sigma^*$), and $\mathcal{Z}$ is be a set of derivations allowed by a grammar (e.g., a context-free grammar). We assume the existence of an unknown joint probability distribution $p(\boldsymbol{x}, \mathbf{y})$ over $\mathcal{X} \times \mathcal{Z}$. (Since we are discussing discrete input and output spaces, p will denote a probability mass function.) We are interested in estimating the distribution p from examples, either in a supervised setting, where we are provided with examples of the form $(\boldsymbol{x}, \mathbf{y}) \in \mathcal{X} \times \mathcal{Z}$, or in the unsupervised setting, where we are provided only with examples of the form $\boldsymbol{x} \in \mathcal{X}$. We first consider the supervised setting and return to the unsupervised setting in section 4.4. We will use q to denote the estimated distribution.

In order to estimate p as accurately as possible using $q(\boldsymbol{x}, \mathbf{y})$, we are interested in minimizing the *log-loss*, i.e., in finding q_{opt}, from a fixed family of distributions $\mathcal{Q}$ (also called "the concept space"), such that

$$\begin{aligned} q_{\text{opt}} &= \operatorname*{argmin}_{q \in \mathcal{Q}} \mathbb{E}_p[-\log q] \\ &= \operatorname*{argmin}_{q \in \mathcal{Q}} -\sum_{\boldsymbol{x}, \mathbf{y}} p(\boldsymbol{x}, \mathbf{y}) \log q(\boldsymbol{x}, \mathbf{y}). \end{aligned} \tag{4.1}$$

Note that if $p \in \mathcal{Q}$, then this quantity achieves the minimum when $q_{\text{opt}} = p$, in which case the value of the log-loss is the entropy of p. Indeed, more generally, the above optimization is equivalent to finding q such that it minimizes the KL divergence from p to q.

Since p is unknown, we cannot hope to minimize the log-loss directly. However, given a set of examples $(\boldsymbol{x}_1, \mathbf{y}_1), \ldots, (\boldsymbol{x}_n, \mathbf{y}_n)$ there is a natural candidate, the empirical distribution $\tilde{p}_n$, for use in Equation 4.1 instead of p, defined as:

$$\tilde{p}_n(\boldsymbol{x}, \mathbf{y}) = n^{-1} \sum_{i=1}^{n} \mathbb{I}\left\{(\boldsymbol{x}, \mathbf{y}) = (\boldsymbol{x}_i, \mathbf{y}_i)\right\}$$

where $\mathbb{I}\left\{(\boldsymbol{x}, \mathbf{y}) = (\boldsymbol{x}_i, \mathbf{y}_i)\right\}$ is 1 if $(\boldsymbol{x}, \mathbf{y}) = (\boldsymbol{x}_i, \mathbf{y}_i)$ and 0 otherwise.[1] We then set up the problem as the problem of *empirical risk minimization* (ERM), i.e., trying to find q such that:

$$\begin{aligned} q^* &= \operatorname*{argmin}_{q \in \mathcal{Q}} \mathbb{E}_{\tilde{p}_n}[-\log q] \\ &= \operatorname*{argmin}_{q \in \mathcal{Q}} -n^{-1} \sum_{i=1}^{n} \log q(\boldsymbol{x}_i, \mathbf{y}_i) \\ &= \operatorname*{argmax}_{q \in \mathcal{Q}} n^{-1} \sum_{i=1}^{n} \log q(\boldsymbol{x}_i, \mathbf{y}_i) \end{aligned} \tag{4.2}$$

Equation 4.2 immediately shows that minimizing empirical risk using the log-loss is equivalent to the maximizing likelihood (Section 2.2).[2]

[1] We note that $\tilde{p}_n$ itself is a random variable, because it depends on the sample drawn from p.

[2] We note that being able to attain the minimum through an hypothesis q^* is not necessarily possible in the general case. However, in our instantiations of ERM for probabilistic grammars, the minimum can be attained. In fact, in the unsupervised case the minimum can be attained by more than a single hypothesis. In these cases, q^* is arbitrarily chosen to be one of these minimizers.

As mentioned above, our goal is to estimate the probability distribution p while quantifying how accurate our estimate is. One way to quantify the estimation accuracy is by bounding the *excess risk*, which is defined as:

$$\mathcal{E}_p(q; \mathcal{Q}) = \mathcal{E}_p(q) \triangleq \mathbb{E}_p[-\log q] - \min_{q' \in \mathcal{Q}} \mathbb{E}_p[-\log q']$$

We are interested in bounding the excess risk for q^*, $\mathcal{E}_p(q^*)$. The excess risk is reduced to KL-divergence between p and q if $p \in \mathcal{Q}$, since in this case the quantity $\min_{q' \in \mathcal{Q}} \mathbb{E}[-\log q']$ is minimized with $q' = p$, and equals the entropy of p. In a typical case, where we do not necessarily have $p \in \mathcal{Q}$, then the excess risk of q is bounded from above by the KL divergence between p and q.

We can bound the excess risk by showing the double-sided convergence of the empirical process $R_n(\mathcal{Q})$, defined as follows:

$$R_n(\mathcal{Q}) \triangleq \sup_{q \in \mathcal{Q}} |\mathbb{E}_{\tilde{p}_n}[-\log q] - \mathbb{E}_p[-\log q]| \to 0$$

as $n \to \infty$. For any $\epsilon > 0$, if, for large enough n it holds that

$$\sup_{q \in \mathcal{Q}} |\mathbb{E}_{\tilde{p}_n}[-\log q] - \mathbb{E}_p[-\log q]| < \epsilon \tag{4.3}$$

(with high probability), then we can "sandwich" the following quantities:

$$\begin{aligned} \mathbb{E}_p[-\log q_{\text{opt}}] &\leq \mathbb{E}_p[-\log q^*] && (4.4) \\ &\leq \mathbb{E}_{\tilde{p}_n}[-\log q^*] + \epsilon \\ &\leq \mathbb{E}_{\tilde{p}_n}[-\log q_{\text{opt}}] + \epsilon \\ &\leq \mathbb{E}_p[-\log q_{\text{opt}}] + 2\epsilon && (4.5) \end{aligned}$$

where the inequalities come from the fact that q_{opt} minimizes the expected risk $\mathbb{E}_p[-\log q]$ for $q \in \mathcal{Q}$, and q^* minimizes the empirical risk $\mathbb{E}_{\tilde{p}_n}[-\log q]$ for $q \in \mathcal{Q}$. The consequence of Equations 4.4–4.5 is that the expected risk of q^* is at most 2ϵ away from the expected risk of q_{opt}, and as a result, we find the excess risk $\mathcal{E}_p(q^*)$, for large enough n, is smaller than 2ϵ. Intuitively, this means that, under a large sample, q^* does not give much worse results than q_{opt} under the criterion of the log-loss.

Unfortunately, the regularity conditions which are required for the convergence of $R_n(\mathcal{Q})$ do not hold because the log-loss can be unbounded. This means

that a modification is required for the empirical process in a way that will actually guarantee some kind of convergence. We give a treatment of this in the next section.

We note that all discussion of convergence in this section has been about convergence *in probability*. For example, we want Equation 4.3 to hold with high probability—for most samples of size n. We will make this notion more rigorous in section 4.1.2.

4.1.1 Empirical Risk Minimization and Structural Risk Minimization Methods

It has been noted in the literature (Vapnik, 1998; Koltchinskii, 2006) that often the class $\mathcal{Q}$ is too complex for empirical risk minimization using a fixed number of data points. It is therefore desirable in these cases to create a family of subclasses $\{\mathcal{Q}_\alpha \mid \alpha \in \mathcal{A}\}$ that have increasing complexity. The more data we have, the more complex our $\mathcal{Q}_\alpha$ can be for empirical risk minimization. Structural risk minimization (Vapnik, 1998) and the method of sieves (Grenander, 1981) are examples of methods that adopt this such an approach. Structural risk minimization, for example, can be represented in many cases as a penalization of the empirical risk method, using a regularization term.

In our case, the level of "complexity" is related to allocation of small probabilities to derivations in the grammar by a distribution $q \in \mathcal{Q}$. The basic problem is this: whenever we have a derivation with a small probability, the log-loss becomes very large (in absolute value), and this makes it hard to show the convergence of the empirical process $R_n(\mathcal{Q})$. Because grammars can define probability distributions over infinitely many discrete outcomes, probabilities can be arbitrarily small and log-loss can be arbitrarily large.

To solve this issue with the complexity of $\mathcal{Q}$, we define in section 4.3 a series of approximations $\{\mathcal{Q}_n \mid n \in \mathbb{N}\}$ for probabilistic grammars such that $\bigcup_n \mathcal{Q}_n = \mathcal{Q}$. Our framework for empirical risk minimization is then set up to minimize the empirical risk with respect to $\mathcal{Q}_n$, where n is the number of samples we draw for the learner:

$$q_n^* = \operatorname*{argmin}_{q \in \mathcal{Q}_n} \mathbb{E}_{\tilde{p}_n}[-\log q] \tag{4.6}$$

We are then interested in the convergence of the empirical process

$$R_n(\mathcal{Q}_n) = \sup_{q \in \mathcal{Q}_n} |\mathbb{E}_{\tilde{p}_n}[-\log q] - \mathbb{E}_p[-\log q]| \tag{4.7}$$

In section 4.3 we show that the minimizer q_n^* is an *asymptotic* empirical risk minimizer (in our specific framework), which means that $\mathbb{E}_p[-\log q_n^*] \to \mathbb{E}_p[-\log q^*]$. Since we have $\bigcup_n \mathcal{Q}_n = \mathcal{Q}$, the implication of having asymptotic empirical risk minimization is that we have $\mathcal{E}_p(q_n^*; \mathcal{Q}_n) \to \mathcal{E}_p(q^*; \mathcal{Q})$.

4.1.2 Sample Complexity Bounds

Knowing that we are interested in the convergence of $R_n(\mathcal{Q}_n) = \sup_{q \in \mathcal{Q}_n} |\mathbb{E}_{\tilde{p}_n}[-\log q] - \mathbb{E}_p[-\log q]|$, a natural question to ask is, "at what *rate* does this empirical process converge?"

Since the quantity $R_n(\mathcal{Q}_n)$ is a random variable, we need to give a probabilistic treatment to its convergence. More specifically, we ask the question that is typically asked when learnability is considered (Vapnik, 1998): "how many samples n are required so that with probability $1 - \delta$ we have $R_n(\mathcal{Q}_n) < \epsilon$?" Bounds on this number of samples are also called "sample complexity bounds," and in a distribution-free setting they are described as a function $N(\epsilon, \delta, \mathcal{Q})$, independent of the distribution p that generates the data.

A complete distribution-free setting is not appropriate for analyzing natural language. This setting poses technical difficulties with the convergence of $R_n(\mathcal{Q}_n)$ and needs to take into account pathological cases that can be ruled out in natural language data. Instead, we will make assumptions about p, parametrize these assumptions in several ways, and then calculate sample complexity bounds of the form $N(\epsilon, \delta, \mathcal{Q}, p)$, where the dependence on the distribution is expressed as dependence on the parameters in the assumptions about p.

The learning setting, then, can be described as follows. The user decides on a level of accuracy (ϵ) which the learning algorithm has to reach with confidence ($1-\delta$). Then, $N(\epsilon, \delta, \mathcal{Q}, p)$ samples are drawn from p and presented to the learning algorithm. The learning algorithm then returns a hypothesis according to Equation 4.6.

4.2 General Setting

We first lay out the connection between Section 4.1 and probabilistic grammars as they are described in Section 2.1. Going back to the notation in section 4.1, $\mathcal{Q}$ would be a collection of probabilistic grammars, parametrized by $\boldsymbol{\theta}$, and q would be a specific probabilistic grammar with a specific $\boldsymbol{\theta}$. We therefore treat the problem of ERM with probabilistic grammars as the problem of parameter

estimation—identifying $\boldsymbol{\theta}$ from complete data or incomplete data (strings $\boldsymbol{x}$ are visible but the derivations $\mathbf{y}$ are not). We can also view parameter estimation as the identification of a *hypothesis* from the concept space $\mathcal{Q} = \mathcal{H}(\mathbf{G}) = \{h_{\boldsymbol{\theta}}(\mathbf{y}) \mid \boldsymbol{\theta} \in \Theta_{\mathbf{G}}\}$ (where $h_{\boldsymbol{\theta}}$ is a distribution of the form of Equation 2.1) or, equivalently, from negated log-concept space $\mathcal{F}(\mathbf{G}) = \{-\log h_{\boldsymbol{\theta}}(\mathbf{y}) \mid \boldsymbol{\theta} \in \Theta_{\mathbf{G}}\}$ For simplicity of notation, we assume that there is a fixed grammar $\mathbf{G}$ and use $\mathcal{H}$ to refer to $\mathcal{H}(\mathbf{G})$ and $\mathcal{F}$ to refer to $\mathcal{F}(\mathbf{G})$.

4.2.1 Distributional Assumptions about Language

In this section, we describe a parametrization of assumptions we make about the distribution $p(\boldsymbol{x}, \mathbf{y})$, the distribution that generates derivations from $D(\mathbf{G})$ (note that p does not have to be a probabilistic grammar). We first describe empirical evidence about the decay of the frequency of long strings $\boldsymbol{x}$.

Figure 4.1 shows the frequency of sentence length for treebanks in various languages.[3] The trend in the plots clearly shows that in the extended tail of the curve, all languages have an exponential decay of probabilities as a function of sentence length. To test this, we performed a simple regression of frequencies using an exponential curve. We estimated each curve for each language using a curve of the form $f(l; c, \alpha) = cl^{\alpha}$. This estimation was done by minimizing squared error between the frequency vs. sentence length curve and the approximate version of this curve. The data points used for the approximation are (l_i, p_i), where l_i denotes sentence length and p_i denotes frequency, selected from the extended tail of the distribution. Extended tail here refers to all points with length longer than l_1, where l_1 is the length with the highest frequency in the treebank. The goal of focusing on the tail is to avoid approximating the head of the curve, which is actually a monotonically increasing function. We plotted the approximate curve together with a length versus frequency curve for new syntactic data. It can be seen (Figure 4.1) that the approximation is rather accurate in these corpora.

As a consequence of this observation, we make a few assumptions about $\mathbf{G}$ and $p(\boldsymbol{x}, \mathbf{y})$:

- Derivation length proportional to sentence length: There is an $\alpha \geq 1$ such that, for all $\mathbf{y}$, $|\mathbf{y}| \leq \alpha |s(\mathbf{y})|$. Further, $|\mathbf{y}| \geq |\boldsymbol{x}|$. (This prohibits unary

[3] Treebanks offer samples of cleanly segmented sentences. It is important to note that the distributions estimated may not generalize well to samples from other domains in these languages. Our argument is that the family of the estimated curve is reasonable, not that we can correctly estimate the curve's parameters.

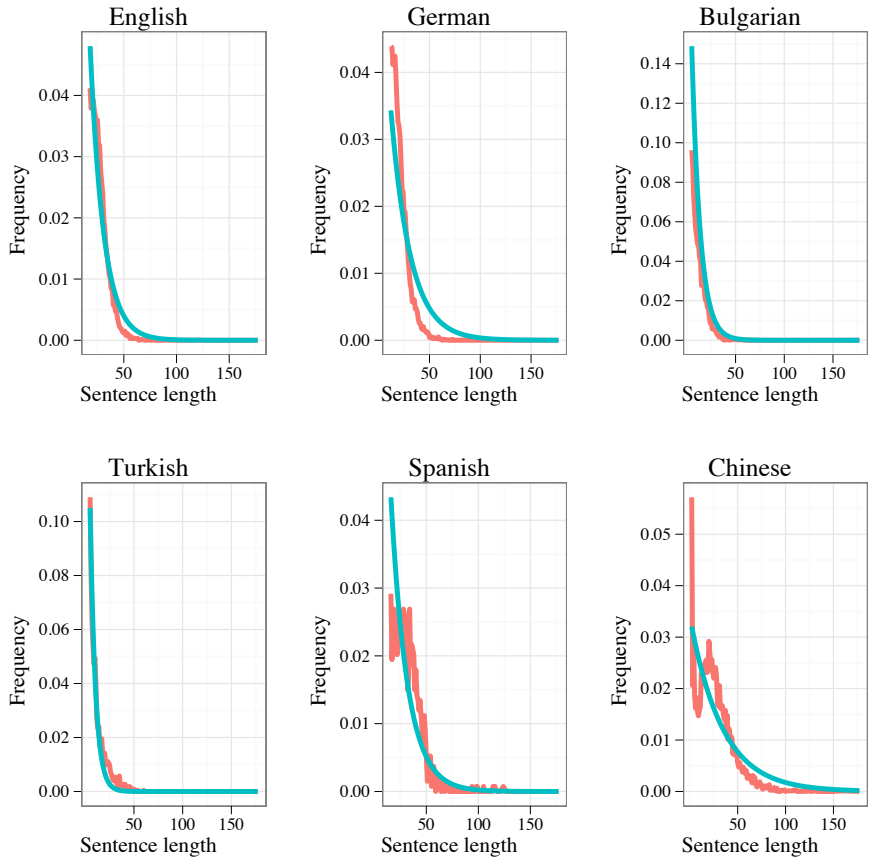


Figure 4.1: A plot of the tail of frequency vs. sentence length in treebanks for English, German, Bulgarian, Turkish, Spanish and Chinese. Red lines denote data from the treebank (not data used for estimation), blue lines denote an approximation using which uses an exponential function of the form $f(l; c, \alpha) = cl^{\alpha}$. The parameters (c, α) are (0.19, 0.92) for English, (0.06, 0.94) for German, (0.26, 0.89) for Bulgarian, (0.26, 0.83) for Turkish, (0.11, 0.93) for Spanish and (0.03, 0.97) for Chinese. Squared errors are 0.0005, 0.0003, 0.0007, 0.0003, 0.001, 0.002 for English, German, Bulgarian, Turkish, Spanish, and Chinese, respectively.

cycles.)

- Exponential decay of derivations: There is a constant $r < 1$ and a constant $L \geq 0$ such that $p(\mathbf{y}) \leq Lr^{|\mathbf{y}|}$. Note that the assumption here is about the frequency of length of separate derivations, and not the aggregated frequency of all sentences of a certain length (cf. the discussion above referring to Figure 4.1).

- Exponential decay of strings: Let $\Lambda(k) = |\{z \in D(\mathbf{G}) \mid |\mathbf{y}| = k\}|$ be the number derivations of length k in $\mathbf{G}$. We assume that $\Lambda(k)$ is an increasing function, and complete it such that it is defined over positive numbers by taking $\Lambda(t) \triangleq \Lambda(\lceil t \rceil)$. Taking r as above, we assume there exists a constant $q < 1$, such that $\Lambda^2(k)r^k \leq q^k$ (and as a consequence, $\Lambda(k)r^k \leq q^k$). This implies that the number of derivations of length k may be exponentially large (e.g., as with many PCFGs), but is bounded by $(q/r)^k$.

- Bounded expectations of rules: There is a $B < \infty$ such that $\mathbb{E}_p[\psi_{k,i}(\mathbf{y})] \leq B$ for all k and i.

These assumptions must hold for any p whose support consists of a finite set. These assumptions also hold in many cases when p itself is a probabilistic grammar. Also, we note that the last requirement of bounded expectations is optional, and it can be inferred from the rest of the requirements: $B \leq L/(1-q)^2$. We make this requirement explicit for simplicity of notation later. We denote the family of distributions that satisfy all of the requirements above by $\mathcal{P}(\alpha, L, r, q, B, \mathbf{G})$.

There are other cases in the literature of language learning where additional assumptions are made on the learned family of models in order to obtain positive learnability results. For example, Clark and Thollard (2004) put a bound on the expected length of strings generated from any state of probabilistic finite state automata, which resembles the exponential decay of strings we have for p in this chapter.

An immediate consequence of the above assumptions is that the entropy of p is finite and bounded by a quantity that depends on L, r and q.[4] Bounding entropy of labels (derivations) given inputs (sentences) is a common way to quantify the *noise* in a distribution. Here, both the *sentential* entropy ($H_s(p) = -\sum_{\boldsymbol{x}} p(\boldsymbol{x}) \log p(\boldsymbol{x})$) is bounded as well as the *derivational* entropy ($H_d(p) = -\sum_{\boldsymbol{x},\mathbf{y}} p(\boldsymbol{x}, \mathbf{y}) \log p(\boldsymbol{x}, \mathbf{y})$). This is stated in the following result:

[4]For simplicity and consistency with the log-loss, we measure entropy in nats, which means we use the natural logarithm when computing entropy.

Proposition 4.1 *Let $p \in \mathcal{P}(\alpha, L, r, q, B, \mathbf{G})$ be a distribution. Then, we have:*

$$H_s(p) \leq H_d(p) \leq -\log L + \frac{L \log r}{(1-q)^2} \log \frac{1}{r} + \frac{\lceil (1+\log L)/\log \frac{1}{r} \rceil}{e} \Lambda \left(\left\lceil \frac{1+\log L}{\log \frac{1}{r}} \right\rceil \right)$$

Proof First note that $H_s(p) \leq H_d(p)$ holds because the sentential probability distribution $p(\boldsymbol{x})$ is a coarser version of the derivational probability distribution $p(\boldsymbol{x}, \mathbf{y})$. Now, consider $p(\boldsymbol{x}, \mathbf{y})$. For simplicity of notation, we use $p(\mathbf{y})$ instead of $p(\boldsymbol{x}, \mathbf{y})$. The yield of $\mathbf{y}$, $\boldsymbol{x}$, is a function of $\mathbf{y}$, and therefore can be omitted from the distribution. It holds that:

$$\begin{aligned} H_d(p) &= -\sum_{\mathbf{y}} p(\mathbf{y}) \log p(\mathbf{y}) \\ &= -\sum_{z \in \mathbf{y}_1} p(\mathbf{y}) \log p(\mathbf{y}) - \sum_{z \in \mathbf{y}_2} p(\mathbf{y}) \log p(\mathbf{y}) \\ &= H_d(p, \mathbf{y}_1) + H_d(p, \mathbf{y}_2) \end{aligned}$$

where $\mathbf{y}_1 = \{\mathbf{y} \mid p(\mathbf{y}) > 1/e\}$ and $\mathbf{y}_2 = \{\mathbf{y} \mid p(\mathbf{y}) \leq 1/e\}$. Note that the function $-\alpha \log \alpha$ reaches its maximum for $\alpha = 1/e$. We therefore have:

$$H_d(p, \mathbf{y}_1) \leq \frac{|\mathbf{y}_1|}{e}$$

We give a bound on $|\mathbf{y}_1|$, the number of "high probability" derivations. Since we have $p(\boldsymbol{x}, \mathbf{y}) \leq L r^{|\mathbf{y}|}$, we can find the maximum length of a derivation that has a probability of more than $1/e$ (and hence, it may appear in $\mathbf{y}_1$) by solving $1/e \leq L r^{|\mathbf{y}|}$ for $|\mathbf{y}|$, which leads to $|\mathbf{y}| \leq \log(1/eL)/\log r$. Therefore, there are at most $\sum_{k=1}^{\lceil (1+\log L)/\log \frac{1}{r} \rceil} \Lambda(k)$ derivations in $|\mathbf{y}_1|$ and therefore we have

$$\begin{aligned} |\mathbf{y}_1| &\leq \left\lceil (1+\log L)/\log \frac{1}{r} \right\rceil \Lambda \left(\left\lceil (1+\log L)/\log \frac{1}{r} \right\rceil \right) \\ H_d(p, \mathbf{y}_1) &\leq \frac{\lceil (1+\log L)/\log \frac{1}{r} \rceil}{e} \Lambda \left(\left\lceil (1+\log L)/\log \frac{1}{r} \right\rceil \right) \end{aligned} \tag{4.8}$$

where we use the monotonicity of Λ. Consider $H_d(p, \mathbf{y}_2)$ (the "low probability"

derivations). We have:

$$
\begin{aligned}
H_d(p, \mathbf{y}_2) &\leq -\sum_{\mathbf{y} \in \mathbf{y}_2} L r^{|\mathbf{y}|} \log \left(L r^{|\mathbf{y}|} \right) \\
&\leq -\log L - (L \log r) \sum_{\mathbf{y} \in \mathbf{y}_2} |\mathbf{y}| r^{|\mathbf{y}|} \\
&\leq -\log L - (L \log r) \sum_{k=1}^{\infty} \Lambda(k) k r^k \\
&\leq -\log L - (L \log r) \sum_{k=1}^{\infty} k q^k \qquad (4.9) \\
&= -\log L + \frac{L \log r}{(1-q)^2} \log \frac{1}{q} \qquad (4.10)
\end{aligned}
$$

where Equation 4.9 holds from the assumptions about p. Putting Equation 4.8 and Equation 4.10 together, we obtain the result. □

We note that another common way to quantify the noise in a distribution is through the notion of Tsybakov noise (Tsybakov, 2004; Koltchinskii, 2006). We discuss this further in section 4.6.1, where we show that Tsybakov noise is too permissive, and probabilistic grammars do not satisfy its conditions.

4.2.2 Limiting the Degree of the Grammar

When approximating a family of probabilistic grammars, it is much more convenient when the degree of the grammar is limited. See Section 4.3 for discussion. In this chapter, we limit the degree of the grammar by making the assumption that all $N_k \leq 2$. This assumption may seem, at first glance, somewhat restrictive, but we show next that for probabilistic context-free grammars (and as a consequence, other formalisms, such as tree substitution grammars), this assumption does not limit the total generative capacity that we can have across all context-free grammars.

We first show that any context-free grammar with arbitrary degree can be mapped to a corresponding grammar that generates derivations equivalent to derivations in the original grammar. Such a grammar is also called a "covering grammar" (Nijholt, 1980; Leermakers, 1989). Let $\mathbf{G}$ be a CFG. Let A be the kth nonterminal. Consider the rules $A \rightarrow \alpha_i$ for $i \leq N_k$ where A appears on the left side. For each rule $A \rightarrow \alpha_i$, $i < N_k$, we create a new nonterminal in $\mathbf{G}'$ such

Context-free grammar

S → NP VP
S → NP VP NP
S → NP VP PP
PP → P NP
NP → N | DET N
VP → V
N → *park* | *boy* | *girl* | *I*
DET → *a* | *the*
P → *at* | *on* | *in*
V → *watch*

Binarized grammar

S → NP VP | S1
S1 → NP VP | NP | S2
S2 → NP VP
PP
PP → P NP
NP → N | DET N
VP → V
N → *park* | N1
N1 → *boy* | N2
N2 → *girl* | N3
N3 → *I*
DET → *a* | *the*
P → *at* | P1
P1 → *on* | P2
P2 → *in*
V → *watch*

Figure 4.2: Exampe of a context-free grammar and its equivalent binarized form.

that A_i has two rewrite rules: $A_i \rightarrow \alpha_i$ and $A_i \rightarrow A_{i+1}$. In addition, we create rules $A \rightarrow A_1$ and $A_{N_k} \rightarrow \alpha_{N_k}$. Figure 4.2 demonstrates an example of this transformation on a small context-free grammar.

It is easy to verify that the resulting grammar $\mathbf{G}'$ has an equivalent capacity to the original CFG, $\mathbf{G}$. A simple transformation that converts each derivation in the new grammar to a derivation in the old grammar would involve collapsing any path of nonterminals added to $\mathbf{G}'$ (i.e. all A_i for nonterminal A) so that we end up with nonterminals from the original grammar only. Similarly, any derivation in $\mathbf{G}$ can be converted to a derivation in $\mathbf{G}'$, by adding new nonterminals through unary application of rules of the form $A_i \rightarrow A_{i+1}$. Given a derivation $\mathbf{y}$ in $\mathbf{G}$, we denote by $\Upsilon_{\mathbf{G} \mapsto \mathbf{G}'}(\mathbf{y})$ the corresponding derivation in $\mathbf{G}'$ after adding the new non-terminals A_i to $\mathbf{y}$. Throughout this chapter, we will refer to the normalized form of $\mathbf{G}'$ as a "binary normal form."[5]

[5]We note that this notion of binarization is different from previous types of binarization appearing in computational linguistics for grammars. Typically in previous work about binarized grammars such as context free-grammars, the grammars are constrained to have at most two nonterminals in the right side in Chomsky normal form. Another form of binarization for linear context-free rewriting systems is restriction of the *fan-out* of the rules to two (Gómez-Rodríguez

Note that K', the number of multinomials in the binary normal form, is a function of both the number of nonterminals in the original grammar and the number of rules in that grammar. More specifically, we have that $K' = \sum_{k=1}^{K} N_k + K$. To make the equivalence complete, we need to show that any *probabilistic* context-free grammar can be translated to a PCFG with $\max_k N_k \leq 2$ such that the two PCFGs induce the same equivalent distributions over derivations.

Utility Lemma 4.2 *Let* $a_i \in [0, 1]$, $i \in \{1, \ldots, N\}$ *such that* $\sum_i a_i = 1$. *Define* $b_1 = a_1$, $c_1 = 1 - a_1$, $b_i = \left(\frac{a_i}{a_{i-1}}\right)\left(\frac{b_{i-1}}{c_{i-1}}\right)$ *and* $c_i = 1 - b_i$ *for* $i \geq 2$. *Then* $a_i = \left(\prod_{j=1}^{i-1} c_j\right) b_i$.

Proof See Appendix A. □

Theorem 4.3 *Let* $\langle \mathbf{G}, \boldsymbol{\theta} \rangle$ *be a probabilistic context-free grammar. Let* $\mathbf{G}'$ *be the binarizing transformation of* $\mathbf{G}$ *as defined above. Then, there exists* $\boldsymbol{\theta}'$ *for* $\mathbf{G}'$ *such that for any* $\mathbf{y} \in D(\mathbf{G})$ *we have* $p(\mathbf{y} \mid \boldsymbol{\theta}, \mathbf{G}) = p(\Upsilon_{\mathbf{G} \mapsto \mathbf{G}'}(\mathbf{y}) \mid \boldsymbol{\theta}', \mathbf{G}')$.

Proof For the grammar $\mathbf{G}$, index the set $\{1, ..., K\}$ with nonterminals ranging from A_1 to A_K. Define $\mathbf{G}'$ as above. We need to define $\boldsymbol{\theta}'$. Index the multinomials in $\mathbf{G}'$ by (k, i), each having two events. Let $\mu_{(k,i),1} = \theta_{k,i}$, $\mu_{(k,i),2} = 1 - \theta_{k,i}$ for $i = 1$ and set $\mu_{k,i,1} = \theta_{k,i} / \mu_{(k,i-1),2}$, and $\mu_{(k,i-1),2} = 1 - \mu_{(k,i-1),2}$.

$\langle \mathbf{G}', \boldsymbol{\mu} \rangle$ is a *weighted* context-free grammar such that the $\mu_{(k,i),1}$ corresponds to the ith event in the k multinomial of the original grammar. Let $\mathbf{y}$ be a derivation in $\mathbf{G}$ and $\mathbf{y}' = \Upsilon_{\mathbf{G} \mapsto \mathbf{G}'}(\mathbf{y})$. Then, from Utility Lemma 4.2 and the construction of

and Satta, 2009; Gildea, 2010). We, however, limit the number of *rules* for each nonterminal (or more generally, the number of elements in each multinomial).

g', we have that:

$$
\begin{aligned}
p(\mathbf{y} \mid \boldsymbol{\theta}, \mathbf{G}) &= \prod_{k=1}^{K} \prod_{i=1}^{N_k} \theta_{k,i}^{\psi_{k,i}(\mathbf{y})} \\
&= \prod_{k=1}^{K} \prod_{i=1}^{N_k} \prod_{l=1}^{\psi_{k,i}(\mathbf{y})} \theta_{k,i} \\
&= \prod_{k=1}^{K} \prod_{i=1}^{N_k} \prod_{l=1}^{\psi_{k,i}(\mathbf{y})} \left(\prod_{j=1}^{i-1} \mu_{(k,j),2} \right) \mu_{(k,i),1} \\
&= \prod_{k=1}^{K} \prod_{i=1}^{N_k} \left(\prod_{j=1}^{i-1} \mu_{(k,j),2}^{\psi_{k,i}(\mathbf{y})} \right) \mu_{(k,i),1}^{\psi_{k,i}(\mathbf{y})} \\
&= \prod_{k=1}^{K} \prod_{j=1}^{N_k} \prod_{i=1}^{2} \mu_{(k,j),i}^{\psi_{k,j}(\mathbf{y}')} \\
&= p(\mathbf{y}' \mid \boldsymbol{\mu}, \mathbf{G}')
\end{aligned}
$$

From Chi (1999), we know that the weighted grammar $\langle \mathbf{G}', \boldsymbol{\mu} \rangle$ can be converted to a probabilistic context-free grammar $\langle \mathbf{G}', \boldsymbol{\theta}' \rangle$, through a construction of $\boldsymbol{\theta}'$ based on $\boldsymbol{\mu}$, such that $p(\mathbf{y}' \mid \boldsymbol{\mu}, \mathbf{G}') = p(\mathbf{y}' \mid \boldsymbol{\theta}', \mathbf{G}')$. □

The proof for Theorem 4.3 gives a construction the parameters $\boldsymbol{\theta}'$ of $\mathbf{G}'$ such that $\langle \mathbf{G}, \boldsymbol{\theta} \rangle$ is equivalent to $\langle \mathbf{G}', \boldsymbol{\theta}' \rangle$. The construction of $\boldsymbol{\theta}'$ can also be reversed: given $\boldsymbol{\theta}'$ for $\mathbf{G}'$, we can construct $\boldsymbol{\theta}$ for $\mathbf{G}$ so that again we have equivalence between $\langle \mathbf{G}, \boldsymbol{\theta} \rangle$ and $\langle \mathbf{G}', \boldsymbol{\theta}' \rangle$.

In this section, we focused on presenting parametrized, empirically justified distributional assumptions about language data that will make the analysis in later sections more manageable. We showed that these assumptions bound the amount of entropy as a function of the assumption parameters. We also made an assumption about the *structure* of the grammar family, and showed that it entails no loss of generality for context-free grammars. Many other formalisms can follow similar arguments to show that the structural assumption is justified for them as well.

4.3 Proper Approximations

In order to follow the empirical risk minimization described in section 4.1.1, we have to define a series of approximations for $\mathcal{F}$, which we denote by the log-

concept spaces $\mathcal{F}_1, \mathcal{F}_2, \ldots$. We also have to replace two-sided uniform convergence (Equation 4.3) with convergence on the sequence of concept spaces we defined (Equation 4.7). The concept spaces in the sequence vary as a function of the number of samples we have. We next construct the sequence of concept spaces, and in section 4.4 we return to the learning model. Our approximations are based on the concept of *bounded approximations* (Abe et al., 1990; Dasgupta, 1997), which were originally designed for graphical models.[6] A bounded approximation is a subset of a concept space which is controlled by a parameter that determines its tightness. Here we use this idea to define a series of subsets of the original concept space $\mathcal{F}$ as approximations, while having two asymptotic properties that control the series' tightness.

Let $\mathcal{F}_m$ (for $m \in \{1, 2, \ldots\}$) be a sequence of concept spaces. We consider three properties of elements of this sequence, which should hold for $m > M$ for a fixed M.

The first is *containment* in $\mathcal{F}$:

$$\mathcal{F}_m \subseteq \mathcal{F}$$

The second property is *boundedness*:

$$\exists K_m \geq 0, \forall f \in \mathcal{F}_m, \;\; \mathbb{E}\left[|f| \times \mathbb{I}\left\{|f| \geq K_m\right\}\right] \leq \epsilon_{\text{bound}}(m)$$

where ϵ_{bound} is a non-increasing function such that $\epsilon_{\text{bound}}(m) \underset{m\to\infty}{\longrightarrow} 0$. This states that the expected values of functions from $\mathcal{F}_m$ on values larger than some K_m is small. This is required to obtain uniform convergence results in the revised empirical risk minimization model from section 4.1.1. Note that K_m can grow arbitrarily large.

The third property is *tightness*:

$$\exists C_m \in \mathcal{F} \to \mathcal{F}_m, \;\; p\left(\bigcup_{f\in\mathcal{F}} \{\mathbf{y} \mid C_m(f)(\mathbf{y}) - f(\mathbf{y}) \geq \epsilon_{\text{tail}}(m)\}\right) \leq \epsilon_{\text{tail}}(m)$$

where ϵ_{tail} is a non-increasing function such that $\epsilon_{\text{tail}}(m) \underset{m\to\infty}{\longrightarrow} 0$, and C_m denotes an operator that maps functions in $\mathcal{F}$ to $\mathcal{F}_m$. This ensures that our approximation

[6] There are other ways to manage the unboundedness of KL divergence in the language learning literature. Clark and Thollard (2004), for example, decompose the KL divergence between probabilistic finite-state automata into several terms according to a decomposition Carrasco (1997) and then bound each term separately.

actually converges to the original concept space $\mathcal{F}$. We will show in section 4.3.3 that this is actually a well-motivated characterization of convergence for probabilistic grammars in the supervised setting.

We say that the sequence $\mathcal{F}_m$ *properly approximates* $\mathcal{F}$ if there exist $\epsilon_{\text{tail}}(m)$, $\epsilon_{\text{bound}}(m)$, and C_m such that, for all m larger than some M, containment, boundedness, and tightness all hold.

In a good approximation, K_m would increase at a fast rate as a function of m and $\epsilon_{\text{tail}}(m)$ and $\epsilon_{\text{bound}}(m)$ and decrease quickly as a function of m. As we will see in section 4.4, we cannot have an arbitrarily fast convergence rate (by, for example, taking a subsequence of $\mathcal{F}_m$), because the size of K_m has a great effect on the number of samples required to obtain accurate estimation.

4.3.1 Constructing Proper Approximations for Probabilistic Grammars

We now focus on constructing proper approximations for probabilistic grammars whose degree is limited to 2.

Proper approximations could, in principle, be used with losses other than the log-loss, though their main use is for unbounded losses. Starting from this point in the chapter, we focus on using such proper approximations with the log-loss.

We construct $\mathcal{F}_m$. For each $f \in \mathcal{F}$ we define a transformation $T(f, \gamma)$ that shifts every binomial parameter $\boldsymbol{\theta}_k = \langle \theta_{k,1}, \theta_{k,2} \rangle$ in the probabilistic grammar by at most γ:

$$\langle \theta_{k,1}, \theta_{k,2} \rangle \leftarrow \begin{cases} \langle \gamma, & 1-\gamma \rangle & \text{if} \quad \theta_{k,1} < \gamma \\ \langle 1-\gamma, & \gamma \rangle & \text{if} \quad \theta_{k,1} > 1-\gamma \\ \langle \theta_{k,1}, & \theta_{k,2} \rangle & \text{otherwise} \end{cases}$$

Note that $T(f, \gamma) \in \mathcal{F}$ for any $\gamma \leq 1/2$. Fix a constant $s > 1$.[7] We denote by $T(\boldsymbol{\theta}, \gamma)$ the same transformation on $\boldsymbol{\theta}$ (which outputs the new shifted parameters) and we denote by $\Theta_{\mathbf{G}}(\gamma) = \Theta(\gamma)$ the set $\{T(\boldsymbol{\theta}, \gamma) \mid \boldsymbol{\theta} \in \Theta_{\mathbf{G}}\}$. For each $m \in \mathbb{N}$, define $\mathcal{F}_m = \{T(f, m^{-s}) \mid f \in \mathcal{F}\}$.

When considering our approach to approximate a probabilistic grammar by increasing its parameter probabilities to be over a certain threshold, it becomes clear why we are required to limit the grammar to have only two rules and why we are

[7] By varying s we get a family of approximations. The larger s is, the tighter the approximation is. Also, the larger s is, as we see later, the more loose our sample complexity bound will be.

Rule	$\boldsymbol{\theta}$	approx. #1	approx. #2	approx. #3
S $\rightarrow$ NP VP	0.09	0.1	0.1	0.1
S $\rightarrow$ NP	0.11	0.11	0.1	0.105
S $\rightarrow$ VP	0.8	0.79	0.8	0.795

Table 4.1: Example of a PCFG where there is more than a single way to approximate it by truncation, because it has more than two rules. We assume $\gamma = 0.1$.

required to use the normal from Section 4.2.2 with grammars of degree 2. Consider the PCFG rules in Table 4.1. There are different ways to move probability mass to the rule with small probability. This leads to a problem with identifability of the approximation: how does one decide how to reallocate probability to the small probability rules? By binarizing the grammar in advance, we arrive at a single way to reallocate mass when required (i.e., move mass from the high-probability rule to the low-probability rule). This leads to a simpler proof for sample complexity bounds and a single bound (rather than different bounds depending on different smoothing operators). We note, however, that the choices made in binarizing the grammar imply a particular way of smoothing the probability across the original rules.

We now describe how the construction of approximations mentioned above satisfies the properties in section 4.3, specifically the boundedness property and the tightness property.

Proposition 4.4 *Let $p \in \mathcal{P}(\alpha, L, r, q, B, \mathbf{G})$ and let $\mathcal{F}_m$ as defined above. There exists a constant $\beta = \beta(L, q, s, N) > 0$ such that $\mathcal{F}_m$ has the boundedness property with $K_m = sN \log^3 m$ and $\epsilon_{\text{bound}}(m) = m^{-\beta \log m}$.*

Proof See Appendix A. □

We next show that $\mathcal{F}_m$ is tight with respect to $\mathcal{F}$ with $\epsilon_{\text{tail}}(m) = \dfrac{N \log^2 m}{m^s - 1}$.

Proposition 4.5 *Let $p \in \mathcal{P}(\alpha, L, r, q, B, \mathbf{G})$ and let $\mathcal{F}_m$ as defined above. There exists an M such that for any $m > M$ we have:*

$$p\left(\bigcup_{f \in \mathcal{F}} \{\mathbf{y} \mid C_m(f)(\mathbf{y}) - f(\mathbf{y}) \geq \epsilon_{\text{tail}}(m)\}\right) \leq \epsilon_{\text{tail}}(m)$$

for $\epsilon_{\text{tail}}(m) = \dfrac{N \log^2 m}{m^s - 1}$ and $C_m(f) = T(f, m^{-s})$.

Proof See Appendix A. □

We now have proper approximations for probabilistic grammars. These approximations are defined as a series of probabilistic grammars, related to the family of probabilistic grammars we are interested in estimating. They consist of three properties: containment (they are a subset of the family of probabilistic grammars we are interested in estimating), boundedness (their log-loss does not diverge to infinity quickly) and they are tight (there is a small probability mass at which they are not tight approximations).

4.3.2 Coupling Bounded Approximations with Number of Samples

At this point, the number of samples n is decoupled from the bounded approximation ($\mathcal{F}_m$) that we choose for grammar estimation. To couple between these two, we need to define m as a function of the number of samples, $m(n)$. As mentioned above, there is a clear trade-off between choosing a fast rate for $m(n)$ (such as $m(n) = n$) and a slower rate (such as $m(n) = \log n$). The faster the rate is, the tighter the family of approximations that we use for n samples. However, if the rate is too fast, then K_m grows quickly as well. In that case, because our sample complexity bounds are increasing functions of such K_m, the bounds will degrade.

To balance the trade-off, we choose $m(n) = n$. As we see later, this gives sample complexity bounds which are asymptotically interesting for both the supervised and unsupervised case.

4.3.3 Asymptotic Empirical Risk Minimization

It would be compelling to determine whether the empirical risk minimizer over $\mathcal{F}_n$ is an *asymptotic empirical risk minimizer*. This would mean that the risk of the empirical risk minimizer over $\mathcal{F}_n$ converges to the risk of the maximum likelihood estimate. As a conclusion to this section about proper approximations, we motivate the three requirements that we posed on proper approximations by showing that this is indeed true. We now unify n, the number of samples, and m, the index of the approximation of the concept space $\mathcal{F}$. Let f_n^* be the minimizer of the empirical risk over $\mathcal{F}$, ($f_n^* = \operatorname{argmin}_{f \in \mathcal{F}} \mathbb{E}_{\tilde{p}_n}[f]$) and let g_n be the minimizer of the empirical risk over $\mathcal{F}_n$ ($g_n = \operatorname{argmin}_{f \in \mathcal{F}_n} \mathbb{E}_{\tilde{p}_n}[f]$).

Let $D = \{\mathbf{y}_1, ..., \mathbf{y}_n\}$ be a sample from $p(\mathbf{y})$. The operator $(g_n =) \operatorname{argmin}_{f \in \mathcal{F}_n} \mathbb{E}_{\tilde{p}_n}[f]$ is an asymptotic empirical risk minimizer if $\mathbb{E}\left[\mathbb{E}_{\tilde{p}_n}[g_n] - \mathbb{E}_{\tilde{p}_n}[f_n^*]\right] \to 0$ as $n \to \infty$ (Shalev-Shwartz et al., 2009). Then, we have the following:

Lemma 4.6 *Denote by $\mathcal{Z}_{\epsilon,n}$ the set $\bigcup_{f \in \mathcal{F}} \{\mathbf{y} \mid C_n(f)(\mathbf{y}) - f(\mathbf{y}) \geq \epsilon\}$. Denote by $A_{\epsilon,n}$ the event "one of $y_i \in D$ is in $\mathcal{Z}_{\epsilon,n}$." Then if $\mathcal{F}_n$ properly approximates $\mathcal{F}$ then:*

$$\begin{aligned} &\mathbb{E}\left[\mathbb{E}_{\tilde{p}_n}[g_n] - \mathbb{E}_{\tilde{p}_n}[f_n^*]\right] \\ &\leq \left|\mathbb{E}\left[\mathbb{E}_{\tilde{p}_n}[C_n(f_n^*)] \mid A_{\epsilon,n}\right]\right| p(A_{\epsilon,n}) + \left|\mathbb{E}\left[\mathbb{E}_{\tilde{p}_n}[f_n^*] \mid A_{\epsilon,n}\right]\right| p(A_{\epsilon,n}) + \epsilon_{\text{tail}}(n) \end{aligned} \tag{4.11}$$

where the expectations are taken with respect to the dataset D.

Proof See Appendix A. □

Proposition 4.7 *Let $D = \{\mathbf{y}_1, ..., \mathbf{y}_n\}$ be a sample of derivations from $\mathbf{G}$. Then $g_n = \operatorname{argmin}_{f \in \mathcal{F}_n} \mathbb{E}_{\tilde{p}_n}[f]$ is an asymptotic empirical risk minimizer.*

Proof Let $f_0 \in \mathcal{F}$ be the concept that puts uniform weights over $\boldsymbol{\theta}$, i.e., $\boldsymbol{\theta}_k = \langle \frac{1}{2}, \frac{1}{2} \rangle$ for all k. Note that

$$\begin{aligned} &|\mathbb{E}[\mathbb{E}_{\tilde{p}_n}[f_n^*] \mid A_{\epsilon,n}]| p(A_{\epsilon,n}) \\ &\leq |\mathbb{E}[\mathbb{E}_{\tilde{p}_n}[f_0] \mid A_{\epsilon,n}]| p(A_{\epsilon,n}) = \tfrac{\log 2}{n} \textstyle\sum_{l=1}^{n} \sum_{k,i} \mathbb{E}[\psi_{k,i}(y_l) \mid A_{\epsilon,n}] p(A_{\epsilon,n}) \end{aligned}$$

Let $A_{j,\epsilon,n}$ for $j \in \{1, \ldots, n\}$ be the event "$\mathbf{y}_j \in \mathcal{Z}_{\epsilon,n}$". Then $A_{\epsilon,n} = \bigcup_j A_{j,\epsilon,n}$. We have that:

$$\begin{aligned} &\mathbb{E}[\psi_{k,i}(y_l) \mid A_{\epsilon,n}] p(A_{\epsilon,n}) \leq \sum_j \sum_{y_l} p(y_l, A_{j,\epsilon,n}) |y_l| \\ &\leq \sum_{j \neq l} \sum_{y_l} p(y_l) p(A_{j,\epsilon,n}) |y_l| + \sum_{y_l} p(y_l, A_{l,\epsilon,n}) |y_l| \qquad (4.12) \\ &\leq \left(\sum_{j \neq l} p(A_{j,\epsilon,n}) \right) B + \mathbb{E}[\psi_{k,i}(\mathbf{y}) \mid \mathbf{y} \in \mathcal{Z}_{\epsilon,n}] p(z \in \mathcal{Z}_{\epsilon,n}) \\ &\leq (n-1) B p(z \in \mathcal{Z}_{\epsilon,n}) + \mathbb{E}[\psi_{k,i}(\mathbf{y}) \mid \mathbf{y} \in \mathcal{Z}_{\epsilon,n}] p(z \in \mathcal{Z}_{\epsilon,n}) \end{aligned}$$

where Equation 4.12 comes from $\mathbf{y}_l$ being independent. Also, B is the constant from section 4.2.1. Therefore, we have:

$$\begin{aligned} &\frac{1}{n} \sum_{l=1}^{n} \sum_{k,i} \mathbb{E}[\psi_{k,i}(y_l) \mid A_{\epsilon,n}] p(A_{\epsilon,n}) \leq \\ &\sum_{k,i} \left(\mathbb{E}[\psi_{k,i}(\mathbf{y}) \mid \mathbf{y} \in \mathcal{Z}_{\epsilon,n}] p(z \in \mathcal{Z}_{\epsilon,n}) + (n-1) B p(z \in \mathcal{Z}_{\epsilon,n}) \right) \end{aligned}$$

From the construction of our proper approximations (Proposition 4.5), we know that only derivations of length $\log^2 n$ or greater can be in $\mathcal{Z}_{\epsilon,n}$. Therefore:

$$\mathbb{E}[\psi_{k,i} \mid \mathcal{Z}_{\epsilon,n}]p(\mathcal{Z}_{\epsilon,n}) \leq \sum_{z:|\mathbf{y}|>\log^2 n} p(\mathbf{y})\psi_{k,i}(\mathbf{y}) \leq \sum_{l>\log^2 n}^{\infty} L\Lambda(l)r^l l \leq \kappa q^{\log^2 n} = o(1)$$

where $\kappa > 0$ is a constant. Similarly, we have $p(z \in \mathcal{Z}_{\epsilon,n}) = o(n^{-1})$. This means that $|\mathbb{E}[\mathbb{E}_{\tilde{p}_n}[-\log -f_n^*] \mid A_{\epsilon,n}]|p(A_{\epsilon,n}) \underset{n\to\infty}{\longrightarrow} 0$. In addition, it can be shown that $|\mathbb{E}[\mathbb{E}_{\tilde{p}_n}[C_n(f_n^*) \mid A_{\epsilon,n}]|p(A_{\epsilon,n}) \underset{n\to\infty}{\longrightarrow} 0$ using the same proof technique we used above, while relying on the fact that $C_n(f_n^*) \in \mathcal{F}_n$, and therefore $C_n(f_n^*)(\mathbf{y}) \leq sN|\mathbf{y}| \log n$. □

4.4 Sample Complexity Bounds

Equipped with the framework of proper approximations as described above, we now give our main sample complexity results for probabilistic grammars. These results hinge on the convergence of $\sup_{f\in\mathcal{F}_n} |\mathbb{E}_{\tilde{p}_n}[f] - \mathbb{E}_p[f]|$. Indeed, proper approximations replace the use of $\mathcal{F}$ in these convergence results. The rate of this convergence can be fast, if the *covering numbers* for $\mathcal{F}_n$ do not grow too fast.

4.4.1 Covering Numbers and Bounds on Covering Numbers

We next give a brief overview of covering numbers. A cover provides a way to reduce a class of functions to a much smaller (finite, in fact) representative class such that each function in the original class is represented using a function in the smaller class. Let $\mathcal{G}$ be a class of functions. Let $d(f, g)$ be a distance measure between two functions f, g from $\mathcal{G}$. An ϵ-cover is a subset of $\mathcal{G}$, denoted by $\mathcal{G}'$, such that for every $f \in \mathcal{G}$ there exists an $f' \in \mathcal{G}'$ such that $d(f, f') < \epsilon$. The covering number $\mathcal{N}(\epsilon, \mathcal{G}, d)$ is the size of the smallest ϵ-cover of $\mathcal{G}$ for the distance measure d.

We are interested in a specific distance measure which is dependent on the empirical distribution $\tilde{p}_n$ that describes the data $\mathbf{y}_1, ..., \mathbf{y}_n$. Let $f, g \in \mathcal{G}$. We will use:

$$d^{\tilde{p}_n}(f, g) = \mathbb{E}_{\tilde{p}_n}[|f - g|] = \sum_{\mathbf{y}\in D(\mathbf{G})} |f(\mathbf{y}) - g(\mathbf{y})|\, \tilde{p}_n(\mathbf{y}) = \frac{1}{n}\sum_{i=1}^{n} |f(y_i) - g(y_i)|$$

Instead of using $\mathcal{N}(\epsilon, \mathcal{G}, d^{\tilde{p}_n})$ directly, we bound this quantity with $\mathcal{N}(\epsilon, \mathcal{G}) = \sup_{\tilde{p}_n} \mathcal{N}(\epsilon, \mathcal{G}, d^{\tilde{p}_n})$, where we consider all possible samples (yielding $\tilde{p}_n$). The following is the key result regarding the connection between covering numbers and the double-sided convergence of the empirical process $\sup_{f \in \mathcal{F}_n} |\mathbb{E}_{\tilde{p}_n}[f] - \mathbb{E}_p[f]|$ as $n \to \infty$. This result is a general purpose result that has been used frequently to prove the convergence of empirical processes of the type we discuss in this chapter.

Lemma 4.8 *Let $\mathcal{F}_n$ be a permissible class[8] of functions such that for every $f \in \mathcal{F}_n$ we have $\mathbb{E}[|f| \times \mathbb{I}\{|f| \leq K_n\}] \leq \epsilon_{\text{bound}}(n)$. Let $\mathcal{F}_{\text{truncated},n} = \{f \times \mathbb{I}\{f \leq K_n\} \mid f \in \mathcal{F}_m\}$, i.e., the set of functions from $\mathcal{F}_n$ after being truncated by K_n. Then for $\epsilon > 0$ we have,*

$$p\left(\sup_{f \in \mathcal{F}_n} |\mathbb{E}_{\tilde{p}_n}[f] - \mathbb{E}_p[f]| > 2\epsilon\right) \leq 8\mathcal{N}(\epsilon/8, \mathcal{F}_{\text{truncated},n}) \exp\left(-\frac{1}{128} n\epsilon^2/K_n^2\right) + \epsilon_{\text{bound}}(n)/\epsilon$$

provided $n \geq K_n^2/4\epsilon^2$ and $\epsilon_{\text{bound}}(n) < \epsilon$.

Proof See Pollard (1984) (Chapter 2, pages 30–31). See also Appendix A. □

Covering numbers are rather complex combinatorial quantities which are hard to compute directly. Fortunately, they can be bounded using the pseudo-dimension (Anthony and Bartlett, 1999), a generalization of the VC dimension for real functions. In the case of our "binomialized" probabilistic grammars, the pseudo-dimension of $\mathcal{F}_n$ is bounded by N, because we have $\mathcal{F}_n \subseteq \mathcal{F}$, and the functions in $\mathcal{F}$ are linear with N parameters. Hence, $\mathcal{F}_{\text{truncated},n}$ also has pseudo-dimension that is at most N. We have:

Lemma 4.9 *(From Pollard (1984); Haussler (1992).) Let $\mathcal{F}_n$ be the proper approximations for probabilistic grammars, for any $0 < \epsilon < K_n$ we have:*

$$\mathcal{N}(\epsilon, \mathcal{F}_{\text{truncated},n}) < 2\left(\frac{2eK_n}{\epsilon} \log \frac{2eK_n}{\epsilon}\right)^N$$

[8]The "permissible class" requirement is a mild regularity condition regarding measurability that holds for proper approximations. We refer the reader to Pollard (1984) for more details.

4.4.2 Supervised Case

Lemmas 4.8 and 4.9 can be combined to get the following sample complexity result:

Theorem 4.10 *Let* $\mathbf{G}$ *be a grammar. Let* $p \in \mathcal{P}(\alpha, L, r, q, B, \mathbf{G})$ *(section 4.2.1). Let* $\mathcal{F}_n$ *be a proper approximation for the corresponding family of probabilistic grammars. Let* $\mathbf{y}_1, ..., \mathbf{y}_n$ *be a sample of derivations. Then there exists a constant* $\beta(L, q, s, N)$ *and constant* M *such that for any* $0 < \delta < 1$ *and* $0 < \epsilon < K_n$ *and any* $n > M$ *and if*

$$n \geq \max\left\{\frac{128K_n^2}{\epsilon^2}\left(2N\log(16eK_n/\epsilon) + \log\frac{32}{\delta}\right), \frac{\log 4/\delta + \log 1/\epsilon}{\beta(L, q, s, N)}\right\}$$

then we have

$$P\left(\sup_{f\in\mathcal{F}_n} |\mathbb{E}_{\tilde{p}_n}[f] - \mathbb{E}_p[f]| \leq 2\epsilon\right) \geq 1 - \delta$$

where $K_n = sN\log^3 n$.

Proof Sketch $\beta(L, q, s, N)$ *is the constant from Proposition 4.4. The main idea in the proof is to solve for* n *in the following two inequalities (based on Equation 17) while relying on Lemma 4.9:*

$$8\mathcal{N}(\epsilon/8, \mathcal{F}_{\text{truncated},n})\exp\left(-\frac{1}{128}n\epsilon^2/K_n^2\right) \leq \delta/2$$
$$\epsilon_{\text{bound}}(n)/\epsilon \leq \delta/2$$

■

Theorem 4.10 gives little intuition about the number of samples required for accurate estimation of a grammar because it considers the "additive setting:" the empirical risk is within ϵ from the expected risk. More specifically, it is not clear how we should pick ϵ for the log-loss, since the log-loss can obtain arbitrary values.

We turn now to converting the additive bound in Theorem 4.10 to a multiplicative bound. Multiplicative bounds can be more informative than additive bounds when the range of the values that the log-loss can obtain is not known a priori. However, it is important to note that the two views are equivalent, and it is possible to convert a multiplicative bound to an additive bound and vice versa. Let

$\rho \in (0, 1)$ and choose $\epsilon = \rho K_n$. Then, substituting this ϵ in Theorem 4.10, we get that if:

$$n \geq \max \left\{ \frac{128}{\rho^2} \left(2N \log \frac{16e}{\rho} + \log \frac{32}{\delta} \right), \frac{\log 4/\delta + \log 1/\rho}{\beta(L, q, s, N)} \right\}$$

then with probability $1 - \delta$:

$$\sup_{f \in \mathcal{F}_n} \left| 1 - \frac{\mathbb{E}_{\tilde{p}_n}[f]}{\mathbb{E}_p[f]} \right| \leq \frac{\rho \times 2sN \log^3(n)}{H(p)} \tag{4.13}$$

where $H(p)$ is the Shannon entropy of p. This stems from the fact that $\mathbb{E}_p[f] \geq H(p)$ for any f. This means that if we are interested in computing a sample complexity bound such that the ratio between the empirical risk and the expected risk (for log-loss) is close to 1 with high probability, we need to pick ρ such that the righthand side of Equation 4.13 is smaller than the desired accuracy level (between 0 and 1). Note that Equation 4.13 is an oracle inequality—it requires knowing the entropy of p or some upper bound on it.

4.4.3 Unsupervised Case

In the unsupervised setting, we have n *yields* of derivations from the grammar, $x_1, ..., x_n$, and our goal again is to identify grammar parameters $\boldsymbol{\theta}$ from these yields. Our concept classes are now the sets of log marginalized distributions from $\mathcal{F}_n$. For each $f_{\boldsymbol{\theta}} \in \mathcal{F}_n$, we define $f'_{\boldsymbol{\theta}}$ as:

$$f'_{\boldsymbol{\theta}}(\boldsymbol{x}) = -\log \sum_{\mathbf{y} \in D_{\boldsymbol{x}}(\mathbf{G})} \exp(-f_{\boldsymbol{\theta}}(\mathbf{y})) = -\log \sum_{\mathbf{y} \in D_{\boldsymbol{x}}(\mathbf{G})} \exp \left(\sum_{k=1}^{K} \sum_{i=1}^{N_k} \psi_{i,k}(\mathbf{y}) \boldsymbol{\theta}_{i,k} \right)$$

We denote the set of $\{f'_{\boldsymbol{\theta}}\}$ by $\mathcal{F}'_n$. Analogously, we define $\mathcal{F}'$. Note that we also need to define the operator $C'_n(f')$ as a first step towards defining $\mathcal{F}'_n$ as proper approximations (for $\mathcal{F}'$) in the unsupervised setting. Let $f' \in \mathcal{F}'$. Let f be the concept in $\mathcal{F}$ such that $f'(\boldsymbol{x}) = \sum_{\mathbf{y}} f(\boldsymbol{x}, \mathbf{y})$. Then we define $C'_n(f')(\boldsymbol{x}) = \sum_{\mathbf{y}} C_n(f)(\boldsymbol{x}, \mathbf{y})$.

It does not immediately follow that $\mathcal{F}'_n$ is a proper approximation for $\mathcal{F}'$. It is not hard to show that the boundedness property is satisfied with the same K_n and the same form of $\epsilon_{\text{bound}}(n)$ as in Proposition 4.4 (we would have $\epsilon'_{\text{bound}}(m) = m^{-\beta' \log m}$ for some $\beta'(L, q, s, N) = \beta' > 0$). This relies on the property of bounded derivation length of p. See Appendix A, Proposition A.2. The following result shows that we have tightness as well:

Utility Lemma 4.11 *For $a_i, b_i \geq 0$, if $-\log \sum_i a_i + \log \sum_i b_i \geq \epsilon$ then there exists an i such that $-\log a_i + \log b_i \geq \epsilon$.*

Proposition 4.12 *There exists an M such that for any $n > M$ we have:*

$$p\left(\bigcup_{f' \in \mathcal{F}'} \{\boldsymbol{x} \mid C'_n(f')(\boldsymbol{x}) - f'(\boldsymbol{x}) \geq \epsilon_{\text{tail}}(n)\}\right) \leq \epsilon_{\text{tail}}(n)$$

for $\epsilon_{\text{tail}}(n) = \dfrac{N \log^2 n}{n^s - 1}$ and the operator $C'_n(f)$ as defined above.

Proof Sketch *From Utility Lemma 4.11 we have:*

$$p\left(\bigcup_{f' \in \mathcal{F}'} \{\boldsymbol{x} \mid C'_n(f')(\boldsymbol{x}) - f'(\boldsymbol{x}) \geq \epsilon_{\text{tail}}(n)\}\right) \leq p\left(\bigcup_{f \in \mathcal{F}} \{\boldsymbol{x} \mid \exists \mathbf{y} C_n(f)(\mathbf{y}) - f(\mathbf{y}) \geq \epsilon_{\text{tail}}(n)\}\right)$$

Define $\mathcal{X}(n)$ to be all $\boldsymbol{x}$ such that there exists a $\mathbf{y}$ with $s(\mathbf{y}) = \boldsymbol{x}$ and $|\mathbf{y}| \geq \log^2 n$. From the proof of Proposition 4.5 and the requirements on p, we know that there exists an $\alpha \geq 1$ such that

$$\begin{aligned} p\left(\textstyle\bigcup_{f \in \mathcal{F}} \{\boldsymbol{x} \mid \exists \mathbf{y} \text{ s.t. } C_n(f)(\mathbf{y}) - f(\mathbf{y}) \geq \epsilon_{\text{tail}}(n)\}\right) &\leq \sum_{x \in \mathcal{X}(n)} p(\boldsymbol{x}) \\ \leq \sum_{x : |\boldsymbol{x}| \geq \log^2 n/\alpha} p(\boldsymbol{x}) &\leq \sum_{k = \lfloor \log^2 n/\alpha \rfloor}^{\infty} L\Lambda(k) r^k \leq \epsilon_{\text{tail}}(n) \end{aligned}$$

where the last inequality happens for some n larger than a fixed M. ■

Computing either the covering number or the pseudo-dimension of $\mathcal{F}'_n$ is a hard task, because the function in the classes includes the "log-sum-exp." Dasgupta (1997) overcomes this problem for Bayesian networks with fixed structure by giving a bound on the covering number for (his respective) $\mathcal{F}'$ which depends on the covering number of $\mathcal{F}$.

Unfortunately, we cannot fully adopt this approach, because the derivations of a probabilistic grammar can be arbitrarily large. Instead, we present the following Proposition, which is based on the "Hidden Variable Rule" from Dasgupta (1997). This proposition showed that the covering number of $\mathcal{F}'$ (or more accurately, its bounded approximations) can be bounded in terms of the covering number of the bounded approximations of $\mathcal{F}$, and the constants which control the underlying distribution p mentioned in Section 4.2.

Utility Lemma 4.13 *For any two positive-valued sequences* $(a_1, ..., a_n)$ *and* $(b_1, ..., b_n)$ *we have that* $\sum_i |\log a_i/b_i| \geq |\log (\sum a_i / \sum b_i)|$.

Proposition 4.14 (Hidden Variable Rule for Probabilistic Grammars) *Let* $m = \dfrac{\log \dfrac{4K_n}{\epsilon(1-q)}}{\log \dfrac{1}{q}}$. *Then,* $\mathcal{N}(\epsilon, \mathcal{F}'_{\text{truncated},n}) \leq \mathcal{N}\left(\dfrac{\epsilon}{2\Lambda(m)}, \mathcal{F}_{\text{truncated},n}\right)$.

Proof Let $\mathcal{Z}(m) = \{\mathbf{y} \mid |\mathbf{y}| \leq m\}$ be the subset of derivations of length shorter than m. Consider $f, f_0 \in \mathcal{F}_{\text{truncated},n}$. Let f' and f'_0 be the corresponding functions in $\mathcal{F}'_{\text{truncated},n}$. Then, for any distribution p:

$$
\begin{aligned}
d^p(f', f'_0) &= \sum_{\boldsymbol{x}} |f'(\boldsymbol{x}) - f'_0(\boldsymbol{x})|\, p(\boldsymbol{x}) \leq \sum_{\boldsymbol{x}} \sum_{z} |f(\boldsymbol{x}, \mathbf{y}) - f_0(\boldsymbol{x}, \mathbf{y})|\, p(\boldsymbol{x}) \\
&= \sum_{\boldsymbol{x}} \sum_{z \in \mathcal{Z}(m)} |f(\boldsymbol{x}, \mathbf{y}) - f_0(\boldsymbol{x}, \mathbf{y})|\, p(\boldsymbol{x}) + \sum_{\boldsymbol{x}} \sum_{z \notin \mathcal{Z}(m)} |f(\boldsymbol{x}, \mathbf{y}) - f_0(\boldsymbol{x}, \mathbf{y})|\, p(\boldsymbol{x}) \\
&\leq \sum_{\boldsymbol{x}} \sum_{z \in \mathcal{Z}(m)} |f(\boldsymbol{x}, \mathbf{y}) - f_0(\boldsymbol{x}, \mathbf{y})|\, p(\boldsymbol{x}) + \sum_{\boldsymbol{x}} \sum_{z \notin \mathcal{Z}(m)} 2K_n p(\boldsymbol{x}) \qquad (4.14) \\
&\leq \sum_{\boldsymbol{x}} \sum_{z \in \mathcal{Z}(m)} |f(\boldsymbol{x}, \mathbf{y}) - f_0(\boldsymbol{x}, \mathbf{y})|\, p(\boldsymbol{x}) + 2K_n \sum_{\boldsymbol{x}:\, |\boldsymbol{x}| \geq m} |D_x(\mathbf{G})| p(\boldsymbol{x}) \\
&\leq \sum_{\boldsymbol{x}} \sum_{z \in \mathcal{Z}(m)} |f(\boldsymbol{x}, \mathbf{y}) - f_0(\boldsymbol{x}, \mathbf{y})|\, p(\boldsymbol{x}) + 2K_n \sum_{k=m}^{\infty} \Lambda^2(k) r^k \\
&\leq d^{p'}(f, f_0) |\mathcal{Z}(m)| + 2K_n \frac{q^m}{1-q}
\end{aligned}
$$

where $p'(\boldsymbol{x}, \mathbf{y})$ is a probability distribution that uniformly divides the probability mass $p(\boldsymbol{x})$ across all derivations for the specific $\boldsymbol{x}$, i.e.:

$$p'(\boldsymbol{x}, \mathbf{y}) = \frac{p(\boldsymbol{x})}{|D_{\boldsymbol{x}}(\mathbf{G})|}$$

The inequality in Equation 4.14 stems from Utility Lemma 4.13.

Set m to be the quantity that appears in the proposition to get the necessary result (f' and f are arbitrary functions in $\mathcal{F}'_{\text{truncated},n}$ and $\mathcal{F}_{\text{truncated},n}$ respectively. Then consider f'_0 and f_0 to be functions from the respective covers.) □

For the unsupervised case, then, we get the following sample complexity result:

Theorem 4.15 *Let* $\mathbf{G}$ *be a grammar. Let* $\mathcal{F}'_n$ *be a proper approximation for the corresponding family of probabilistic grammars. Let* $p(\boldsymbol{x}, \mathbf{y})$ *be a distribution over derivations which satisfies the requirements in section 4.2.1. Let* $x_1, ..., x_n$ *be a sample of strings from* $p(\boldsymbol{x})$*. Then there exists a constant* $\beta'(L, q, s, N)$ *and constant* M *such that for any* $0 < \delta < 1$ *and* $0 < \epsilon < K_n$ *and any* $n > M$ *and if*

$$n \geq \max\left\{\frac{128K_n^2}{\epsilon^2}\left(2N\log\left(\frac{32eK_n\Lambda(m)}{\epsilon}\right) + \log\frac{32}{\delta}\right), \frac{\log 4/\delta + \log 1/\epsilon}{\beta'(L, q, s, N)}\right\} \tag{4.15}$$

where $m = \dfrac{\log\dfrac{4K_n}{\epsilon(1-q)}}{\log\dfrac{1}{q}}$*, we have that*

$$p\left(\sup_{f\in\mathcal{F}'_n} |\mathbb{E}_{\tilde{p}_n}[f] - \mathbb{E}_p[f]| \leq 2\epsilon\right) \geq 1 - \delta$$

where $K_n = sN\log^3 n$*.*

Theorem 4.15 states that the number of samples we require in order to accurately estimate a probabilistic grammar from unparsed strings depends on the level of ambiguity in the grammar, represented as $\Lambda(m)$. We note that this dependence is polynomial, and we consider this a positive result for unsupervised learning of grammars. More specifically, if Λ is an exponential function (such as the case with PCFGs), when compared to the supervised learning, there is an extra multiplicative factor in the sample complexity in the unsupervised setting that behaves like $\mathcal{O}(\log\log\dfrac{K_n}{\epsilon})$.

We note that Equation 4.15 can again be reduced to a multiplicative case, similarly to the way we described it for the supervised case. Setting $\epsilon = \rho K_n$ ($\rho \in (0, 1)$), we get the following requirement on n:

$$n \geq \max\left\{\frac{128}{\rho^2}\left(2N\log\left(\frac{32e \times t(\rho)}{\rho}\right) + \log\frac{32}{\delta}\right), \frac{\log 4/\delta + \log 1/\epsilon}{\beta'(L, q, s, N)}\right\}$$

where $t(\rho) = \dfrac{\log\dfrac{4}{\rho(1-q)}}{\log\dfrac{1}{q}}$.

4.5 Algorithms for Empirical Risk Minimization

We turn now to describing algorithms and their properties for minimizing empirical risk using the framework described in section 4.3.

4.5.1 Supervised Case

ERM with proper approximations leads to simple algorithms for estimating the probabilities of a probabilistic grammar in the supervised setting. Given an $\epsilon > 0$ and a $\delta > 0$, we draw n examples according to Theorem 4.10. We then set $\gamma = n^{-s}$. To minimize the log-loss with respect to these n examples, we use the proper approximation $\mathcal{F}_n$.

Note that the value of the empirical log-loss for a probabilistic grammar parametrized by $\boldsymbol{\theta}$ is:

$$\begin{aligned}
\mathbb{E}_{\tilde{p}_n}[-\log h(\boldsymbol{x}, \mathbf{y} \mid \boldsymbol{\theta})] &= -\sum_{\boldsymbol{x},\mathbf{y}} \tilde{p}_n(\boldsymbol{x}, \mathbf{y}) \log h(\boldsymbol{x}, \mathbf{y} \mid \boldsymbol{\theta}) \\
&= -\sum_{\boldsymbol{x},\mathbf{y}} \tilde{p}_n(\boldsymbol{x}, \mathbf{y}) \sum_{k=1}^{K} \sum_{i=1}^{N_k} \psi_{k,i}(\boldsymbol{x}, \mathbf{y}) \log(\theta_{k,i}) \\
&= -\sum_{k=1}^{K} \sum_{i=1}^{N_k} \log(\theta_{k,i}) \mathbb{E}_{\tilde{p}_n}[\psi_{k,i}]
\end{aligned}$$

Since we make the assumption that $\deg(\mathbf{G}) \leq 2$ (section 4.2.2), we have:

$$\mathbb{E}_{\tilde{p}_n}[-\log h(\boldsymbol{x}, \mathbf{y} \mid \boldsymbol{\theta})] = -\sum_{k=1}^{K} (\log(\theta_{k,1}) \mathbb{E}_{\tilde{p}_n}[\psi_{k,1}] + \log(1 - \theta_{k,1}) \mathbb{E}_{\tilde{p}_n}[\psi_{k,2}]) \quad (4.16)$$

To minimize the log-loss with respect to $\mathcal{F}_n$, we need to minimize Equation 4.16 under the constraint that $\gamma \leq \theta_{k,i} \leq 1 - \gamma$ and $\theta_{k_1} + \theta_{k,2} = 1$. It can be shown that the solution for this optimization problem is:

$$\theta_{k,i} = \min\left\{1 - \gamma, \max\left\{\gamma, \left(\sum_{j=1}^{n} \hat{\psi}_{j,k,i}\right) / \left(\sum_{j=1}^{n} \sum_{i'=1}^{2} \hat{\psi}_{j,k,i'}\right)\right\}\right\} \quad (4.17)$$

where $\hat{\psi}_{j,k,i}$ is the number of times that $\psi_{k,i}$ fires in example j. (We include a full derivation of this result in Appendix B.) The interpretation of Equation 4.17

is simple: we count the number of times a rule appears in the samples and then normalize this value by the total number of times rules associated with the same multinomial appear in the samples. This frequency count is the maximum likelihood solution with respect to the full hypothesis class $\mathcal{H}$ (Corazza and Satta, 2006). Since we constrain ourselves to obtain a value away from 0 or 1 by a margin of γ, we need to truncate this solution, as done in Equation 4.17.

This truncation to a margin γ can be thought of as a smoothing factor that enables us to compute sample complexity bounds. We explore this connection to smoothing with a Dirichlet prior in a MAP Bayesian setting in section 4.6.2.

4.5.2 Unsupervised Case

Similarly to the supervised case, minimizing the empirical log-loss in the unsupervised setting requires minimizing (with respect to $\boldsymbol{\theta}$):

$$\mathbb{E}_{\tilde{p}_n}[-\log h(\boldsymbol{x} \mid \boldsymbol{\theta})] = -\sum_{\boldsymbol{x}} \tilde{p}_n(\boldsymbol{x}) \log \sum_{z} h(\boldsymbol{x}, \mathbf{y} \mid \boldsymbol{\theta}) \tag{4.18}$$

with the constraint that $\gamma \leq \theta_{k,i} \leq 1 - \gamma$ (i.e. $\boldsymbol{\theta} \in \Theta(\gamma)$) where $\gamma = n^{-s}$. This is done after drawing n examples according to Theorem 4.15.

Hardness of ERM with Proper Approximations

It turns out that minimizing Equation 4.18 under the specified constraints is actually an NP-hard problem when $\mathbf{G}$ is a PCFG. This result follows using a similar proof to the one in Chapter 3 for the hardness of Viterbi training and maximizing log-likelihood for PCFGs. We turn to describe the modification required to the proofs in Chapter 3 to the case of having an arbitrary γ margin constraint.

We first begin by by stating the following decision problem:

Problem 4.1 (Unsupervised Minimization of the Log-Loss with Margin) Input: *A binarized context-free grammar* $\mathbf{G}$*, a set of sentences* $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$*, a value* $\gamma \in [0, \frac{1}{2})$ *and a value* $\alpha \in [0, 1]$.
Output: *1 if there exists* $\boldsymbol{\theta} \in \Theta(\gamma)$ *(and hence,* $h \in \mathcal{H}(\mathbf{G})$*) such that*

$$-\sum_{\boldsymbol{x}} \tilde{p}_n(\boldsymbol{x}) \log \sum_{z} h(\boldsymbol{x}, \mathbf{y} \mid \boldsymbol{\theta}) \leq -\log(\alpha) \tag{4.19}$$

and 0 otherwise.

We will show the hardness result both when γ is not restricted at all as well as when we allow $\gamma > 0$. Given an instance ϕ of the 3-SAT problem, we create a grammar $\mathbf{G}$ and a string s_ϕ as described in Section 3.2.

Theorem 4.16 *Problem 4.1 is NP-hard when either requiring $\gamma > 0$ or when fixing $\gamma = 0$.*

Proof We first describe the reduction for the case of $\gamma = 0$. In Problem 4.1, set $\gamma = 0$, $\alpha = 1$, $\mathbf{G} = \mathbf{G}_\phi$, $\gamma = 0$ and $\boldsymbol{x}_1 = \boldsymbol{s}_\phi$. If ϕ is satisfiable, then the left side of Equation 4.19 can get value 0, by setting the rule probabilities according to Lemma 3.2, hence we would return 1 as the result of running Problem 4.1.

If ϕ is unsatisfiable, then we would still get value 0 only if $L(\mathbf{G}) = \{s_\phi\}$. If $\mathbf{G}_\phi$ generates a single derivation for $(10)^{3m}$, then we actually do have a satisfying assignment from Lemma 3.1. Otherwise (more than a single derivation), the optimal $\boldsymbol{\theta}$ would have to give fractional probabilities to rules of the form $V_{Y_r} \to \{0, 1\}$ (or $V_{\bar{Y}_r} \to \{0, 1\}$). In that case, it is no longer true that $(10)^{3m}$ is the only generated sentence, and this is a contradiction to getting value 0 for Problem 4.1.

We next show that Problem 4.1 is NP-hard even if we require $\gamma > 0$. Let $\gamma < \frac{1}{20m}$. Set $\alpha = \gamma$, and the rest of the inputs to Problem 4.1 the same just as before. Assume that ϕ is satisfiable. Let $\boldsymbol{\theta}$ be the rule probabilities from Equation 3.2 after being shifted with a margin of γ. Then, since there is a derivation that uses only rules that have probability $1 - \gamma$, we have:

$$\begin{aligned} h(\boldsymbol{x}_1 \mid T(\boldsymbol{\theta}, \gamma), \mathbf{G}_\phi) &= \sum_{\mathbf{y}} p(\boldsymbol{x}_1, \mathbf{y} \mid T(\boldsymbol{\theta}, \gamma), \mathbf{G}_\phi) \\ &\geq (1 - \gamma)^{10m} \\ &> \alpha \end{aligned}$$

because the size of the parse tree for $(10)^{3m}$ is at most $10m$ (using the binarized $\mathbf{G}_\phi$) and assuming $\alpha = \gamma < (1 - \gamma)^{10m}$. This inequality indeed holds whenever $\gamma < \dfrac{1}{20m}$. Therefore, we have $-\log h(\boldsymbol{x}_1 \mid \boldsymbol{\theta}) > -\log \alpha$. Problem 4.1 would return 0 in this case.

Now, assume that ϕ is not satisfiable. That means that any parse tree for the string $(10)^{3m}$ would have to contain two different rules headed by the same nonterminal. This means that:

$$\begin{aligned} h(\boldsymbol{x}_1 \mid T(\boldsymbol{\theta}, \gamma), \mathbf{G}_\phi) &= \sum_{\mathbf{y}} p(\boldsymbol{x}_1, \mathbf{y} \mid T(\boldsymbol{\theta}, \gamma), \mathbf{G}_\phi) \\ &\leq \gamma \end{aligned}$$

and therefore $-\log h(\boldsymbol{x}_1 \mid T(\boldsymbol{\theta}, \gamma)) \leq -\log \alpha$, and Problem 4.1 would return 1. □

An Expectation-Maximization Algorithm

Instead of solving the optimization problem implied by Equation 4.16, we propose a rather simple modification to the expectation-maximization algorithm (Dempster et al., 1977) to approximate the optimal solution—this algorithm finds a local maximum for the maximum likelihood problem using proper approximations. The modified algorithm is given in Algorithm 1.

The modification from the usual expectation-maximization algorithm is done in the M-step: instead of using the expected value of the sufficient statistics by counting and normalizing, we truncate the values by γ. It can be shown that if $\boldsymbol{\theta}^{(0)} \in \Theta(\gamma)$, then the likelihood is guaranteed to increase (and hence, the log-loss is guaranteed to decrease) after each iteration of the algorithm.

The reason for this likelihood increase stems from the fact that the M-step solves the optimization problem of minimizing the log-loss (with respect to $\boldsymbol{\theta} \in \Theta(\gamma)$) when the posterior calculate at the E-step as the base distribution is used. This means that the M-step minimizes (in iteration t): $\mathbb{E}_r[-\log h(\boldsymbol{x}, \mathbf{y} \mid \boldsymbol{\theta}^{(t)})]$ where the expectation is taken with respect to the distribution $r(\boldsymbol{x}, \mathbf{y}) = \tilde{p}_n(\boldsymbol{x}) p(\mathbf{y} \mid \boldsymbol{x}, \boldsymbol{\theta}^{(t-1)})$. With this notion in mind, the likelihood increase after each iteration follows from principles similar to those described in Bishop (2006) for the EM algorithm.

4.6 Discussion

Our framework can be specialized to improve the two main criteria which have a trade-off: the tightness of the proper approximation and the sample complexity. For example, we can improve the tightness of our proper approximations by taking a subsequence of $\mathcal{F}_n$. However, this will make the sample complexity bound degrade, because K_n will grow faster. Table 4.2 shows the trade-offs between parameters in our model and the effectiveness of learning.

We note that the sample complexity bounds that we give in this chapter give insight about the asymptotic behavior of grammar estimation, but are not necessarily sufficiently tight to be used in practice. It still remains an open problem to obtain sample complexity bounds which are sufficiently tight in this respect. For

Algorithm 1: Expectation-Maximization Algorithm with Proper Approximations.

Input: grammar $\mathbf{G}$ in binary normal form, initial parameters $\boldsymbol{\theta}^{(0)}$, $\epsilon > 0$, $\delta > 0$, $s > 1$

Output: learned parameters $\boldsymbol{\theta}$

draw $\boldsymbol{x} = \langle \boldsymbol{x}_1, ..., \boldsymbol{x}_n \rangle$ from p following Theorem 4.15 ;

$t \leftarrow 1$;

$\gamma \leftarrow n^{-s}$;

repeat

// $\mathbb{E}_{\boldsymbol{\theta}^{(t-1)}}[\psi_{k,i}(\mathbf{y}) \mid \boldsymbol{x}_j]$ denotes the expected counts of event i in multinomial k under the distribution $\tilde{p}_n(\boldsymbol{x})p(\mathbf{y} \mid \boldsymbol{x}, \boldsymbol{\theta}^{(t-1)})$

Compute for each training example $j \in \{1, \ldots, n\}$, for each event $i \in \{1, 2\}$ in each multinomial $k \in \{1, \ldots, K\}$:

$\hat{\psi}_{j,k,i} \leftarrow \mathbb{E}_{\boldsymbol{\theta}^{(t-1)}}[\psi_{k,i}(\mathbf{y}) \mid \boldsymbol{x}_j]$;

Set $\boldsymbol{\theta}^{(t)}_{i,k} = \min\{1 - \gamma, \max\{\gamma, \left(\sum_{j=1}^{n} \hat{\psi}_{j,k,i}\right) / \left(\sum_{j=1}^{n} \sum_{i'=1}^{2} \hat{\psi}_{j,k,i'}\right)\}\}$;

$t \leftarrow t + 1$;

until *convergence*;

return $\boldsymbol{\theta}^{(t)}$

criterion	as K_n increases ...	as s increases ...
tightness of proper approximation	improves	improves
sample complexity bound	degrades	degrades

Table 4.2: Trade-off between quantities in our learning model and effectiveness of different criteria. K_n is the constant that satisfies the boundedness property (Theorems 4.10 and 4.15) and s is a fixed constant larger than 1 (section 4.3.1).

a discussion about the connection of grammar learning in theory and practice, we refer the reader to Clark and Lappin (2010). See also Section 7.1.3.

4.6.1 Tsybakov Noise

In this chapter, we chose to introduce assumptions about distributions that generate natural language data. The choice of these assumptions was motivated by observations about properties shared among treebanks. The main consequence of making these assumptions is bounding the amount of *noise* in the distribution, i.e., the amount of variation in probabilities across labels given a fixed input.

There are other ways to restrict the noise in a distribution. One condition for such noise restriction, which has received considerable recent attention in the statistical literature is the Tsybakov noise condition (Tsybakov, 2004; Koltchinskii, 2006). Showing that a distribution satisfies the Tsybakov noise condition enables the use of techniques (e.g., from Koltchinskii 2006) for deriving distribution-dependent sample complexity bounds that depend on the parameters of the noise. It is therefore of interest to see whether Tsybakov noise holds under the assumptions presented in section 4.2.1. We show that this is not the case, and that Tsybakov noise is too permissive. In fact, we show that p can be a probabilistic grammar itself (and hence, satisfy the assumptions in section 4.2.1), and still not satisfy the Tsybakov noise conditions.

Tsybakov noise was originally introduced for classification problems (Tsybakov, 2004), and was later extended to more general settings, such as the one we are facing in this chapter (Koltchinskii, 2006). We next explain the definition of Tsybakov noise in our context.

Let $C > 0$ and $\kappa \geq 1$. We say that a distribution $p(\boldsymbol{x}, \mathbf{y})$ satisfies the (C, κ) Tsybakov noise condition if for any $\epsilon > 0$ and $h, g \in \mathcal{H}$ such that $h, g \in \{h' \mid \mathcal{E}_p(h', \mathcal{H}) \leq \epsilon\}$, we have:

$$\mathrm{dist}(g, h) \triangleq \sqrt{\mathbb{E}_p\left[\left(\frac{\log g}{\log h}\right)^2\right]} \leq C\epsilon^{1/k} \tag{4.20}$$

This interpretation of Tsybakov noise implies that the diameter of the set of functions from the concept class that has small excess risk should shrink to 0 at the rate in Equation 4.20. Distribution-dependent bounds from Koltchinskii (2006) are monotone with respect to the diameter of this set of functions, and therefore, demonstrating that it goes to 0 enables sharper derivations of to derive sharper sample complexity bounds.

We turn now to illustrating that the Tsybakov condition does not hold for probabilistic grammars in most cases. Let $\mathbf{G}$ be a probabilistic grammar. Define $A = A_{\mathbf{G}}(\boldsymbol{\theta})$ as a matrix such that:

$$\left(A_{\mathbf{G}}(\boldsymbol{\theta})\right)_{(k,i),(k',i')} \triangleq \frac{\mathbb{E}[\psi_{k,i} \times \psi_{k',i'}]}{\mathbb{E}[\psi_{k,i}]\mathbb{E}[\psi_{k',i'}]}$$

In Appendix C we show that $A_{\mathbf{G}}(\boldsymbol{\theta})$ is positive semi-definite for any choice of $\boldsymbol{\theta}$.

Theorem 4.17 *Let $\mathbf{G}$ be a grammar with $K \geq 2$ and degree 2. Assume that p is $\langle \mathbf{G}, \boldsymbol{\theta}^* \rangle$ for some $\boldsymbol{\theta}^*$, such that $\theta^*_{1,1} = \theta^*_{2,1} = \mu$ and that $c_1 \leq c_2$. If $A_{\mathbf{G}}(\boldsymbol{\theta}^*)$ is positive definite, then p does not satisfy the Tsybakov noise condition for any (C, κ), where $C > 0$ and $\kappa \geq 1$.*

Proof See Appendix C. □

The main intuition behind the proof is that given a probabilistic grammar p, we can construct an hypothesis h such that the KL divergence between p and h is small, but $\mathrm{dist}(p, h)$ is lower bounded and is not close to 0.

We conclude that probabilistic grammars, as generative distributions of data, do not generally satisfy the Tsybakov noise condition. This motivates an alternative choice of assumptions that could lead to better understanding of rates of convergences and bounds on the excess risk. Section 4.2.1 stated such assumptions which were also justified empirically.

4.6.2 Comparison to Dirichlet Maximum *A Posteriori* Solutions

The transformation $T(\boldsymbol{\theta}, \gamma)$ from section 4.3.1 can be thought of as a *smoother* for the probabilities $\boldsymbol{\theta}$: it ensures that the probability of each rule is at least γ (and as a result, the probabilities of all rules cannot exceed $1 - \gamma$). Adding pseudo-counts to frequency counts is also a common way to smooth probabilities in models based on multinomial distributions, including probabilistic grammars (Manning and Schütze, 1999). These pseudo-counts can be framed as a maximum *a posteriori* (MAP) alternative to the maximum likelihood problem, with the choice of Bayesian prior over the parameters in the form of a Dirichlet distribution. In comparison to our framework, with (symmetric) Dirichlet smoothing, instead of

truncating the probabilities with a margin γ, we would set the probability of each rule (in the supervised setting) to:

$$\hat{\theta}_{k,i} = \frac{\sum_{j=1}^{n} \hat{\psi}_{j,k,i} + \alpha}{\sum_{j=1}^{n} \hat{\psi}_{j,k,1} + \sum_{j=1}^{n} \hat{\psi}_{j,k,2} + 2\alpha} \tag{4.21}$$

for $i = 1, 2$, where $\hat{\psi}_{k,i}$ are the counts in the data of event i in multinomial k for example j. Dirichlet smoothing can be formulated as the result of adding a symmetric Dirichlet prior over the parameters $\theta_{k,i}$ with hyperparameter α. Then, Equation 4.21 is the mode of the posterior after observing $\hat{\psi}_{k,i}$ appearances of event i in multinomial k.

The effect of Dirichlet smoothing becomes weaker as we have more samples, because the frequency counts $\hat{\psi}_{j,k,i}$ become dominant in both the numerator and the denominator when there is more data. In this sense, the prior's effect on learning diminishes as we use more data. A similar effect occurs in our framework: $\gamma = n^{-s}$ where n is the number of samples—the more samples we have, the more we trust the counts in the data to be reliable. There is a subtle difference, however. With the Dirichlet MAP solution, the smoothing is less dominant only if the counts of the features are large, regardless of the number of samples we have. With our framework, smoothing depends *only* on the number of samples we have. These two scenarios are related, of course: the more samples we have, the more likely it is that the counts of the events will grow large.

4.6.3 Other Derivations of Sample Complexity Bounds

In this section, we make some notes about other possible solutions to the problem of deriving sample complexity bounds for probabilistic grammars.

Using Talagrand's Inequality Our bounds are based on VC theory together with classical results for empirical processes (Pollard, 1984). There have been some recent developments to the derivation of rates of convergence in statistical learning theory (Massart, 2000; Bartlett et al., 2005; Koltchinskii, 2006), most prominently through the use of Talagrand's inequality (Talagrand, 1994), which is a concentration of measure inequality, in the spirit of Lemma 4.8.

The bounds achieved with Talagrand's inequality are also distribution-dependent, and are based on the diameter of the ϵ-minimal set—the set of hypotheses which have an excess risk smaller than ϵ. We saw in section 4.6.1 that the diameter of

the ϵ-minimal set does not follow the Tsybakov noise condition, but it is perhaps possible to find meaningful bounds for it, in which case, we may be able to get tighter bounds using Talagrand's inequality. We note that it may be possible to obtain *data-dependent* bounds for the diameter of the ϵ-minimal set, following Koltchinskii (2006), by calculating the diameter of the ϵ-minimal set using $\tilde{p}_n$.

Simpler Bounds for the Supervised Case As noted in section 4.5.1, minimizing empirical risk with the log-loss leads to a simple frequency count for calculating the estimated parameters of the grammar. In Corazza and Satta (2006), it has been also noted that to minimize the expected risk, it is necessary to set the parameters of the grammar to the normalized *expected* count of the features.

This means that we can get bounds on the deviation of a certain parameter from the optimal parameter by applying modifications to rather simple inequalities such as Hoeffding's inequality, which determines the probability of the average of a set of i.i.d. random variables deviating from its mean. The modification would require us to split the event space into two cases: one in which the count of some features is larger than some fixed value (which will happen with small probability because of the bounded expectation of features), and one in which they are all smaller than that fixed value. Handling these two cases separately is necessary because Hoeffding's inequality requires that the count of the rules is bounded.

The bound on the deviation from the mean of the parameters (the true probability) can potentially lead to a bound on the excess risk in the supervised case. However, this formulation of the problem would not generalize to the unsupervised case, where the empirical risk minimization does not amount to simple frequency count.

4.7 Summary

We presented a framework for performing empirical risk minimization for probabilistic grammars, in which sample complexity bounds, for the supervised case and the unsupervised case, can be derived. Our framework is based on the idea of bounded approximations used in the past to derive sample complexity bounds for graphical models.

Our framework required assumptions about the probability distribution that generates sentences or derivations in the language of the given grammar. These assumptions were tested using corpora, and found to fit the data well.

We also discussed algorithms that can be used for minimizing empirical risk in our framework, given enough samples. We showed that directly trying to minimizing empirical risk in the unsupervised case is NP-hard, and suggested an approximation based on an expectation-maximization algorithm.

Chapter 5

Soft Parameter-Tying in Estimation of Probabilistic Grammars

As suggested in Section 2.1, the parameter space for a typical probabilistic grammar is decomposable into a collection of multinomials. Multinomial distributions, in their simplest form, are defined on an unordered finite sample space: each event is allocated its own probability mass. When we specify such a distribution, we do not explicitly specify relationships between the various events of the multinomial (in the form of a covariance structure or any other form).

Still, it is conceivable that such relationships exist with probabilistic grammars. When we consider probabilistic grammars for natural languages, especially those over words or word classes like parts of speech, we *do* expect to see covariance structure. Intuitively, the probability of a particular word or word class having singular nouns as arguments is likely tied to the probability of the same word having *plural* nouns as arguments. Words that tend to attach to one type of parent are expected to tend to attach to similar parents. This is a large part of the empirical motivation for syntactic theories that make use of part of speech and phrase categories (Kroeger, 2005).

In this chapter, we capitalize on this idea, and propose an estimation method for probabilistic grammars that softly ties between the parameters of the grammar using an explicit covariance structure. The estimation method has the following advantages:

1. It permits softly tying parameters of the grammar, while estimating the strengths that should exist between the various weights of the multinomial distributions in a probabilistic grammar.

2. It permits one to encode prior knowledge into the learning process. In Chapter 8, we will see that this prior knowledge comes in the form of a reduction of fine-grained part-of-speech tags to coarse part-of-speech tags.

3. It has a Bayesian interpretation of placing a prior over the parameters of a probabilistic grammar.

Since the estimation procedure we propose has a Bayesian interpretation, we begin by explaining how one would apply the Bayesian approach to probabilistic grammars. We first present the current approach to Bayesian learning of probabilistic grammars, and then suggest our alternative approach. We then explain how our approach yields the proposed estimation procedure.

Some of the work in this chapter has been described in Cohen et al. (2008), Cohen and Smith (2009), and Cohen and Smith (2010a).

5.1 The Bayesian Setting

As mentioned above, the goal of this chapter is to introduce a framework for softly-tying between parameters of a probabilistic grammar composed of multinomials. But how do we represent explicit connections between the various elements of a multinomial during learning, when the events in a multinomial are separate?

One way to do it is by introducing a *Bayesian* prior over the grammar parameters, such that there is a bias in the prior which dictates that simulating from the prior takes into consideration connections between grammar parameters. This prior defines a distribution over $\boldsymbol{\theta}$. It is this prior that we update in the Bayesian setting after observing data, to get the *posterior*, a new distribution over the parameters which takes into account both the information in the prior and the information appearing in the data. As specified before, the prior is defined as a distribution $p(\boldsymbol{\theta} \mid \boldsymbol{\alpha}, \mathbf{G})$ where $\boldsymbol{\alpha}$ are the hyperparameters controlling it. The posterior that we find, in turn, will be a new distribution $p(\boldsymbol{\theta} \mid \boldsymbol{\alpha}', \mathbf{G})$, where $\boldsymbol{\alpha}'$ are the new hyperparameters for the distribution over the parameters.

When we consider the probability of the data that we observe in our Bayesian model, we treat $\boldsymbol{\theta}$ as a hidden variable. It will therefore be integrated out in defining the probability of the data:

$$p(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_M \mid \boldsymbol{\alpha}, \mathbf{G}) = \int p(\boldsymbol{\theta} \mid \boldsymbol{\alpha}, \mathbf{G}) \prod_{m=1}^{M} \sum_{\mathbf{y}} p(\boldsymbol{x}_m, \mathbf{y} \mid \boldsymbol{\theta}, \mathbf{G}) \, d\boldsymbol{\theta}. \quad (5.1)$$

In this setting, it is $\boldsymbol{\alpha}$, the distribution over grammar parameters, that softly ties between the various parameters of the grammar, and it will be $\boldsymbol{\alpha}$ that we estimate when we perform learning.

We consider two alternative variations on the Bayesian idea, illustrated in Figure 5.1. In the first, called "model I," the grammar's probabilities $\boldsymbol{\theta}$ are drawn randomly once per sentence for the whole corpus $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_M$. In "model II," the grammar parameters are drawn *once* for the whole corpus.

Conceptually, both options have advantages and disadvantages when modeling natural language. Drawing $\boldsymbol{\theta}$ for each derivation permits more flexibility across derivations, perhaps allowing the learner to capture variation across the corpus (even if not systematically, as the grammars are drawn IID), arising from different authors, for example. Generating $\boldsymbol{\theta}$ only once suggests we need to do inference in a smaller space: we only need to find the posterior over a single $\boldsymbol{\theta}$, perhaps leading to better generalization. We will consider both forms in our experiments (Section 8.4).

Model I:

For $m \in \{1, \ldots, M\}$:

1. Draw $\boldsymbol{\theta}_m$ from the prior $p(\boldsymbol{\theta} \mid \mathbf{G}, \ldots)$.

2. Draw $(\boldsymbol{x}_m, \mathbf{y}_m)$ from $p(\boldsymbol{x}_m, \mathbf{y}_m \mid \boldsymbol{\theta}_m, \mathbf{G})$.

Model II:

1. Draw $\boldsymbol{\theta}$ from the prior $p(\boldsymbol{\theta} \mid \mathbf{G}, \ldots)$.

2. For $m \in \{1, \ldots, M\}$:

   Draw $(\boldsymbol{x}_m, \mathbf{y}_m)$ from $p(\boldsymbol{x}_m, \mathbf{y}_m \mid \boldsymbol{\theta}, \mathbf{G})$.

Figure 5.1: Two variations on Bayesian modeling of probabilistic grammars.

5.1.1 Prior Distributions

A question that remains is what prior should we choose in order to represent ties between the various grammar parameters. In their early work about conjugate priors, Raiffa and Schlaifer (1961) set desiderata for prior distributions in parametric models. These desiderata include: (i) analytical tractability—the posterior using a certain prior family should stay in the prior family, while it is reasonably easy to identify the posterior from a sample and a prior; (ii) richness—there should be a member in the prior family that is able to express the modeler's beliefs and

prior information; (iii) interpretability—the prior should be easily interpreted so the modeler can verify that the choice of prior matches prior judgments.

Much of the Bayesian literature for probabilistic grammars and even in general has diverged considerably from these desiderata, and focused only on the first requirement of analytical tractability, which usually yields closed-form solutions for identifying the posterior. There are probably two reasons for this: (i) unlike richness and interpretability, analytical tractability is rigorously defined; (ii) the fact that probabilistic grammars define distributions over combinatorial structures, such as trees, requires fast learning algorithms, which is the result of having an analytically tractable prior. As a result, most of the Bayesian language learning literature has focused on Bayesian models with a Dirichlet prior (Johnson et al., 2007; Goldwater and Griffiths, 2007; Toutanova and Johnson, 2007; Kurihara and Sato, 2006, *inter alia*), which is *conjugate* to the multinomial family (satisfying analytical tractability), and therefore – the posterior after observing sentences and their derivations, which is a distribution over the probabilistic grammar parameters, is a Dirichlet as well. The family of Dirichlet priors is the only family of priors which is conjugate to the multinomial distributions. As a result, a set of independent Dirichlet distributions, defining a distribution over collections of multinomials, is the only possible prior which is conjugate to the parameters of a probabilistic grammar.

We argue that the last two requirements in the desiderata are actually more important than the first one, which is motivated by mere mathematical and computational convenience (and leads to the definition of conjugate priors). We suggest replacing the first requirement with "computational tractability", which requires the following:

- It should be easy to represent the posterior (or an approximation of it) *computationally* (as opposed to having an analytic "closed-form" solution).

- There should be an efficient procedure for identifying the posterior (or its approximation) computationally.

When we replace analytical tractability with computational tractability, the modeler can focus on choosing *rich* priors that can more properly model different structural elements of a grammar. To solve the problem of inference, we can now use approximate inference algorithms such as the one we give in Section 5.4. Indeed, approximations are sometimes required even for the conjugate case, and are always required when the data are incomplete.

We have already defined richness, in a sense, as the notion of softly tying between grammar parameters by using a covariance structure. In the next section, we describe the Dirichlet prior, and explain why it does not satisfy this definition of richness.

5.1.2 Dirichlet Distributions

From the computational perspective, the Dirichlet distribution is indeed a natural choice for a prior over the parameters of the grammar because of its analytical tractability, which makes inference more elegant and less computationally intensive in Bayesian (Equation 5.1) settings. In addition, a Dirichlet prior can encourage sparse solutions (i.e., many $\theta_{k,i} = 0$), a property which is desirable in natural language learning (Johnson et al., 2007), as it corresponds to eliminating unnecessary grammar rules. (Indeed, learning to exclude rules by setting their probabilities to zero might be one way of going about symbolic grammar induction.)

If we use a Dirichlet distribution with a probabilistic grammar, then the hyperparameters for the grammar consist of K vectors with positive elements, the kth of which has length N_k. We denote these hyperparameters by $\boldsymbol{\alpha}$, in which case the prior over the grammar parameters $\boldsymbol{\theta}$ has the form:

$$p(\boldsymbol{\theta} \mid \boldsymbol{\alpha}) = \prod_{k=1}^{K} \left(\frac{\prod_{i=1}^{N_k} \Gamma(\alpha_{k,i})}{\Gamma\left(\sum_{i=1}^{N_k} \alpha_{k,i}\right)} \prod_{i=1}^{N_k} \theta_{k,i}^{\alpha_{k,i}-1} \right) = B(\boldsymbol{\alpha}) \times \prod_{k=1}^{K} \prod_{i=1}^{N_k} \theta_{k,i}^{\alpha_{k,i}-1},$$

where $\Gamma(\cdot)$ is the Gamma function and the factor $B(\boldsymbol{\alpha})$ is constant with respect to $\boldsymbol{\theta}$.

Consider again the simple model of Example 2.1. If we embed it inside model I (Figure 5.1) we arrive at the latent Dirichlet allocation model of Blei et al. (2003), where each example is a document (not a sentence).

The Dirichlet distribution can also be derived as a normalized set of variables of exponentiated *independent* Gamma-distributed variables. More precisely, for each multinomial $\boldsymbol{\theta}_k$ ($k \in \{1, \ldots, K\}$), we can draw N_k independent random samples $v_{k,1}, \ldots, v_{k,N_k}$ from Gamma distributions with shapes $\alpha_{k,1}, \ldots, \alpha_{k,N_k}$, respectively, and scale 1 and then let:

$$\theta_{k,i} = \frac{v_{k,i}}{\sum_{i'=1}^{N_k} v_{k,i'}}.$$

This alternative representation of the Dirichlet distribution points to a weakness: there is no explicit covariance structure present when $\boldsymbol{\theta}$ are drawn from a Dirichlet. The only way $\boldsymbol{\theta}_k$ covary is through the normalization that maps $v_{k,i}$ to the probability simplex. In fact, the correlation between $\theta_{k,i}$ and $\theta_{k,i'}$ is always negative and equals $-\frac{(\alpha_{k,i}\alpha_{k,i'})^{1/2}}{((\alpha_{k,0}-\alpha_{k,i})(\alpha_{k,0}-\alpha_{k,i'}))^{1/2}}$ where $\alpha_{k,0} = \sum_{i=1}^{N_k} \alpha_{k,i}$. This relates back to the desiderata of Raiffa and Schaifer: the covariance (and in fact, variance) structure that the Dirichlet distribution offers is not rich. This is especially true when modeling language, as we explain in the section below.

5.2 Modeling Covariance with Logistic Normal Distributions

A natural candidate for a distribution that models covariance is the multivariate normal distribution. However, values drawn from the multivariate normal distribution can be both positive and negative, and they also do not necessarily normalize to 1, both are requirements from $\boldsymbol{\theta}$ (see Equations 2.2–2.3). Aitchison (1986) suggested a logistic transformation on a multivariate normal variable to get values which correspond to points on the probability simplex. He called it the "logistic normal" distribution.

The logistic normal (LN) distribution maps a $(d-1)$-dimensional multivariate Gaussian to a distribution on the d-dimensional probability simplex, $\{\langle z_1, \ldots, z_d\rangle \in \mathbb{R}^d : z_i \geq 0, \sum_{i=1}^d z_i = 1\}$, as follows:

1. Draw $\boldsymbol{\eta} = \langle \eta_1, \ldots, \eta_{d-1}\rangle$ from a multivariate Gaussian with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$.
2. Let $\eta_d = 0$.
3. For $i \in \{1, \ldots, d\}$, let:
$$z_i = \frac{\exp \eta_i}{\sum_{j=1}^d \exp \eta_j}.$$

Drawing from a $(d-1)$-dimensional Gaussian preserves identifiability; a d-dimensional Gaussian would have an extra degree of freedom, allowing more than one outcome of $\boldsymbol{\eta}$ to lead to the same $\mathbf{z}$.

For probabilistic grammars, we define one LN distribution per multinomial. This gives a prior over each $\boldsymbol{\theta}_k$ that permits covariance among $\langle \theta_{k,1}, \ldots, \theta_{k,N_k}\rangle$.

Blei and Lafferty (2006) and Ahmed and Xing (2007) successfully used the LN distribution for topic models, extending the latent Dirichlet allocation model (Blei et al., 2003). This permits one to have correlation between the various topics for a given document.

We note that the family of logistic normal distributions and the family of Dirichlet distributions are very different from each other. One cannot find two distributions from each class which are arbitrary close to each other in any meaningful sense. However, it can be shown (Aitchison, 1986) that given a Dirichlet distribution with very large α, we can find a logistic normal distribution such that the KL-divergence between the Dirichlet distribution and logistic normal distribution is small.

5.2.1 Sharing Across Multinomials

$$
\left.\begin{array}{lcllcllll}
I_1 & = & \{1{:}2, 3{:}6, 7{:}9\} & = & \{ & I_{1,1}, & I_{1,2}, & I_{1,L_1} & \} \\
I_2 & = & \{1{:}2, 3{:}6\} & = & \{ & I_{2,1}, & I_{2,L_2} & & \} \\
I_3 & = & \{1{:}4, 5{:}7\} & = & \{ & & I_{3,1}, & I_{3,L_3} & \} \\
I_N & = & \{1{:}2\} & = & \{ & I_{4,L_4} & & & \} \\
 & & & & & J_1 & J_2 & J_K &
\end{array}\right\} \text{partition structure } \mathcal{S}
$$

$$
\left.\begin{array}{lcll}
\boldsymbol{\eta}_1 & = & \langle \eta_{1,1}, \eta_{1,2}, \ \eta_{1,3}, \eta_{1,4}, \eta_{1,5}, \eta_{1,6}, \ \eta_{1,7}, \eta_{1,8}, \eta_{1,\ell_1} \rangle & \sim \text{Normal}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1) \\
\boldsymbol{\eta}_2 & = & \langle \eta_{2,1}, \eta_{2,2}, \ \eta_{2,3}, \eta_{2,4}, \eta_{2,5}, \eta_{2,\ell_2} \rangle & \sim \text{Normal}(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2) \\
\boldsymbol{\eta}_3 & = & \langle \eta_{3,1}, \eta_{3,2}, \eta_{3,3}, \eta_{3,4}, \ \eta_{3,5}, \eta_{3,6}, \eta_{3,\ell_3} \rangle & \sim \text{Normal}(\boldsymbol{\mu}_3, \boldsymbol{\Sigma}_3) \\
\boldsymbol{\eta}_4 & = & \langle \eta_{4,1}, \eta_{4,\ell_4} \rangle & \sim \text{Normal}(\boldsymbol{\mu}_4, \boldsymbol{\Sigma}_4)
\end{array}\right\} \text{sample } \boldsymbol{\eta}
$$

$$
\left.\begin{array}{lcl}
\tilde{\boldsymbol{\eta}}_1 & = & \frac{1}{3}\langle \eta_{1,1} + \eta_{2,1} + \eta_{4,1}, \ \eta_{1,2} + \eta_{2,2} + \eta_{4,2} \rangle \\
\tilde{\boldsymbol{\eta}}_2 & = & \frac{1}{3}\langle \eta_{1,3} + \eta_{2,3} + \eta_{3,1}, \ \eta_{1,4} + \eta_{2,4} + \eta_{3,2}, \ \eta_{1,5} + \eta_{2,5} + \eta_{3,3}, \\
 & & \eta_{1,6} + \eta_{2,6} + \eta_{3,4} \rangle \\
\tilde{\boldsymbol{\eta}}_3 & = & \frac{1}{2}\langle \eta_{1,7} + \eta_{3,5}, \ \eta_{1,8} + \eta_{3,6}, \ \eta_{1,9} + \eta_{3,7} \rangle
\end{array}\right\} \text{combine } \boldsymbol{\eta}
$$

$$
\left.\begin{array}{lcl}
\boldsymbol{\theta}_1 & = & (\exp \tilde{\boldsymbol{\eta}}_1) \Big/ \sum_{i'=1}^{N_1} \exp \tilde{\eta}_{1,i'} \\
\boldsymbol{\theta}_2 & = & (\exp \tilde{\boldsymbol{\eta}}_2) \Big/ \sum_{i'=1}^{N_2} \exp \tilde{\eta}_{2,i'} \\
\boldsymbol{\theta}_3 & = & (\exp \tilde{\boldsymbol{\eta}}_3) \Big/ \sum_{i'=1}^{N_3} \exp \tilde{\eta}_{3,i'}
\end{array}\right\} \text{softmax}
$$

Figure 5.2: An example of a shared logistic normal distribution, illustrating Def. 5.1. $N = 4$ experts are used to sample $K = 3$ multinomials; $L_1 = 3$, $L_2 = 2$, $L_3 = 2$, $L_4 = 1$, $\ell_1 = 9$, $\ell_2 = 6$, $\ell_3 = 7$, $\ell_4 = 2$, $N_1 = 2$, $N_2 = 4$, and $N_3 = 3$. From top to bottom: the partition structure $\mathcal{S}$ describes I_j which tell how segment a normal expert into parts which are matched to multinomials ("partition structure $\mathcal{S}$"). Each normal expert is sampled from a multivariate normal ("sample $\boldsymbol{\eta}$"), and then matched and averaged according to the partition strcture ("combine $\boldsymbol{\eta}$"). The final step is exponentiating and normalizing $\boldsymbol{\eta}$ to get $\boldsymbol{\theta}$ ("softmax"). This figure is best viewed in color.

The LN distribution has an inherent limitation when we consider probabilistic models made up of more than one multinomial distribution, such as probabilistic grammars. Each multinomial is drawn separately from an independent Gaussian, so that covariance can only be imposed among events competing within one multinomial, not across multinomials.

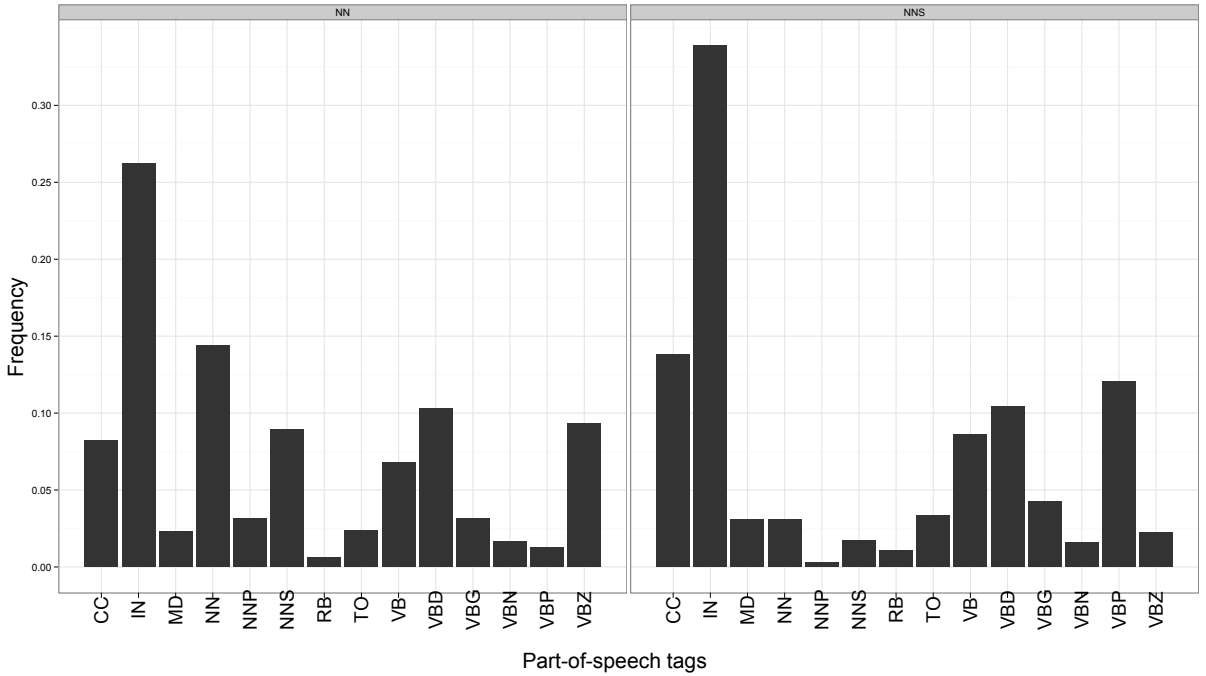


Figure 5.3: Distributions over dependents for plural noun and singular noun for dependency trees (over part-of-speech tags) extracted from the Penn Treebank. NN corresponds to a singular noun and NNS corresponds to a plural noun. The Penn Treebank's phrase-structure annotations were converted to dependencies using the head rules of Yamada and Matsumoto, which are very similar to the ones by Collins (2003). See `http://www.jaist.ac.jp/~h-yamada`.

The question that should be brought up at this point is whether we actually expect this kind of correlation with language data, i.e., correlation across multinomials. Figure 5.3 gives an affirmative answer. The plot in the figure describes two distributions over dependents of singular noun and dependents of plural nouns for dependency trees extracted from the Penn Treebank in English (Marcus et al., 1993). One can imagine these two distributions functioning as two multinomials for generating the children of singular nouns and plural nouns in a probabilistic grammar defined for dependency trees. The key observation here is that clearly see striking similarities between these two distributions. Not only are the distributions over parameters correlated, they actually obtain similar probabilities for various part-of-speech tags being dependents. Therefore, there is real limitation in the inability to model correlation across multinomials.

One way to mend this limitation is to define a single Gaussian over $N \triangleq \sum_{k=1}^{K} N_k$ variables with one $N \times N$ covariance matrix. Then, instead of applying the logistic transformation to the whole vector as a single multinomial, we can apply it to subvectors to get disjoint multinomials. When learning, the large covariance matrix captures correlations between all pairs of events in all multinomials. The induced distribution is called the *partitioned* logistic normal (PLN) distribution. It is a generalization of the LN distribution (see Aitchison, 1986).

In practice, creating a covariance matrix of size $N \times N$ is likely to be too expensive. The dependency model with valence (Klein and Manning, 2004), a grammar that we use for our experiments in Chapter 8, for example, has $O(t^2)$ weights for a part-of-speech vocabulary of size t, requiring a very large multivariate normal distribution with $O(t^4)$ covariance parameters.

To solve this problem, we suggest a refinement of the class of PLN distributions. Instead of using a single normal vector for all of the multinomials, we use several normal vectors, partition each one and then *recombine* parts which correspond to the same multinomial, as an average. Next, we apply the logistic transformation on the mixed vectors (each of which is normally distributed as well). Figure 5.2 gives an example of a non-trivial case of using a SLN distribution, where three multinomials are generated from four normal experts.

We now formalize this notion. For a natural number N, we denote by $1{:}N$ the set $\{1, \ldots, N\}$. For a vector in $v \in \mathbb{R}^N$ and a set $I \subseteq 1{:}N$, we denote by v_I the vector created from v by using the coordinates in I. Recall that K is the number of multinomials in the probabilistic grammar, and N_k is the number of events in the kth multinomial. We define a shared logistic normal distribution with N "experts" over a collection of K multinomial distributions:

Definition 5.1 *Let $\boldsymbol{\eta}_n \sim \mathrm{Normal}(\boldsymbol{\mu}_n, \Sigma_n)$ be a set of multivariate normal variables for $n \in 1{:}N$, where the length of $\boldsymbol{\eta}_n$ is denoted ℓ_n. Let $I_n = \{I_{n,j}\}_{j=1}^{L_n}$ be a partition of $1{:}\ell_n$ into L_n sets, such that $\cup_{j=1}^{L_n} I_{n,j} = 1{:}\ell_n$ and $I_{n,j} \cap I_{n,j'} = \emptyset$ for $j \neq j'$. Let J_k for $k \in 1{:}K$ be a collection of (disjoint) subsets of $\{I_{n,j} \mid n \in 1{:}N, j \in 1{:}\ell_n, |I_{n,j}| = N_k\}$, such that all sets in J_k are of the same size, N_k. Let $\tilde{\boldsymbol{\eta}}_k = \frac{1}{|J_k|} \sum_{I_{n,j} \in J_k} \boldsymbol{\eta}_{n,I_{n,j}}$, and $\theta_{k,i} = \exp(\tilde{\eta}_{k,i}) \,/ \sum_{i'} \exp(\tilde{\eta}_{k,i'})$. We then say $\boldsymbol{\theta}$ distributes according to the* shared logistic normal distribution *with* partition structure $\mathcal{S} = (\{I_n\}_{n=1}^N, \{J_k\}_{k=1}^K)$ *and* normal experts $\{(\boldsymbol{\mu}_n, \Sigma_n)\}_{n=1}^N$ *and denote it by $\boldsymbol{\theta} \sim \mathrm{SLN}(\boldsymbol{\mu}, \boldsymbol{\Sigma}, \mathcal{S})$.*

The partitioned LN distribution in Aitchison (1986) can be formulated as a shared LN distribution where $N = 1$. The LN collection presented in Section 5.2 is the special case where $N = K$, each $L_n = 1$, each $\ell_k = N_k$, and each $J_k = \{I_{k,1}\}$.

We note that there is an issue with identifiability that we need to resolve with SLN distributions, as with the LN distribution. It is required that for all multinomials, we set the first value of the samples from the normal expert to 0. For simplicity, we did not include it explicitly in Definition 5.1, because this can be achieved by setting the normal expert's mean and variance values to 0 in the first index of each normal expert ($\eta_{n,1} = 0$ for all n).

The covariance among arbitrary $\theta_{k,i}$ is not defined directly; it is implied by the definition of the normal experts $\boldsymbol{\eta}_{n,I_{n,j}}$, for each $I_{n,j} \in J_k$. We note that a SLN can be represented as a PLN by relying on the distributivity of the covariance operator, and merging all the partition structure into one (perhaps sparse) covariance matrix. SLNs, in that case, represent a subset of PLNs with a factored structure on the covariance matrices.

It is convenient to think of each $\eta_{i,j}$ as a weight associated with a unique event's probability, a certain outcome of a certain multinomial in the probabilistic grammar. By letting different $\eta_{i,j}$ covary with each other, we strengthen the relationships among $\theta_{k,j}$ and permit learning of the one to affect the learning of the other. Definition 5.1 also suggests that in order to get the multinomial family, we define local log-linear models in the form of a product-of-experts (Hinton, 1999), because the exponential of an average of normals becomes a product of (unnormalized) probabilities. We discuss this more in Section 5.2.2.

We note that the partition structure is a predetermined hyperparameter, which remains fixed during learning. In our experiments, it encodes domain knowledge about the languages we experiment with (Section 8.5). We believe this is a key advantage of SLN in this setting: marrying the notions of prior knowledge and a

Bayesian prior. The beliefs of the model about a language can be encoded as a distribution over the parameters.

5.2.2 Local Log-Linear Models over Parameters

We give now another interpretation of the shared logistic normal prior using a feature representation. A probabilistic grammar with a shared logistic normal prior can be thought of as a probabilistic grammar where the grammar's parameters are themselves modeled using a local log-linear model with a Gaussian prior over the weights of this log-linear model. Let $\boldsymbol{\theta}_k$ be a multinomial in the collection of multinomials for a probabilistic grammar. Then, according to Definition 5.1 we have:

$$\theta_{k,i} = \frac{\exp\left(\boldsymbol{g}_k(i) \cdot \boldsymbol{\eta}\right)}{Z_k(\boldsymbol{\eta})},$$

where $\boldsymbol{\eta}$ is a vector of length $\sum_{n=1}^{N} \ell_n$, a concatenation of all normal experts, and $\boldsymbol{g}_k(i)$ is a feature vector, again of length $\sum_{n=1}^{N} \ell_n$, which is divided into subvectors $\boldsymbol{g}_{k,n}(i)$ each of length ℓ_n. $g_{k,n,j}(i) = 1/|J_k|$ if the ith event in the kth multinomial uses the jth coordinate of the nth normal expert—that is, there exists an $I_{n,r} \in I_n \cap J_k$ such that $j \in I_{n,r}$ (according to Definition 5.1)—and 0 otherwise. The term $Z_k(\boldsymbol{\eta})$ is a normalization constant of the form:

$$Z_k(\boldsymbol{\eta}) = \sum_{i'} \exp\left(\boldsymbol{g}_k(i') \cdot \boldsymbol{\eta}\right).$$

Note that the features in the local log-linear model refer to the *hyperparameters* of the SLN, more specifically, the partition structure. They do not refer to the observed data or the latent structural elements in the probabilistic grammar. These features have a Gaussian prior over them, represented by the normal experts' mean values and covariance matrices ($\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$). In that case, the Gaussian prior which we optimize during inference using empirical Bayes (Section 5.4) can be thought of as a quadratic penalty on the local log-linear weights. We note that in most cases in the literature, Gaussian priors (or L_2 regularizers) are used with mean value $\mathbf{0}$ and a uniform diagonal covariance matrix, in order to push feature weights to values close to 0. This is not the case with our model.

Berg-Kirkpatrick et al. (2010) used the idea of local log-linear models for several unsupervised natural language processing tasks, including dependency grammar induction and part-of-speech tagging. Instead of using features that are based

on a Gaussian prior, they used a set of ordinary binary features, which describe relationships between different parameters in a similar way to the ones presented in Section 8.5. In Berg-Kirkpatrick et al. (2010), one can use essentially any binary feature on the parameter space, which would require estimating an additional parameter for the model (the weight of the feature). With the logistic normal, the number of parameters we require to estimate grows with the number of normal experts that we use for estimation. Each such normal expert introduces a new set of features according to the formulation above.

5.3 Variational Inference with Logistic Normal Distributions

The lack of conjugacy of the logistic normal distribution to the multinomial family complicates the inference of distributions over $\boldsymbol{\theta}$ and distributions over the hidden derivations $\mathbf{y}$ from the probabilistic grammar, given a sequence of observed sentences $\boldsymbol{x}_1, ..., \boldsymbol{x}_M$.

Mimno et al. (2008) explored inference with the logistic normal distribution using sampling with an auxiliary variable method. However, sampling is notoriously slow to converge, especially with complicated structures such as grammatical derivations. The algorithm Mimno et al. suggest is also rather complicated, while alternatives, such as mean-field variational inference (Wainwright and Jordan, 2008), offer faster convergence and a more intuitive solution to the problem of non-conjugacy of the logistic normal distribution.

Variational inference is a deterministic alternative to MCMC, which casts posterior inference as an optimization problem (Jordan et al., 1999; Wainwright and Jordan, 2008). The optimized function is a bound on the marginal likelihood of the observations, which is expressed in terms of a so-called "variational distribution" over the hidden variables. When the bound is tightened, that distribution is close to the posterior of interest. Not only do variational methods tend to converge faster than MCMC, they can be more easily parallelized over multiple processors in a framework such as MapReduce (Dean and Ghemawat, 2004).

Variational inference algorithms have been successfully applied to various grammar and syntax learning tasks (Kurihara and Sato, 2006; Liang et al., 2007; Cohen et al., 2008; Headden et al., 2009; Boyd-Graber and Blei, 2010, *inter alia*). We give the full technical details of mean-field variational inference for probabilistic grammars with logistic normal priors in Section 5.3.2, and turn to give a

brief overview of the main technical details next, under the simplifying assumption that we have a single observation $\boldsymbol{x}$.

Mean-field variational inference in the Bayesian setting relies on two principal approximations: the first approximation is done to the marginalized log-likelihood. Using Jensen's inequality and an auxiliary distribution $q(\boldsymbol{\theta}, \mathbf{y})$, later to be used as our approximate posterior, we bound the log-likelihood, marginalizing out the parameters and the hidden derivations in the grammar:

$$\log \int \sum_{\mathbf{y}} p(\boldsymbol{x}, \mathbf{y}, \boldsymbol{\theta} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}, \mathcal{S}, \mathbf{G}) \, d\boldsymbol{\theta} \geq \mathbb{E}_q[\log p(\boldsymbol{x}, \mathbf{y}, \boldsymbol{\theta} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}, \mathcal{S}, \mathbf{G})] + H(q), \tag{5.2}$$

where $H(q)$ denotes the Shannon entropy of q. The goal of the approximation in Equation 5.2 is to derive a bound which is optimized with respect to q, instead of optimizing the marginalized log-likelihood, which is intractable. The distribution q serves as our approximate posterior.

The bound in Equation 5.2 requires further approximation, the mean-field approximation, to be tractable. This mean-field approximation states that $q(\boldsymbol{\theta}, \mathbf{y})$ is factorized and has the following form:

$$q(\boldsymbol{\theta}, \mathbf{y}) = q(\boldsymbol{\theta}) q(\mathbf{y}).$$

The variational distributions, $q(\boldsymbol{\theta})$ and $q(\mathbf{y})$ can take an arbitrary form, as long as the bound in Equation 5.2 can be efficiently maximized with respect to these variational distributions. For the case of logistic normal priors, an additional approximation will be necessary (a first-order Taylor approximation to the log of the normalization of the logistic normal distribution), because of the lack of conjugacy of the logistic normal priors to the multinomial family (see Section 5.3.2). We show in Section 5.3.2 that even though $q(\mathbf{y})$ can have an arbitrary form, in order to maximize the variational bound it needs to have the form of a probabilistic grammar, dominated by the grammar's variational parameters. This makes inference tractable through the use of an inside-outside algorithm with a weighted grammar of the same form as the original model. The mean-field approximation yields an elegant algorithm, which looks similar to the Expectation-Maximization algorithm (Section 2.2), alternating between optimizing the bound in Equation 5.2 with respect to $q(\boldsymbol{\theta})$ and with respect to $q(\mathbf{y})$.

5.3.1 Variational EM

The variational inference algorithm in Section 5.3 assumes that the $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are fixed. We are interested in obtaining an *estimate* for $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$, so that we can fit the data and then use the learned model as described in Section 5.4.1 to decode new data (e.g., the test set in our experiments). To achieve this, we will use the above variational method within an EM algorithm that estimates $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ in empirical Bayes fashion. In the E-step, we maximize the bound with respect to the variational parameters using coordinate ascent as in Section 5.3. We optimize each of these separately in turn, cycling through them, using appropriate optimization algorithms for each. In the M-step, we apply maximum likelihood estimation with respect to $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ given sufficient statistics gathered from the variational parameters in the E-step.

The algorithm for variational inference with probabilistic grammars using a logistic normal prior is defined in Algorithms 2–4.[1] Since the updates for $\tilde{\zeta}_k^{(t)}$ are fast, we perform them after each optimization routine in the E-step (suppressed for clarity). There are variational parameters for each training example, indexed by m. We denote by B the variational bound in Equation 5.6. Our stopping criterion relies on the likelihood of a held-out set using a point estimate of the model.

5.3.2 Derivation of the Variational Inference Bound

This section can be skimmed over in first reading. Its goal is to give the details of the derivation of the variational inference algorithm. We give a derivation of a variational inference algorithm for model I, with the shared logistic normal distribution as a prior. The derivation is based on the one given in Blei and Lafferty (2006). The derivation for model II can be followed similarly, as explained below. For model I, we seek to find an approximation posterior function $q(\boldsymbol{\eta}_1, ..., \boldsymbol{\eta}_M, \mathbf{y}_1, ..., \mathbf{y}_M)$ that maximizes a lower bound (the negated variational free energy) on the log-likelihood, a bound which is achieved using Jensen's inequality (the following probability quantities should be understood as if we always

[1] An implementation of the algorithm is available at `http://www.ark.cs.cmu.edu/DAGEEM`.

condition on the grammar $\mathbf{G}$):

$$\sum_{m=1}^{M} \log \sum_{\mathbf{y}} p(\boldsymbol{x}_m, \mathbf{y} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}, \mathcal{S})$$
$$\geq \sum_{m=1}^{M} \left(\sum_{i=1}^{N} \mathbb{E}_q \left[\log p(\boldsymbol{\eta}_{m,i} \mid \boldsymbol{\mu}_i, \Sigma_i) \right] + \mathbb{E}_q \left[\log p(\boldsymbol{x}_m, \mathbf{y}_m \mid \boldsymbol{\eta}_m, \mathcal{S}) \right] \right) + H(q) \quad (5.3)$$

$H(\cdot)$ denotes the Shannon entropy.

We make a mean-field assumption, and assume that the posterior has the following form:

$$q(\boldsymbol{\eta}_1, ..., \boldsymbol{\eta}_M, \mathbf{y}_1, ..., \mathbf{y}_M) = \prod_{m=1}^{M} q_m(\boldsymbol{\eta}_m, \mathbf{y}_m), \quad (5.4)$$

where

$$q_m(\boldsymbol{\eta}_m, \mathbf{y}_m) = \left(\prod_{k=1}^{N} \prod_{i=1}^{L_k} q_m(\eta_{m,k,i} \mid \tilde{\mu}_{m,k,i}, \tilde{\sigma}^2_{m,k,i}) \right) \times q_m(\mathbf{y}_m),$$

and $q_m(\eta_{m,k,i} \mid \tilde{\mu}_{m,k,i}, \tilde{\sigma}^2_{m,k,i})$ is a Gaussian with mean $\tilde{\mu}_{m,k,i}$ and variance $\tilde{\sigma}^2_{m,k,i}$. Note that this means that the *variational* distributions have a diagonal matrix for their covariance structure. The model covariance matrices (the hyperparameters $\boldsymbol{\Sigma}$) can still have covariance structure. This selection of variational distributions makes inference much easier. The factorized form of Equation 5.4 implies the following identities:

$$\mathbb{E}_q \left[\log p(\boldsymbol{\eta}_{m,i} \mid \boldsymbol{\mu}_i, \Sigma_i) \right] = \mathbb{E}_{q_m} \left[\log p(\boldsymbol{\eta}_{m,i} \mid \boldsymbol{\mu}_i, \Sigma_i) \right],$$
$$\mathbb{E}_q \left[\log p(\boldsymbol{x}_m, \mathbf{y}_m \mid \boldsymbol{\eta}_m, \mathcal{S}) \right] = \mathbb{E}_{q_m} \left[\log p(\boldsymbol{x}_m, \mathbf{y}_m \mid \boldsymbol{\eta}_m, \mathcal{S}) \right],$$
$$H(q) = \sum_{m=1}^{N} H(q_m).$$

Let $\eta^C_{k,i}$ be an intermediate variable, denoting the average of the normal experts which appear in the partition structure and determine the value of the ith event in the kth multinomial of the grammar. More formally, we define the vector η^C_k of length N_k to be:

$$\eta^C_k \triangleq \frac{1}{|J_k|} \sum_{I_{r,j} \in J_k} \eta_{r,I_{r,j}}.$$

Unfolding the expectation with respect to $q_m(\mathbf{y}_m)$ in the second term in Equation 5.3, while recalling that $\boldsymbol{\theta}_m$ is a deterministic function of $\boldsymbol{\eta}_m$ that averages different subvectors from the collection of multinomials $\boldsymbol{\eta}_m$ according to the partition structure $\mathcal{S}$, we have that:

$$\begin{aligned}
&\mathbb{E}_{q_m}\left[\log p(\boldsymbol{x}_m, \mathbf{y}_m \mid \boldsymbol{\eta}_m, \mathcal{S})\right] \\
&= \mathbb{E}_{q_m(\boldsymbol{\eta}_m)}\left[\sum_{k=1}^{K}\sum_{i=1}^{N_k} \underbrace{\sum_{\mathbf{y}} q_m(\mathbf{y}_m) f_{k,i}(\boldsymbol{x}_m, \mathbf{y}_m)}_{\tilde{f}_{m,k,i}} \log \theta_{m,k,i}\right] \\
&= \mathbb{E}_{q_m(\boldsymbol{\eta}_m)}\left[\sum_{k=1}^{K}\sum_{i=1}^{N_k} \tilde{f}_{m,k,i}\left(\eta^C_{m,k,i} - \log \sum_{i'=1}^{N_k} \exp \eta^C_{m,k,i'}\right)\right], \qquad (5.5)
\end{aligned}$$

where $\tilde{f}_{m,k,i}$ is the expected number of occurrences of the ith event in distribution k, under $q_m(\mathbf{y}_m)$. With many kinds of probabilistic grammars, this quantity can be computed using a dynamic programming algorithm like the forward-backward or inside-outside algorithm.

The logarithm term in Equation 5.5 is problematic because of the expectation with respect to $q_m(\boldsymbol{\eta}_m)$. We approximate it with a first-order Taylor expansion, introducing $M \times K$ more variational parameters $\tilde{\zeta}_{m,k}$ for $m \in \{1, ..., M\}$ and $K \in \{1, ..., K\}$:

$$\log\left(\sum_{i'=1}^{N_k} \exp \eta^C_{m,k,i'}\right) \leq \log \tilde{\zeta}_{m,k} - 1 + \frac{1}{\tilde{\zeta}_{m,k}} \sum_{i'=1}^{N_k} \exp \eta^C_{m,k,i'}.$$

We note that the value $\mathbb{E}_{q_m(\boldsymbol{\eta}_m)}\left[\exp(\eta^C_{m,k,i'})\right]$ can be calculated by evaluating the moment-generating function of the normal distribution $g(t) = \mathbb{E}_{q_m(\boldsymbol{\eta}_m)}\left[\exp(t\eta^C_{m,k,i'})\right]$ at $t = 1$. We now have:

Algorithm 2: Variational EM for probabilistic grammars with LN prior

Input: initial parameters $\boldsymbol{\mu}^{(0)}$, $\boldsymbol{\Sigma}^{(0)}$, training data $\boldsymbol{x}$, and development data $\boldsymbol{x}'$

Output: learned parameters $\boldsymbol{\mu}$, $\boldsymbol{\Sigma}$

$t \leftarrow 1$;

repeat

Call E-Step for each training example $m = 1, ..., M$ *(Algorithm 3)*

Call M-Step (Algorithm 4)

$t \leftarrow t + 1$;

until *likelihood of held-out data,* $p(\boldsymbol{x}' \mid E[\boldsymbol{\mu}^{(t)}])$*, decreases*;

return $\boldsymbol{\mu}^{(t)}$, $\boldsymbol{\Sigma}^{(t)}$

$$
\begin{aligned}
&\mathbb{E}_{q_m}[\log p(\boldsymbol{x}_m, \mathbf{y}_m \mid \boldsymbol{\eta}_m, \mathcal{S})] \\
&\geq \mathbb{E}_{q_m(\boldsymbol{\eta}_m)}\left[\sum_{k=1}^{K}\sum_{i=1}^{N_k} \tilde{f}_{m,k,i}\left(\eta_{m,k,i} - \log\tilde{\zeta}_{m,k} + 1 - \frac{1}{\tilde{\zeta}_{m,k}}\sum_{i'=1}^{N_k}\exp\eta_{m,k,i'}\right)\right] \\
&= \sum_{k=1}^{K}\sum_{i=1}^{N_k} \tilde{f}_{m,k,i} \underbrace{\left(\tilde{\mu}_{m,k,i} - \log\tilde{\zeta}_{m,k} + 1 - \frac{1}{\tilde{\zeta}_{m,k}}\sum_{i'=1}^{N_k}\exp\left(\tilde{\mu}^C_{m,k,i} + \frac{(\tilde{\sigma}^C_{m,k,i})^2}{2}\right)\right)}_{\tilde{\psi}_{m,k,i}} \\
&= \sum_{k=1}^{K}\sum_{i=1}^{N_k} \tilde{f}_{m,k,i}\tilde{\psi}_{m,k,i}
\end{aligned}
$$

where we use again the properties of the shared logistic normal distribution and rely on the partition structure $\mathcal{S}$ to define:

$$
\begin{aligned}
\tilde{\mu}^C_{m,k} &\triangleq \frac{1}{|J_k|}\sum_{I_{r,j}\in J_k}\tilde{\mu}_{m,r,I_{r,j}}, \\
(\tilde{\sigma}^C_{m,k})^2 &\triangleq \frac{1}{|J_k|^2}\sum_{I_{r,j}\in J_k}\tilde{\sigma}^2_{m,r,I_{r,j}}.
\end{aligned}
$$

Note the shorthand $\tilde{\psi}_{k,i}$ to denote an expression involving $\tilde{\boldsymbol{\mu}}^C$, $\tilde{\boldsymbol{\sigma}}^C$, and $\tilde{\boldsymbol{\zeta}}$.

The final form of our bound is:[2]

$$\log p(\boldsymbol{x}, \mathbf{y} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) \geq \left(\sum_{k=1}^{K} \mathbb{E}_q \left[\log p(\boldsymbol{\eta}_k \mid \boldsymbol{\mu}_k, \Sigma_k) \right] \right) + \left(\sum_{k=1}^{K} \sum_{i=1}^{N_k} \tilde{f}_{k,i} \tilde{\psi}_{k,i} \right) + H(q). \tag{5.6}$$

Using an EM-style algorithm, we will alternate between finding the maximizing $q(\boldsymbol{\eta})$ and the maximizing $q(\mathbf{y})$. Maximization with respect to $q_m(\boldsymbol{\eta}_m)$ is not hard, because $q(\boldsymbol{\eta})$ is parameterized. The following lemma shows that fortunately, finding the maximizing $q_m(\mathbf{y}_m)$, which we did not parameterize originally, is not hard either:

Lemma 5.2 *Let* $r(\mathbf{y}_m \mid \boldsymbol{x}_m, e^{\tilde{\boldsymbol{\psi}}_m})$ *denote the conditional distribution over* $\mathbf{y}_m$ *given* $\boldsymbol{x}_m$ *defined as:*

$$r_m(\mathbf{y}_m \mid \boldsymbol{x}_m, e^{\tilde{\boldsymbol{\psi}}}) = \frac{1}{Z_m(\tilde{\boldsymbol{\psi}}_m)} \prod_{k=1}^{K} \prod_{i=1}^{N_k} \exp\left(\tilde{\psi}_{m,k,i} f_{m,k,i}(\boldsymbol{x}_m, \mathbf{y}_m) \right)$$

where $Z_m(\tilde{\boldsymbol{\psi}}_m)$ *is a normalization constant. Then* $q_m(\mathbf{y}_m) = r_m(\mathbf{y}_m \mid \boldsymbol{x}_m, e^{\tilde{\boldsymbol{\psi}}_m})$ *maximizes the bound in Equation* 5.6*.*

Proof First note that $H(q_m) = H(q_m(\boldsymbol{\eta}_m \mid \tilde{\boldsymbol{\mu}}_m, \tilde{\boldsymbol{\sigma}}_m)) + H(q_m(\mathbf{y}_m))$. This means that the terms we are interested in maximizing from Equation 5.6 are the following, after plugging in $\tilde{f}_{m,k,i}$ explicitly:

$$L = \operatorname*{argmax}_{q_m(\mathbf{y}_m)} \sum_{\mathbf{y}_m} q_m(\mathbf{y}_m) \left(\sum_{k=1}^{K} \sum_{i=1}^{N_k} f_{m,k,i}(\boldsymbol{x}_m, \mathbf{y}_m) \tilde{\psi}_{m,k,i} \right) + H(q_m(\mathbf{y}_m)).$$

Then, note that:

$$L = \operatorname*{argmin}_{q_m(\mathbf{y}_m)} D_{\mathrm{KL}} \left(q_m(\mathbf{y}_m) \;\middle\|\; r_m(\mathbf{y}_m \mid \boldsymbol{x}_m, e^{\tilde{\boldsymbol{\psi}}_m}) \right), \tag{5.7}$$

where D_{KL} denotes the KL divergence. To see that, combine the definition of KL divergence with the fact that $\sum_{k=1}^{K} \sum_{i=1}^{N_k} f_{m,k,i}(\boldsymbol{x}, \mathbf{y}) \tilde{\psi}_{m,k,i} - \log Z_m(\tilde{\boldsymbol{\psi}}_m) = \log r_m(\mathbf{y}_m \mid \boldsymbol{x}_m, e^{\tilde{\boldsymbol{\psi}}_m})$ where $\log Z_m(\tilde{\boldsymbol{\psi}})$ does not depend on $q_m(\mathbf{y}_m)$. Equation 5.7 is minimized when $q_m = r_m$. □

[2] A tighter bound, based on a second-order approximation, was proposed in Ahmed and Xing (2007). We use a first-order approximation for simplicity, similar to Blei and Lafferty (2006).

The above lemma demonstrates that the minimizing $q_m(\mathbf{y}_m)$ has the same form as the probabilistic grammar $\mathbf{G}$, only without having sum-to-one constraints on the weights (leading to the required normalization constant $Z_m(\tilde{\boldsymbol{\psi}}_m)$). As in classic EM with probabilistic grammars, we never need to represent $q_m(\mathbf{y}_m)$ explicitly; we need only $\tilde{\mathbf{f}}_m$, which can be calculated as expected feature values under $r_m(\mathbf{y}_m \mid \boldsymbol{x}_m, e^{\tilde{\boldsymbol{\psi}}_m})$ using dynamic programming.

Variational inference for model II is done similarly to model I (Figure 5.1). The main difference is that instead of having variational parameters for each $q_m(\boldsymbol{\eta}_m)$, we have a single distribution $q(\boldsymbol{\eta})$, and the sufficient statistics from the inside-outside algorithm are used altogether to update it during variational inference.

5.4 Decoding: Inferring $\mathbf{y}$

The estimation procedure in Section 5.3.1 eventually outputs a set of hyperparameters that parametrize the distribution over the parameters. We consider four approaches to using these estimated hyperparameters to infer $\mathbf{y}$, i.e. to output derivations or structures given unseen input sentences. These four approaches are Viterbi decoding, minimum Bayes risk decoding, posterior decoding, and a new method which is specialized for exploiting the covariance structure present with the shared logistic normal distribution, which we call "committee decoding." We now explain each approach in detail.

5.4.1 Viterbi Decoding

Classical statistical approaches to language processing normally assume that inputs (here, sentences $\boldsymbol{x}$) are independently and identically distributed. Decoding is the problem of choosing an analysis (here, grammatical derivation $\mathbf{y}$) given the input. Most commonly this is accomplished by choosing the most probable analysis:

$$\mathbf{y}^* = \operatorname*{argmin}_{\mathbf{y}} p(\mathbf{y} \mid \boldsymbol{x}, \boldsymbol{\theta}, \mathbf{G}) = \operatorname*{argmin}_{\mathbf{y}} p(\boldsymbol{x}, \mathbf{y} \mid \boldsymbol{\theta}, \mathbf{G}). \tag{5.8}$$

This is commonly called "Viterbi" decoding, referring to the algorithm that accomplishes the maximization for hidden Markov models.

In this case, we use a point estimate of the $\boldsymbol{\theta}$, according to the output of the variational EM algorithm. More specifically, we exponentiate and normalize the

mean values of the estimated logistic normal distribution to obtain $\boldsymbol{\theta}$. Then, we proceed with Viterbi decoding according to Eq. 5.8. When the grammar is a probabilistic context-free grammar, for example, Viterbi decoding would amount to using an algorithm such as the CKY algorithm to find the most probable parse.

5.4.2 Minimum Bayes Risk Decoding

An alternative to Viterbi decoding is to choose the analysis that minimizes *risk*, or the expectation (under the model) of a cost function.[3] Let $\mathrm{cost}(\mathbf{y}, \mathbf{y}^*)$ denote the nonnegative cost of choosing analysis $\mathbf{y}$ when the correct analysis is $\mathbf{y}^*$.

$$\mathbf{y}^* = \operatorname*{argmax}_{\mathbf{y}} \mathbb{E}_{p(\cdot|\boldsymbol{x},\boldsymbol{\theta},\mathbf{G})} \mathrm{cost}(\mathbf{y}, \cdot) = \operatorname*{argmax}_{\mathbf{y}} \sum_{\mathbf{y}'} p(\mathbf{y}' \mid \boldsymbol{x}, \boldsymbol{\theta}, \mathbf{G})\ \mathrm{cost}(\mathbf{y}, \mathbf{y}').$$

This is known as minimum Bayes risk (MBR) decoding (Goodman, 1996).[4] In Chapter 7 we define precisely the cost function in the context of dependency grammar induction.

5.4.3 Posterior Decoding

One additional option of inferring $\mathbf{y}$ for unseen sentences, which makes use of a fully Bayesian setting, is that of "posterior decoding". In the Bayesian setting, decoding might be accomplished using the posterior over derivations, marginalizing out the unknown grammar weights. For model I, this would correspond to:

$$\mathbf{y}^* = \operatorname*{argmax}_{\mathbf{y}} p(\mathbf{y} \mid \boldsymbol{\alpha}, \mathbf{G}) = \operatorname*{argmax}_{\mathbf{y}} \int p(\boldsymbol{\theta} \mid \boldsymbol{\alpha}, \mathbf{G}) p(\boldsymbol{x}, \mathbf{y} \mid \boldsymbol{\theta}, \mathbf{G})\ d\boldsymbol{\theta}. \quad (5.9)$$

Unfortunately, there is no closed-form solution for the integral in Equation 5.9 and finding $\mathbf{y}^*$ is intractable. We therefore have to resort to approximate inference similar to the variational inference algorithm described above. More specifically, we would have to run the variational EM algorithm with the training data, and then take an extra E-step with the unseen sentences, to obtain the posterior distributions

[3] We note that as a matter of fact, minimizing the expectation under the model of a cost function is actually a generalization of Viterbi decoding. We can recover Viterbi decoding by using the 0-1 cost function.

[4] In some cases, decoding selects only certain salient aspects of a derivation, such as the derived tree corresponding to a tree adjoining grammar's *derivation* tree. In such cases, Viterbi and/or MBR decoding may require approximations.

over the structures of the unseen sentences.[5] We performed some preliminary experiments using this method, and concluded that its performance is very close to the performance of Viterbi decoding. Yet, this approach to decoding is much more expensive, because of the additional E-step. For this reason, in Chapter 8 we do not report results for using this approach, but instead focus on the other three methods, which are fast.

5.4.4 Comittee Decoding

Neither Viterbi nor MBR decoding uses the entire distribution over grammar weights. In the LN case, for example, the covariance matrix Σ is ignored. We suggest "committee decoding," in which a set of randomly sampled grammar weights are drawn for each sentence to be parsed. The weights are drawn from the learned distribution over grammar weights, parameterized by $\boldsymbol{\mu}$ and Σ in the LN case. Viterbi or MBR decoding can then be applied. Note that this decoding mechanism is randomized: we sample a grammar per sentence, and use it to decode. In our empirical study (Chapter 8), we apply this decoding mechanism ten times, and average performance. This decoding method is attractive because it has generalization error guarantees: in a PAC-Bayesian framework, it can be shown that the error of committee parsing on the sample given should be close to the expected error (see Seeger, 2002; McAllester, 2003; Banerjee, 2006).

5.5 Summary

In this chapter, we presented an estimation method for probabilistic grammars which is framed in a Bayesian setting. In this setting, the estimation method uses a prior family based on the logistic normal distribution to softly tie between the various parameters of the grammar.

Softly tying between various parameters in the grammar is linguistically motivated. We will consider in Chapter 7 a grammar induction task, where the probabilistic grammar is defined over part of speech tags. In this case, we can consider two versions for choosing the set of part of speech tags: coarse part of speech tags and more fine-grained part of speech tags.

Estimating the grammar directly from the coarse part of speech tags leads to poor results. Information is lost when we do learning this way. When using fine-

[5] We note that model II creates dependence among the derivations of the different sentences in the training set, requiring a different inference procedure.

grained part of speech tags, performance improves, and it improves even more considerably when we softly tie between the various fine-grained part of speech tag categories.

Algorithm 3: E-Step (subroutine for Algorithm 2)

repeat

optimize for $\tilde{\mu}_{m,k}^{(t)}, k = 1, ..., K$: use conjugate gradient descent with

$$\frac{\partial B}{\partial \tilde{\mu}_{m,k,i}} = -\left((\Sigma_k^{(t-1)})^{-1})(\mu_k^{(t-1)} - \tilde{\mu}_{m,k})\right)_i - \tilde{f}_{m,k,i} + \sum_{i'=1}^{N_k} \left(\tilde{f}_{m,k,i'}/\tilde{\zeta}_{m,k}\right) \exp\left(\frac{\tilde{\mu}_{k,i'} + \tilde{\sigma}_{k,i'}^2}{2}\right)$$

optimize $\tilde{\sigma}_{m,k}^{(t)}, k = 1, ..., K$: use Newton's method for each coordinate (with $\tilde{\sigma}_{m,k,i} > 0$) with

$$\frac{\partial B}{\partial \tilde{\sigma}_{m,k,i}^2} = -\frac{\Sigma_{k,ii}^{(t-1)}}{2} - \frac{\left(\sum_{i'=1}^{N_k} \tilde{f}_{m,k,i'}\right) \exp\left(\frac{\tilde{\mu}_{m,k,i} + \tilde{\sigma}_{m,k,i}^2}{2}\right)}{2\tilde{\zeta}_{m,k}} + \frac{1}{2\tilde{\sigma}_{m,k,i}^2}$$

update $\tilde{\zeta}_{m,k}^{(t)}, \forall k$:

$$\tilde{\zeta}_{m,k}^{(t)} \leftarrow \sum_{i=1}^{N_k} \exp\left(\tilde{\mu}_{m,k,i}^{(t)} + \frac{(\tilde{\sigma}_{m,k,i}^{(t)})^2}{2}\right)$$

update $\tilde{\boldsymbol{\psi}}_{m,k}^{(t)}, \forall k$:

$$\tilde{\psi}_{m,k,i}^{(t)} \leftarrow \tilde{\mu}_{m,k,i}^{(t)} - \log \tilde{\zeta}_{m,k}^{(t)} + 1 - \frac{1}{\tilde{\zeta}_{m,k}^{(t)}} \sum_{i'=1}^{N_k} \exp\left(\tilde{\mu}_{m,k,i}^{(t)} + \frac{(\tilde{\sigma}_{m,k,i}^{(t)})^2}{2}\right)$$

compute expected counts $\tilde{\mathbf{f}}_{m,k}^{(t)}, k = 1, ..., K$: use an inside-outside algorithm to re-estimate expected counts $\tilde{f}_{m,k,i}^{(t)}$ in weighted grammar $q(y)$ with weights $e^{\tilde{\boldsymbol{\psi}}_m}$;

until *B does not change*;

Algorithm 4: M-Step (subroutine for Algorithm 2)

Estimate $\boldsymbol{\mu}^{(t)}$ and $\boldsymbol{\Sigma}^{(t)}$ using the following maximum likelihood closed-form solution:

$$\mu_{k,i}^{(t)} \leftarrow \frac{1}{M} \sum_{m=1}^{M} \tilde{\mu}_{m,k,i}^{(t)}$$

$$\left[\Sigma_k^{(t)}\right]_{i,j} \leftarrow \frac{1}{M} \left(\sum_{m=1}^{M} \tilde{\mu}_{m,k,i}^{(t)} \tilde{\mu}_{m,k,j}^{(t)} + (\tilde{\sigma}^{(t)})_{m,k,i}^2 \delta_{i,j} + M \mu_{k,i}^{(t)} \mu_{k,j}^{(t)} - \mu_{k,j}^{(t)} \sum_{m=1}^{M} \tilde{\mu}_{m,k,i}^{(t)} - \mu_{k,i}^{(t)} \sum_{m=1}^{M} \tilde{\mu}_{m,k,j}^{(t)} \right),$$

where $\delta_{i,j} = 1$ if $i = j$ and 0 otherwise.

Chapter 6

Estimation of Probabilistic Grammars in the Nonparametric Setting

Bayesian nonparametric approaches (NP Bayes) to grammar learning have an attractive property, that the size of the model grows as we have more data to explain the patterns in the data. The main application of these nonparametric methods in computational linguistics employ the Dirichlet process (Antoniak, 1974; Pitman, 2002).

The reason for focusing on the Dirichlet process is similar to the reason for the multinomial family being a building block in the distribution catalog of computational linguistics. In the underlying definition of the Dirichlet process (especially when viewed through the stick breaking process) there is a multinomial over a discrete, infinite space of states that can correspond to nonterminals in a grammar (Finkel et al., 2007; Liang et al., 2007) or other elements in a grammar.

Indeed, one example of a successful use of NP Bayes in a grammatical setting is that of **adaptor grammars** (Johnson et al., 2006; Johnson, 2008b,a; Johnson and Goldwater, 2009), which define a distribution over derivations in a PCFG such that the posterior exhibits a rich-get-richer property: whole subtrees that have been sampled frequently will tend to have higher weight in the posterior.

In the unsupervised setting, adaptor grammars require an expensive posterior inference procedure. This inference procedure, which appeared first in Johnson et al. (2006), is based on a Metropolis-Hastings sampler nested in a Gibbs sampler.

In this chapter, we offer an empirical Bayesian framework to do inference and estimation with adaptor grammars. Like in Chapter 5, this framework is based on

variational inference. The key advantage that it has over the sampler suggested in Johnson et al. (2006) is that it is parallelizable. This holds because of the more general reason that the E-step in variational expectation-maximization is parallelizable.

In a sense, our variational inference algorithm reduces the nonparametric form of adaptor grammars to a more managable parametrically represented form. This form is embodied in a variational distribution estimated through a variational EM algorithm.

The rest of this chapter is organized as follows. In Section 6.1 we describe a stick-breaking representation of adaptor grammars, which enables variational inference (Section 6.2) and a well-defined incorporation of recursion into adaptor grammars. In Section 6.3 we use our variational inference algorithm to show that it indeed achieves similar results to the sampler of Johnson et al. (2006) for the problem of word segmentation. We discuss our framework in Section 6.4 and summarize in 6.5.

Some of the work in this chapter has been described in Cohen et al. (2010).

6.1 Adaptor Grammars

We review adaptor grammars and develop a stick-breaking representation of the tree distribution.

6.1.1 Definition of Adaptor Grammars

Adaptor grammars capture syntactic regularities in sentences by placing a nonparametric prior over the distribution of syntactic trees that underlie them. The model exhibits "rich get richer" dynamics: once a tree is generated, it is more likely to reappear.

Adaptor grammars were developed by Johnson et al. (2006). An adaptor grammar is a tuple $\boldsymbol{A} = \langle \mathbf{G}, \mathcal{M}, \boldsymbol{a}, \boldsymbol{b}, \boldsymbol{\alpha} \rangle$, which contains: (i) a context-free grammar $\mathbf{G} = \langle \mathcal{W}, \mathcal{N}, \boldsymbol{R}, S \rangle$ where $\mathcal{W}$ is the set of terminals, $\mathcal{N}$ is the set of nonterminals, $\boldsymbol{R}$ is a set of production rules, and $S \in \mathcal{N}$ is the start symbol—we denote by $\boldsymbol{R}_A$ the subset of $\boldsymbol{R}$ with left-hand side A; (ii) a set of adapted nonterminals, $\mathcal{M} \subseteq \mathcal{N}$; and (iii) parameters $\boldsymbol{a}$, $\boldsymbol{b}$ and $\boldsymbol{\alpha}$, which are described below.

An adaptor grammar assumes the following generative process of trees. First, the multinomial distributions $\boldsymbol{\theta}$ for a PCFG based on $\mathbf{G}$ are drawn from Dirichlet

distributions. Specifically, multinomial $\boldsymbol{\theta}_A \sim \mathrm{Dir}(\alpha_A)$ where $\boldsymbol{\alpha}$ is collection of Dirichlet parameters, indexed by $A \in \mathcal{N}$.

Trees are then generated top-down starting with S. Any non-adapted nonterminal $A \in \mathcal{N} \setminus \mathcal{M}$ is expanded by drawing a rule from $\boldsymbol{R}_A$. There are two ways to expand $A \in \mathcal{M}$:

1. With probability $(n_z - b_A)/(n_A + a_A)$ we expand A to subtree z (a tree rooted at A with a yield in $\mathcal{W}^*$), where n_z is the number of times the tree z was previously generated and n_A is the total number of subtrees (tokens) previously generated root being A. We denote by $\boldsymbol{a}$ the *concentration parameters* and $\boldsymbol{b}$ the *discount parameters*, both indexed by $A \in \mathcal{M}$. We have $a_A > -b_A$ and $b_A \in [0, 1]$.
2. With probability $(a_A + k_A b_A)/(n_A + a_A)$, A is expanded as in a PCFG by a draw from $\boldsymbol{\theta}_A$ over $\boldsymbol{R}_A$, where k_A is the number of subtrees (types) previously generated with root A.

For the expansion of adapted nonterminals, this process can be explained using the Chinese restaurant process (CRP) metaphor: a "customer" (corresponding to a partially generated tree) enters a "restaurant" (corresponding to a nonterminal) and selects a "table" (corresponding to a subtree) to attach to the partially generated tree. If she is the first customer at the table, the PCFG $\langle \mathbf{G}, \boldsymbol{\theta} \rangle$ produces the new table's associated "dish" (a subtree). We note that our construction deviates from the strict definition of adaptor grammars (Johnson et al., 2006): (i) in our construction, we assume (as prior work does in practice) that the adaptors in $\boldsymbol{A} = \langle \mathbf{G}, \mathcal{M}, \boldsymbol{a}, \boldsymbol{b}, \boldsymbol{\alpha} \rangle$ follow the Pitman-Yor (PY) process (Pitman and Yor, 1997), though in general other stochastic processes might be used; and (ii) we place a symmetric Dirichlet over the parameters of the PCFG, $\boldsymbol{\theta}$, whereas Johnson et al. used a fixed PCFG for the definition (though in their actual experiments they used a Dirichlet prior).

When adaptor grammars are defined using the CRP, the PCFG $\mathbf{G}$ has to be non-recursive with respect to the adapted nonterminals. More precisely, for $A \in \mathcal{N}$, denote by $\mathrm{Reachable}(\boldsymbol{G}, A)$ all the nonterminals that can be reached from A using a partial derivation from $\mathbf{G}$. Then we restrict $\mathbf{G}$ such that for all $A \in \mathcal{M}$, we have $A \notin \mathrm{Reachable}(\boldsymbol{G}, A)$. Without this restriction, we might end up in a situation where the generative process is ill-defined: in the CRP terminology, a customer could enter a restaurant and select a table whose dish is still in the process of being selected. Consider the simple grammar with rules $\{$ S $\rightarrow$ S S, S $\rightarrow$ a $\}$. Assume that a customer enters the restaurant for S. She sits at a table, and selects a dish, a subtree, which starts with the rule S $\rightarrow$ S S. Perhaps the first

child S is expanded by S $\rightarrow$ a. For the second child S, it is possible to re-enter the "S restaurant" and choose the first table, where the "dish" subtree is still being generated. In the more general form of adaptor grammars with arbitrary adaptors, the problem amounts to mutually dependent definitions of distributions which rely on the others to be defined. We return to this problem in Section 6.2.2.

Inference The inference problem is to compute the posterior distribution of parse trees given observed sentences $\boldsymbol{x} = \langle x_1, \ldots, x_n \rangle$. Typically, inference with adaptor grammars is done with Gibbs sampling. Johnson et al. (2006) use an embedded Metropolis-Hastings sampler (Robert and Casella, 2005) inside a Gibbs sampler, because it is intractable to compute the distribution over trees for a sentence in the corpus conditioning on other trees being observed. The proposal distribution is a PCFG, resembling a tree substitution grammar (TSG; Joshi, 2003). The sampler of Johnson et al. is based on the representation of the PY process as a distribution over partitions of integers. This representation is not amenable to variational inference.

In the empirical Bayesian setting, we can also estimate the hyperparameters of adaptor grammars, more specifically, estimate $\boldsymbol{a}$, $\boldsymbol{b}$ and $\boldsymbol{\alpha}$. Johnson and Goldwater (2009) estimate these hyperparameters by using an MCMC method with a vague prior.

6.1.2 Stick-Breaking Representation

To develop a variational inference algorithm for adaptor grammars, we require an alternative representation of the model in Section 6.1.1. The CRP-based definition implicitly marginalizes out a random distribution over trees. For variational inference, we construct that distribution.

We first review the Dirichlet process and its stick-breaking representation (Sethuraman, 1994). The Dirichlet process defines a distribution over distributions. Samples from the Dirichlet process tend to deviate from a *base distribution* depending on a *concentration parameter*. Let $G \sim \mathrm{DP}(G_0, a)$ be a distribution sampled from the Dirichlet process with base distribution G_0 and concentration parameter a. The distribution G is discrete, which means it puts positive mass on a countable number of atoms drawn from G_0. Repeated draws from G exhibit the "clustering property," which means that they will be assigned to the same value with positive probability. Thus, they exhibit a partition structure. Marginalizing out G, the distribution of that partition structure is given by a CRP with parameter

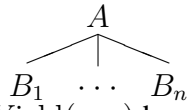
a (Pitman, 2002).

The stick-breaking process gives a constructive definition of G (Sethuraman, 1994). We describe this construction for the Pitman-Yor process, which is an extension of the Dirichlet process. With the stick-breaking process, we first sample "stick lengths" $\boldsymbol{\pi} \sim \mathrm{GEM}(a, b)$ (in the case of Dirichlet process, we have $b = 0$). The GEM partitions the interval $[0, 1]$ into countably many segments. First, draw $v_i \sim \mathrm{Beta}(1 - b, a + ib)$ for $i \in \{1, \ldots\}$. Then, define $\pi_i \triangleq v_i \prod_{j=1}^{i-1}(1 - v_j)$. In addition, we also sample infinitely many "atoms" independently $z_i \sim G_0$. Define G as:

$$G(z) = \sum_{i=1}^{\infty} \pi_i \delta(z_i, z)$$

where $\delta(z_i, z)$ is 1 if $z_i = z$ and 0 otherwise. It can be shown that this random distribution G is drawn from a Pitman-Yor process. Notice the discreteness of G is laid bare in the stick-breaking construction.

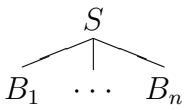
With the stick-breaking representation in hand, we turn to a constructive definition of the distribution over trees given by an adaptor grammar. Let $A_1, \ldots, A_K$ be an enumeration of the nonterminals in $\mathcal{M}$ which satisfies: $i \leq j \Rightarrow A_j \notin \mathrm{Reachable}(\boldsymbol{G}, A_i)$. (That this exists follows from the assumption about the lack of recursiveness of adapted nonterminals.) Let $\mathrm{Yield}(z)$ be the yield of a tree derivation z. The process that generates observed sentences $\boldsymbol{x} = \langle x_1, \ldots, x_n \rangle$ from the adaptor grammar $\boldsymbol{A} = \langle \mathbf{G}, \mathcal{M}, \boldsymbol{a}, \boldsymbol{b}, \boldsymbol{\alpha} \rangle$ is as follows:

1. For each $A \in \mathcal{N}$, draw $\boldsymbol{\theta}_A \sim \mathrm{Dir}(\alpha_A)$.
2. (Construct grammar) For A from A_1 to A_K, define G_A as follows:
   (a) Draw $\boldsymbol{\pi}_A \mid a_A, b_A \sim \mathrm{GEM}(a_A, b_A)$.
   (b) For $i \in \{1, \ldots\}$, grow a tree $z_{A,i}$ as follows:
      i. Draw $A \rightarrow B_1 \ldots B_n$ from $\boldsymbol{R}_A$.
      ii. $z_{A,i} =$ 

      iii. While $\mathrm{Yield}(z_{A,i})$ has nonterminals:
         A. Choose an unexpanded nonterminal B from yield of $z_{A,i}$.
         B. If $B \in \mathcal{M}$, expand B according to G_B (defined on previous iterations of step 2).
         C. If $B \in \mathcal{N} \setminus \mathcal{M}$, expand B with a rule from $\boldsymbol{R}_B$ according to $\mathrm{Mult}(\boldsymbol{\theta}_B)$.
   (c) For $i \in \{1, \ldots\}$, define $G_A(z_{A,i}) = \pi_{A,i}$
3. (Sample corpus) For $i \in \{1, \ldots, n\}$ draw z_i as follows:

(a) If $S \in \mathcal{M}$, draw $z_i \mid G_S \sim G_S$.

(b) If $S \notin \mathcal{M}$, draw z_i as in 2(b):

1. Draw $S \to B_1 \dots B_n$ from $\boldsymbol{R}_S$
2. $z_i =$ 

3. While $\mathrm{Yield}(z_i)$ has nonterminals:
   (a) Choose an unexpanded nonterminal B from yield of $z_{A,i}$
   (b) If $B \in \mathcal{M}$, expand B according to G_B.
   (c) If $B \in \mathcal{N} \backslash \mathcal{M}$, expand B with a rule from $\boldsymbol{R}_B$ according to $\mathrm{Mult}(\boldsymbol{\theta}_B)$.

4. Set $x_i = \mathrm{Yield}(z_i)$ for $i \in \{1, \dots, n\}$.

Here, there are four collections of hidden variables: the PCFG multinomials $\boldsymbol{\theta} = \{\boldsymbol{\theta}_A \mid A \in \mathcal{N}\}$, the stick length proportions $\boldsymbol{v} = \{\boldsymbol{v}_A \mid A \in \mathcal{M}\}$ where $\boldsymbol{v}_A = \langle v_{A,1}, v_{A,2}, \dots \rangle$, the adapted nonterminals' subtrees $\boldsymbol{z}_A = \{z_{A,i} \mid A \in \mathcal{M}; i \in \{1, \dots\}\}$ and the derivations $\boldsymbol{z}_{1:n} = z_1, \dots, z_n$. The symbol $\boldsymbol{z}$ refers to the collection of $\{\boldsymbol{z}_A \mid A \in \mathcal{M}\}$, and $\boldsymbol{z}_{1:n}$ refers to the derivations of the data $\boldsymbol{x}$.

Note that the distribution in 2(c) is defined with the GEM distribution, as mentioned earlier. It is a sample from the Pitman-Yor process, which is later used in 3(a) to sample trees for an adapted non-terminal.

6.2 Variational Inference

We now give a variational inference algorithm for adaptor grammars based on the stick-breaking process we described. For general explanation about variational inference, see Section 5.3.

The variational bound on the likelihood of the data is:

$$\begin{aligned}\log p(\boldsymbol{x} \mid \boldsymbol{a}, \boldsymbol{\alpha}) \geq & H(q) + \sum_{A \in \mathcal{M}} \mathbb{E}_q[\log p(\boldsymbol{v}_A \mid a_A)] + \sum_{A \in \mathcal{M}} \mathbb{E}_q[\log p(\boldsymbol{\theta}_A \mid \alpha_A)] \\ & + \sum_{A \in \mathcal{M}} \mathbb{E}_q[\log p(\boldsymbol{z}_A \mid \boldsymbol{v}, \boldsymbol{\theta})] + \mathbb{E}_q[\log p(\boldsymbol{z}, \boldsymbol{x} \mid \boldsymbol{v}_A)] \qquad (6.1)\end{aligned}$$

Expectations are taken with respect to the variational distribution $q(\boldsymbol{v}, \boldsymbol{\theta}, \boldsymbol{z})$ and $H(q)$ is its entropy.

Before tightening the bound, we define the functional form of the variational distribution. We use the mean-field distribution in which all of the hidden variables are independent and governed by individual variational parameters. (Note that in the true posterior, the hidden variables are highly coupled.) To account for the infinite collection of random variables, for which we cannot define a variational distribution, we use the truncated stick distribution (Blei and Jordan, 2005). Hence, we assume that, for all $A \in \mathcal{M}$, there is some value N_A such that $q(v_{A,N_A} = 1) = 1$. The assigned probability to parse trees in the stick will be 0 for $i > N_A$, so we can ignore $z_{A,i}$ for $i > N_A$. This leads to a factorized variational distribution:

$$q(\boldsymbol{v}, \boldsymbol{\theta}, \boldsymbol{z}) = \prod_{A \in \mathcal{M}} \left(q(\boldsymbol{\theta}_A) \prod_{i=1}^{N_A} q(v_{A,i}) \times q(z_{A,i}) \right) \times \prod_{i=1}^{n} q(z_i)$$

It is natural to define the variational distributions over $\boldsymbol{\theta}$ and $\boldsymbol{v}$ to be Dirichlet distributions with parameters $\boldsymbol{\tau}_A$ and Beta distributions with parameters $\boldsymbol{\gamma}_{A,i}$, respectively. The two distributions over trees, $q(z_{A,i})$ and $q(z_i)$, are more problematic. For example, with $q(z_i \mid \boldsymbol{\phi})$, we need to take into account different subtrees that could be generated by the model and use them with the proper probabilities in the variational distribution $q(z_i \mid \boldsymbol{\phi})$. We follow and extend the idea from Johnson et al. (2006) and use *grammatons* for these distributions. Grammatons are "mini-grammars," inspired by the grammar $\mathbf{G}$.

For two strings in $s, t \in \mathcal{W}^*$, we use "$t \subseteq s$" to mean that t is a substring of s. In that case, a grammaton is defined as follows:

Definition 6.1 *Let $\boldsymbol{A} = \langle \mathbf{G}, \mathcal{M}, \boldsymbol{a}, \boldsymbol{b}, \boldsymbol{\alpha} \rangle$ be an adaptor grammar with $\mathbf{G} = \langle \mathcal{W}, \mathcal{N}, \boldsymbol{R}, S \rangle$. Let s be a finite string over the alphabet of $\mathbf{G}$ and $A \in \mathcal{N}$. Let $\mathcal{U}$ be the set of nonterminals $\mathcal{U} \triangleq \mathrm{Reachable}(\boldsymbol{G}, A) \cap (\mathcal{N} \setminus \mathcal{M})$. The grammaton $\boldsymbol{G}(A, s)$ is the context-free grammar with the start symbol A and the rules*

$$R_A \cup \left(\bigcup_{B \in \mathcal{U}} R_B \right) \cup \bigcup_{A \to B_1 \ldots B_n \in R_A} \bigcup_{i \in \{i \mid B_i \in \mathcal{M}\}} \{B_i \to t \mid t \subseteq s\}.$$

Using a grammaton, we define the distributions $q(z_{A,i} \mid \boldsymbol{\phi}_A)$ and $q(z_i \mid \boldsymbol{\phi})$. This requires a preprocessing step (described in detail in Section 6.2.4) that defines, for each $A \in \mathcal{M}$, a list of strings $\boldsymbol{s}_A = \langle s_{A,1}, \ldots, s_{A,N_A} \rangle$. Then, for $q(z_{A,i} \mid \boldsymbol{\phi}_A)$ we use the grammaton $\boldsymbol{G}(A, s_{A,i})$ and for $q(z_i \mid \boldsymbol{\phi})$ we use the grammaton $\boldsymbol{G}(A, x_i)$ where x_i is the ith observed sentence. We parametrize the

grammaton with weights $\boldsymbol{\phi}_A$ (or $\boldsymbol{\phi}$) for each rule in the grammaton. The selection of these variational distributions makes the variational distributions over the trees for strings $\boldsymbol{s}$ (and trees for $\boldsymbol{x}$) globally normalized weighted grammars. Choosing such distributions is motivated by their ability to make the variational bound tight (similar to Cohen et al., 2008, and Cohen and Smith, 2009 and Chapter 5). In practice we do not have to use rewrite rules for all strings $t \subseteq s$ in the grammaton. It suffices to add rewrite rules only for the strings $t = s_{A,i}$ that have some grammaton attached to them, $\boldsymbol{G}(A, s_{A,i})$.

The variational distribution above yields a variational inference algorithm for approximating the posterior by estimating $\boldsymbol{\gamma}_{A,i}$, $\boldsymbol{\tau}_A$, $\boldsymbol{\phi}_A$ and $\boldsymbol{\phi}$ iteratively, given a fixed set of hyperparameters $\boldsymbol{a}$, $\boldsymbol{b}$ and $\boldsymbol{\alpha}$. Let r be a PCFG rule. Let $\tilde{f}(r, s_{B,k}) = \mathbb{E}_{q(z_k|\boldsymbol{\phi}_{B,k})}[f(r; z_k)]$, where $f(r; z_k)$ counts the number of times that rule r is applied in the derivation z_k. Whenever a rule r does not appear in the grammaton for nonterminal B with the string s_k, we let $\tilde{f}(r, s_{B,k}) = 0$. Let $A \to \beta$ denote a rule from $\mathbf{G}$. The quantity $\tilde{f}(r, s_{B,k})$ is computed using the inside-outside (IO) algorithm. Figure 6.1 gives the variational inference updates.

We use variational EM (Section 5.3.1) to fit the hyperparameters. The E step performs the variational inference mentioned above (Figure 6.1). The M-step optimizes the hyperparameters ($\boldsymbol{a}$, $\boldsymbol{b}$ and $\boldsymbol{\alpha}$) with respect to expected sufficient statistics under the variational distribution. We use Newton-Raphson for each (Boyd and Vandenberghe, 2004); Figure 6.2 gives the objectives.

6.2.1 More Details about the Derivation of the Algorithm

This section can be skimmed over on a first reading. Its goal is to give an intuition on the Equations in Figure 6.1 and Figure 6.2. We first consider the entropy term in Equation 6.1. This unfolds to the following:

$$
\begin{aligned}
H(q) &= \sum_{A \in \mathcal{M}} \left(H(q(\boldsymbol{\theta}_A)) \sum_{i=1}^{N_A} H(q(v_{A,i})) \times H(q(z_{A,i})) \right) \times \sum_{i=1}^{n} H(q(z_i)) \\
&= \sum_{A \in \mathcal{M}} (\log(|R_A|\Gamma(\alpha_A)) - \log\Gamma(|R_A|\alpha_A) + (|R_A|\alpha_A - |R_A|)\Psi(|R_A|\alpha_A) \\
&\quad -|R_A|(\alpha_A - 1)\Psi(\alpha_A) + \sum_{i=1}^{N_A} (\log\Gamma(\gamma^1_{A,i}) + \log\Gamma(\gamma^2_{A,i}) \\
&\quad - \log\Gamma(\gamma^1_{A,i} + \gamma^2_{A,i}) + (\gamma^1_{A,i} + \gamma^2_{A,i} - 2)\Psi(\gamma^1_{A,i} + \gamma^2_{A,i}) + H(q(z_{A,i})))) \\
&\quad + \sum_{i=1}^{n} H(q(z_i))
\end{aligned} \tag{6.2}
$$

Consider the variational bound on the log-likelihood in Equation 6.1. Consider a single term $\mathbb{E}_q[\log p(\boldsymbol{z}_{A,i} \mid \boldsymbol{v}, \boldsymbol{\theta})]$ for some $A \in \mathcal{M}$ and $i \leq N_A$. Then, using the notation from Definition 6.1 we note that it actually has the following form:

$$
\begin{aligned}
&\mathbb{E}_q[\log p(\boldsymbol{z}_{A,i} \mid \boldsymbol{v}, \boldsymbol{\theta})] \\
&= \sum_{A \to \beta \in R_A} \tilde{f}(A \to \beta, s_{A,i}) \times \mathbb{E}_q[\log \theta_{A \to \beta}] \\
&+ \sum_{B \in \mathcal{U}} \sum_{B \to \beta \in R_B} \tilde{f}(B \to \beta, s_{A,i}) \times \mathbb{E}_q[\log \theta_{B \to \beta}] \\
&+ \sum_{B \in \mathcal{M}} \sum_{j=1}^{N_B} \tilde{f}(A \to s_{A,i}, s_{B,j}) \times \left(\sum_{i'=1}^{i-1} \mathbb{E}_q[\log(v_{A,i'})] + \mathbb{E}_q[\log(1 - v_{A,i})] \right)
\end{aligned} \tag{6.3}
$$

This is a direct result of the fact that the stick-breaking process of adaptor grammars is parameterized in such a way that rules in R_A and R_B for $B \in \mathcal{U}$ use the probabilities in $\boldsymbol{\theta}$ and the rules that rewrite to strings use the probabilities from the stick-breaking parameters $\boldsymbol{v}$.

Now, also note that for all $A \in \mathcal{N}$ and i we have:

$$
\begin{aligned}
\mathbb{E}_q[\log \theta_{A \to \beta}] &= \Psi(\tau_{A \to \beta}) - \Psi(\sum_{A \to \beta} \tau_{A \to \beta}) \\
\mathbb{E}_q[v_{A,i}] &= \Psi(\gamma^1_{A,i}) - \Psi(\gamma^1_{A,i} + \gamma^2_{A,i})
\end{aligned}
$$

In addition, we also have that the following terms in Equation 6.1 can be unfolded:

$$\begin{aligned}
&\mathbb{E}_q[\log p(\boldsymbol{v}_A \mid a_A)] \qquad (6.4)\\
&= \sum_{i=1}^{N_A} a_A \left(\Psi(\gamma^2_{A,i}) - \Psi(\gamma^1_{A,i} + \gamma^2_{A,i})\right) + \log\Gamma(a_A + 1 + ib_A)\\
&+ ib_A \left(\Psi(\gamma^2_{A,i}) - \Psi(\gamma^1_{A,i} + \gamma^2_{A,i})\right) - \log\Gamma(ib_A + a_A) - \log\Gamma(1 - b_A)\\
&\mathbb{E}_q[\log p(\boldsymbol{\theta}_A \mid \alpha_A)] \qquad (6.5)\\
&= \log\Gamma(|R_A|\alpha_A) - |R_A|\log\Gamma(\alpha_A)\\
&+ (\alpha_A - 1)\left(\textstyle\sum_{A\to\beta\in R_A} \Psi(\tau_{A\to\beta}) - \Psi\left(\sum_{A\to\beta\in R_A} \tau_{A\to\beta}\right)\right)
\end{aligned}$$

We turn now to consider the updates for the E-step (Figure 6.1). First, consider the update for $\gamma^1_{A,i}$ and $\gamma^2_{A,i}$. The only terms that depend in the variational bound in these two quantities are $H(q(v_{A,i}))$ and $\mathbb{E}_q[\log p(\boldsymbol{z}_{B,j} \mid \boldsymbol{v}, \boldsymbol{\theta})]$ (for B and j in which A appears in the corresponding grammaton and $s_{A,i}$ is a substring of $s_{B,j}$) and also $\mathbb{E}_q[\log p(\boldsymbol{v}_A \mid a_A)]$. Accumulating all terms that depend on these two quantities (from Equation 6.5, Equation 6.4 and Equation 6.2) leads to the following expression:

$$\begin{aligned}
L(\gamma^1_{A,i}, \gamma^2_{A,i}) &= \log\Gamma(\gamma^1_{A,i}) + \log\Gamma(\gamma^2_{A,i}) - \log\Gamma(\gamma^1_{A,i} + \gamma^2_{A,i})\\
&+ (\gamma^1_{A,i} + \gamma^2_{A,i} - 2 + a_A + ib_A)\Psi(\gamma^1_{A,i} + \gamma^2_{A,i})\\
&+ (\gamma^1_{A,i} + a_A - 1)\Psi(\gamma^1_{A,i}) + (\gamma^2_{A,i} + ib_A - 1)\Psi(\gamma^2_{A,i})\\
&\sum_{B\in\mathcal{M}}\sum_{j=1}^{N_B} \tilde{f}(A \to s_{A,i}, s_{B,j}) \times \left(\sum_{i'=1}^{i-1} \mathbb{E}_q[\log(v_{A,i'})] + \mathbb{E}_q[\log(1 - v_{A,i})]\right)
\end{aligned} \qquad (6.6)$$

We can unfold Equation 6.6 further using:

$$\begin{aligned}
\mathbb{E}_q[\log(v_{A,i'})] &= \Psi(\gamma^1_{A,i}) - \Psi(\gamma^1_{A,i} + \gamma^2_{A,i})\\
\mathbb{E}_q[\log(1 - v_{A,i'})] &= \Psi(\gamma^2_{A,i}) - \Psi(\gamma^1_{A,i} + \gamma^2_{A,i})
\end{aligned}$$

Taking the derivative of Equation 6.6 and equating to zero will lead to the update in Equation 6.7 and Equation 6.8. This is similar to Blei et al. (2003). The

process for deriving the updates for $\tau_{\bullet,A\to\beta}$ is similar, only now instead of giving treatment to the Beta distributions for $v_{A,i}$, we give a treatment to the Dirichlet distributions for $\boldsymbol{\theta}_A$ (which are a generalization of the Beta distribution). Following a similar derivation, we will get the updates from Equation 6.9.

The updates for $\boldsymbol{\phi}_A$ and $\boldsymbol{\phi}$ originate in a result which is similar to Lemma 5.2. They are also very similar to the updates one gets with variational Bayesian EM for PCFGs when using a Dirichlet prior. The main difference is that now the rules $A \to s_{A,i}$ are controlled using probabilities in $\boldsymbol{v}_A$, where the probability of rule $A \to s_{A,i}$ is:

$$(1 - v_{A,i}) \times (\prod_{i'=1}^{i-1} v_{A,i'})$$

The updates for the M-step (Figure 6.2) are simpler to derive. They are based on taking the relevant terms to the updated quantity from Equation 6.1. These terms appear in Equation 6.6, Equation 6.4 and Equation 6.3.

6.2.2 Note about Recursive Grammars

With recursive grammars, the stick-breaking process representation gives probability mass to events which are ill-defined. In step 2(iii)(c) of the stick-breaking representation, we assign nonzero probability to an event in which we choose to expand the current tree using a subtree with the same index that we are currently still expanding. In short, with recursive grammars, we can get "loops" inside the trees.

We would still like to use recursion in the cases which are not ill-defined. In the case of recursive grammars, there is no problem with the stick-breaking representation and the order by which we enumerate the nonterminals. This is true because the stick-breaking process separates allocating the probabilities for each index in the stick and allocating the atoms for each index in the stick.

Our variational distributions give probability 0 to any event which is ill-defined in the sense mentioned above. Optimizing the variational bound in this case is equivalent to optimizing the same variational bound with a model p' that (i) starts with p, (ii) assigns probability 0 to ill-defined events, and (iii) renormalizes:

Proposition 6.2 *Let $p(\boldsymbol{x}, \boldsymbol{z})$ be a probability distribution, where $z \in \mathcal{Z}$, and let $\mathcal{S} \subset \mathcal{Z}$. Let $Q = \{q \mid q(z) = 0, \forall z \in \mathcal{S}\}$, a set of distributions. Then:*

$$\operatorname*{argmax}_{q \in Q} \mathbb{E}_q[\log p(\boldsymbol{x}, \boldsymbol{z})] = \operatorname*{argmax}_{q} \mathbb{E}_q[\log p'(\boldsymbol{x}, \boldsymbol{z})]$$

$$\gamma^1_{A,i} = 1 - b_A + \textstyle\sum_{B \in \mathcal{M}} \sum_{k=1}^{N_B} \tilde{f}(A \to s_{A,i}, s_{B,k}) \tag{6.7}$$

$$\gamma^2_{A,i} = a_A + ib_A + \textstyle\sum_{j=1}^{i-1} \sum_{B \in \mathcal{M}} \sum_{k=1}^{N_B} \tilde{f}(A \to s_{A,j}, s_{B,k}) \tag{6.8}$$

$$\tau_{\bullet, A \to \beta} = \alpha_A + \textstyle\sum_{B \in \mathcal{M}} \sum_{k=1}^{N_B} \tilde{f}(A \to \beta, s_{B,k}) \tag{6.9}$$

$$\phi_{\bullet, A \to s_{A,i}} = \Psi(\gamma^1_{A,i}) - \Psi(\gamma^1_{A,i} + \gamma^2_{A,i}) + \textstyle\sum_{j=1}^{i-1} \left(\Psi(\gamma^2_{A,i}) - \Psi(\gamma^1_{A,i} + \gamma^2_{A,i}) \right)$$

$$\phi_{\bullet, A \to \beta} = \Psi(\tau_{\bullet, A \to \beta}) - \Psi\left(\textstyle\sum_{\beta} \tau_{\bullet, A \to \beta} \right)$$

Figure 6.1: Updates for variational inference with adaptor grammars. Ψ is the digamma function. Note that in Equation 6.9, if $A \in \mathcal{M}$, then the sum over nonterminals include a single summand for nonterminal A.

$$\max_{\alpha_A} \log \Gamma(|R_A|\alpha_A) - |R_A| \log \Gamma(\alpha_A) + (\alpha_A - 1) \left(\textstyle\sum_{A \to \beta \in R_A} \Psi(\tau_{A \to \beta}) - \Psi\left(\sum_{A \to \beta \in R_A} \tau_{A \to \beta} \right) \right)$$

$$\max_{a_A} \textstyle\sum_{i=1}^{N_A} a_A \left(\Psi(\gamma^2_{A,i}) - \Psi(\gamma^1_{A,i} + \gamma^2_{A,i}) \right) + \log \Gamma(a_A + 1 + ib_A) - \log \Gamma(ib_A + a_A)$$

$$\max_{b_A} \textstyle\sum_{i=1}^{N_A} ib_A \left(\Psi(\gamma^2_{A,i}) - \Psi(\gamma^1_{A,i} + \gamma^2_{A,i}) \right) + \log \Gamma(a_A + 1 + ib_A) - \log \Gamma(1 - b_A) - \log \Gamma(ib_A + a_A)$$

Figure 6.2: Variational M-step updates. Γ is the gamma function.

where $p'(\boldsymbol{x}, \boldsymbol{z})$ is a probability distribution defined as $p'(\boldsymbol{x}, \boldsymbol{z}) = p(\boldsymbol{x}, \boldsymbol{z}) / \sum_{\boldsymbol{z} \in \mathcal{S}} p(\boldsymbol{x}, \boldsymbol{z})$ for $\boldsymbol{z} \in \mathcal{S}$ and 0 otherwise.

For this reason, our variational approximation allows the use of recursive grammars. The use of recursive grammars with MCMC methods is problematic, since it has no corresponding probabilistic interpretation, enabled by zeroing events that are ill-defined in the variational distribution. Deciding whether a similar interpretation holds when using the sampling algorithm of Johnson et al. (2006) remains an open problem.

6.2.3 Time Complexity

The algorithm in Johnson et al. (2006) works by sampling from a PCFG containing rewrite rules that rewrite to a whole tree fragment. This requires a procedure that uses the inside-outside algorithm. Despite the grammar being bigger (because of the rewrite rules to a string), the asymptotic complexity of the IO algorithm stays $O(|\mathcal{N}|^2|x_i|^3 + |\mathcal{N}|^3|x_i|^2)$ where $|x_i|$ is the length of the ith sentence.[1]

Our algorithm requires running the IO algorithm for each yield in the variational distribution, for each nonterminal, and for each sentence. However, IO runs with much smaller grammars coming from the grammatons. The cost of running the IO algorithm on the yields in the sticks for $A \in \mathcal{M}$ can be taken into account parsing a string that appears in the corpus with the full grammars. This leads to an asymptotic complexity of $O(|\mathcal{N}|^2|x_i|^3 + |\mathcal{N}|^3|x_i|^2)$ for the ith sentence in the same corpus each iteration.

Asymptotically, both sampling and variational EM behave the same. However, there are different constants that hide in these asymptotic runtimes: the number of iterations that the algorithm takes to converge (for which variational EM generally has an advantage over sampling) and the number of additional rewrite rules that rewrite to a string representing a tree (for which MCMC has a relative advantage, because it does not use a fixed set of strings; instead, the size of the grammars it uses grow as sampling proceeds). In Section 6.3, we see that variational EM and sampling methods are similar in the time it takes to complete because of a trade-off between these two constants. Simple parallelization, however, which is possible only with variational inference, provides significant speed-ups.[2]

A note about the memory requirements of the sampling algorithm and the variational Bayes algorithm is in order. The ability to parallelize the adaptor grammar variational EM algorithm comes with a trade-off as far as memory requirement goes. While with sampling, we require to maintain a single PCFG (which may contain nonterminals rewriting to whole strings) for the proposal distribution of the Metropolis-Hastings sampler, with the variational EM algorithm we require to maintain weighted CFGs, corresponding to variational distribution, for each of the strings we use in the non-parametric stick. This poses a heavy memory requirement, if N_A is large for some non-terminal A. However, note that the weighted

[1] This analysis is true for CNF grammars augmented with rules rewriting to a whole string, like those used in our study.

[2] Newman et al. (2009) show how to parallelize sampling algorithms, but in general, parallelizing these algorithms is more complicated than parallelizing variational algorithms and requires further approximation.

CFG variational distributions tend to require less memory, each separately, than the PCFG used for the proposal distribution with sampling. The reason for that is that the variational distributions tend to dominate shorter strings than whole sentences. In addition, the variational weighted CFGs do not always consist of all of the original PCFG rules, because their start symbol is a nonterminal which does not necessarily reach all nonterminals in the original PCFG.

6.2.4 Heuristics for Variational Inference

For the variational approximation from Section 6.2, we need to decide on a set of strings, $s_{A,i}$ (for $A \in \mathcal{M}$ and $i \in \{1, \ldots, N_A\}$) to define the grammatons in the nonparametric stick. Any set of strings will give a valid approximation, but to make the variational approximation as accurate as possible, we require that: (i) the strings in the set must be likely to be generated using the adaptor grammar as constituents headed by the relevant nonterminal, and (ii) strings that are more likely to be generated should be associated with a lower index in the stick. The reason for the second requirement is the exponential decay of coefficients as the index increases.

We show that a simple heuristic leads to an order over the strings generated by the adaptor grammars that yields an accurate variational estimation. We begin with a weighted context-free grammar $\boldsymbol{G}_{\text{heur}}$ that has the same rules as in $\mathbf{G}$, only the weight for all of its rules is 1. We then compute the quantity:

$$c(A, s) = \frac{1}{n} \left(\sum_{i=1}^{n} \mathbb{E}_{\boldsymbol{G}_{\text{heur}}}[f_i(z; A, s)] \right) - \rho \log |s|$$

where $f_i(z; A, s)$ is a function computing the count of constituents headed by A with yield s in the tree z for the sentence x_i. This quantity can be computed by using the IO algorithm on $\boldsymbol{G}_{\text{heur}}$. The term $\rho \log |s|$ is subtracted to avoid preference for shorter constituents, similar to Mochihashi et al. (2009).

While computing $c(A, s)$ using the IO algorithm, we sort the set of all substrings of s according to their expected counts (aggregated over all strings s). Then, we use the top N_A strings in the sorted list for the grammatons of A. The requirement to select N_A in advance is strict. We experimented with dynamic expansions of the stick, in the spirit of Kurihara et al. (2006) and Wang and Blei (2009), but we did not achieve better performance and it had an adverse effect on runtime. For completeness, we give these results in Section 6.3.

6.2.5 Decoding

The variational inference algorithm gives a distributions over parameters and hidden structures (through the grammatons). We experiment with two commonly used decoding methods: Viterbi decoding and minimum Bayes risk decoding (MBR; Goodman, 1996).

To parse a string with Viterbi (or MBR) decoding, we find the tree with highest score for the grammaton which is attached to that string. For all rules which rewrite to strings in the resulting tree, we again perform Viterbi (or MBR) decoding recursively using other grammatons.

(a) Unigram grammar
Sentence → Word$^+$
Word → Char$^+$

(b) Collocation grammar
Sentence → Colloc
Sentence → Colloc Sentence
Colloc → Word$^+$
Word → Char$^+$

(c) Syllable grammar
Sentence → Colloc3$^+$
Colloc3 → Colloc2$^+$
Colloc2 → Colloc1$^+$
Colloc1 → Word$^+$
Word → SyllableIF
Word → SyllableIF (Syllable) (Syllable) SyllableF
Syllable → Onset Rhyme
Onset → Consonant$^+$
Rhyme → Nucleus Coda
Nucleus → Vowel$^+$
Coda → Consonant$^+$
SyllableIF → OnsetI RhymeF
OnsetI → Consonant$^+$
RhymeF → Nucleus CodaF
CodaF → Consonant$^+$
SyllableI → OnsetI Rhyme
SyllableF → Onset RhymeF

Figure 6.3: The grammars tested for the segmentation task, from Johnson and Goldwater (2009). (a) unigram grammar. (b) collocation grammar. (c) syllable grammar with collocations. For brevity, grammars are represented compactly using regular expressions in the right hand side, though in practice they are regular PCFGs. All grammars originally appear in (Johnson and Goldwater, 2009).

		this paper		Johnson and Goldwater (2009)	
grammar	model	Viterbi	MBR	SA	MM
$\mathbf{G}_{\text{Unigram}}$	Dirichlet	0.49	**0.84**	0.57	0.54
	Pitman-Yor	0.49	**0.84**	0.81	0.75
	Pitman-Yor+inc	0.42	0.59	-	-
$\mathbf{G}_{\text{Colloc}}$	Dirichlet	0.40	**0.86**	0.75	0.72
	Pitman-Yor	0.40	**0.86**	0.83	**0.86**
	Pitman-Yor+inc	0.43	0.60	-	-
$\mathbf{G}_{\text{Syllable}}$	Dirichlet	0.77	0.83	0.84	0.84
	Pitman-Yor	0.77	0.83	**0.89**	0.88
	Pitman-Yor+inc	0.75	0.76	-	-

Table 6.1: F_1 performance for word segmentation on the Brent corpus. Dirichlet stands for Dirichlet Process adaptor ($\boldsymbol{b} = 0$), Pitman-Yor stands for Pitman-Yor adaptor ($\boldsymbol{b}$ optimized), and Pitman-Yor+inc stands for Pitman-Yor with iteratively increasing N_A for $A \in \mathcal{M}$. Johnson and Goldwater (2009) are the results adapted from Johnson and Goldwater (2009); SA is sample average decoding, and MM is maximum marginal decoding.

6.3 Experiments with Word Segmentation

We follow the experimental setting of Johnson and Goldwater (2009), who present state-of-the-art results for inference with adaptor grammars using Gibbs sampling on a segmentation problem. With this segmentation problem, the task is to take as an input a stream of phonemic string representations, and segment them into phonemes corresponding to words. We use the standard Brent corpus (Brent and Cartwright, 1996), which includes 9,790 unsegmented phonemic representations of utterances of child-directed speech from the Bernstein-Ratner (1987) corpus. An utterance in the corpus can be "yuwanttulUk&tDIs", and the task is to segment it to "yu want tu lUk &t DIs" ("you want to look at this").

Johnson and Goldwater (2009) test three grammars for this segmentation task. The first grammar is a character unigram grammar ($\mathbf{G}_{\text{Unigram}}$). The second grammar is a grammar that takes into consideration collocations ($\mathbf{G}_{\text{Colloc}}$). The third grammar incorporates more prior knowledge about the syllabic structure of English ($\mathbf{G}_{\text{Syllable}}$). All grammars are given in Figure 6.3. Once an utterance is parsed, Word constituents denote segments.

The value of ρ (penalty term for string length) had little effect on our results and was fixed at $\rho = -0.2$. When N_A (number of strings used in the variational distributions) is fixed, we use $N_A = 15{,}000$. We report results using Viterbi and MBR decoding. Johnson and Goldwater (2009) experimented with two decoding methods, sample average (SA) and maximal marginal decoding (MM), which are closely related to Viterbi and MBR, respectively. With MM, we marginalize the tree structure, rather than the word segmentation induced, similar to MBR decoding. With SA, we compute the probability of a whole tree, by averaging its count in the samples, an approximation to finding the tree with highest probability, like Viterbi. During learning, we initialize each complex grammar parameters by the result of learning the less complex grammar in the hierarchy, i.e., we initialize $\mathbf{G}_{\text{Colloc}}$ using the results of $\mathbf{G}_{\text{Unigram}}$, and we initialize $\mathbf{G}_{\text{Syllable}}$ using the results of $\mathbf{G}_{\text{Colloc}}$.

Table 6.1 gives the results for our experiments. Notice that the results for the Pitman-Yor process and the Dirichlet process are similar. When inspecting the learned parameters, we noticed that the discount parameters ($\boldsymbol{b}$) learned by the variational inference algorithm for the Pitman-Yor process are very close to 0. In this case, the Pitman-Yor process is reduced to the Dirichlet process.

Similar to Johnson and Goldwater's comparisons, we see superior performance when using minimum Bayes risk over Viterbi decoding. Further notice that the variational inference algorithm obtains significantly superior performance for simpler grammars than Johnson et al., while performance using the syllable grammar is lower. The results also suggest that it is better to decide ahead on the set of strings available in the sticks, instead of working gradually and increase the size of the sticks as described in Section 6.2.4. We believe that the reason is that the variational inference algorithm settles in a trajectory that uses fewer strings, then fails to exploit the strings that are added to the stick later. Given that selecting N_A in advance is advantageous, we may inquire if choosing N_A to be too large can lead to degraded performance, because of fragmentation of the grammar. Figure 6.4 suggests it is not the case, and performance stays steady after N_A reaches a certain value.

One of the advantages of variational approximation over sampling methods is the ability to run for fewer iterations before convergence. For example, with $\mathbf{G}_{\text{Unigram}}$ convergence typically takes 40 iterations with variational inference, while Johnson and Goldwater (2009) ran their sampler for 2,000 iterations, for which 1,000 were for burning in. The inside-outside algorithm dominates the iteration's runtime, both for sampling and variational EM. Each iteration with sampling, however, takes less time, despite the asymptotic analysis in Section 6.2.3, be-

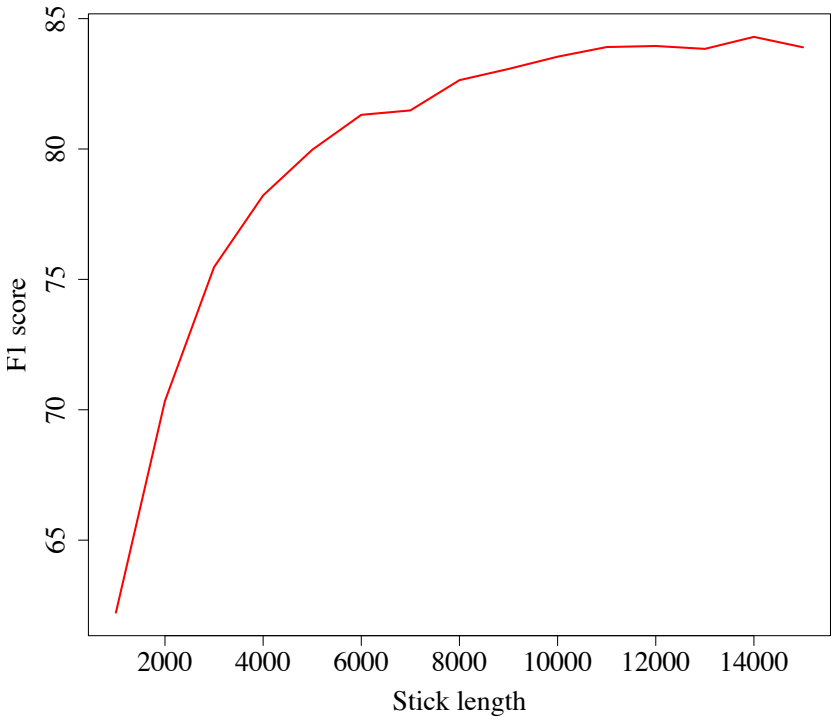


Figure 6.4: F_1 performance of $\mathbf{G}_{\text{Unigram}}$ as influenced by the length of the stick, N_{Word}.

cause of different implementations and the different number of rules that rewrite to a string. Figure 6.5 shows a plot of the negated log-likelihood (for sampling) and the free energy (for variational inference) for the two grammars: unigram and collocation. Note that the measurements of the log-likelihood for sampling are taken every *ten* iterations. We can see that indeed it takes fewer iterations for the variational inference algorithm to converge. In fact, even all through the two thousand iterations, the log-likelihood with sampling decreases, while the free energy with variational inference stays flat after about 25 iterations.

It is interesting to note that the free energy for the collocation grammar is larger than the free energy for the unigram grammar, while the opposite holds for the log-likelihood with sampling. While the model of collocation grammar indeed fits the data better (because it is more complex), with variational inference we need to take into account more elements in the nonparametric stick with the collocation grammar, making the free energy larger.

We now give a comparison of clock time for $\mathbf{G}_{\text{Unigram}}$ for variational inference and sampling as described in Johnson and Goldwater (2009).[3] Replicating the experiment in Johnson and Goldwater (first row in Table 6.1) took 2 hours and 14 minutes. With the variational approximation, we had the following: (i) the preprocessing (Section 6.2.4) step took 114 seconds; (ii) each iteration took approximately 204 seconds, with convergence after 40 iterations, leading to 8,160 seconds of pure variational EM processing; (iii) parsing took another 952 seconds. The total time is 2 hours and 34 minutes.

At first glance it seems that variational inference is slower than MCMC sampling. However, note that the cost of the grammar preprocessing step is amortized over all experiments with the specific grammar, and the E-step with variational inference can be parallelized, while sampling requires an update of a global set of parameters after each tree update. We ran our algorithm on a cluster of 20 1.86GHz CPUs and achieved a significant speed-up: preprocessing took 34 seconds, each variational EM iteration took 43 seconds and parsing took 208 seconds. The total time was 47 minutes, which is 2.8 times faster than sampling.

[3]We used the code and data available at `http://www.cog.brown.edu/~mj/Software.htm`. The machine used for this comparison is a 64-bit machine with 2.6GHz CPU, 1MB of cache memory and 16GB of RAM.

6.4 Discussion

We note that adaptor grammars are not limited to a selection of a Dirichlet distribution as a prior for the grammar rules. Our variational inference algorithm, for example, can be extended (perhaps at great computational expense) to use the logistic normal prior instead of the Dirichlet, as described in Chapter 5. We leave this for future work. We also believe that our variational inference makes it easier to add an additional infinite dimension to the grammar. The set of nonterminals can be extended to be grow nonparametrically, again using a Dirichlet process, similarly to the way it is presented in Liang et al. (2007) and Finkel et al. (2007).

In Chapter 8 we also experiment with a novel application for adaptor grammars: dependency grammar induction.

6.5 Summary

We described in this chapter a variational inference framework for adaptor grammars. Our variational inference framework is based on a novel stick-breaking representation for adaptor grammars. One big advantage of our algorithm is that it is parallelizable. In addition, as we discuss in Chapter 9, since our algorithm is framed in a variational inference framework, it can be extended with some effort to use other priors than the Dirichlet, such as the logistic normal priors and other models similar to adaptor grammars, such as fragment grammars. We demonstrated that our algorithm gets similar performance to an MCMC sampler on a word segmentation task.

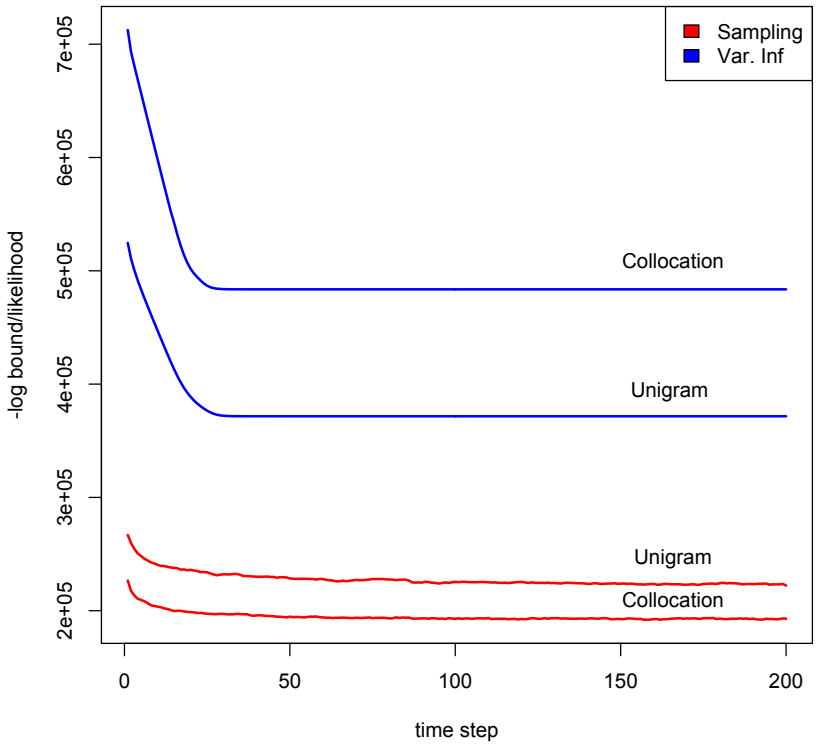


Figure 6.5: A plot of the free energy for variational inference (blue) and the negated log-likelihood for sampling (red) with the unigram and collocation grammars. Log-likelihood for sampling is measured every ten iterations.

Part II

Empirical Study

Chapter 7

Applications and Related Work

In the first part of this dissertation, we described a theoretical analysis of probabilistic grammar estimation as well as some novel estimation techniques for these grammars. In this part of the dissertation, we turn to testing the efficacy of our estimation techniques in a natural language application. This application, grammar induction, which can be thought of as "computational language acquisition", has a rich history in the computational linguistics literature, which we also cover in this chapter.

Grammar induction is an especially attractive application for estimation techniques of probabilistic grammars, given recent research that has identified the advantage of treating the problem of grammar induction in a modular way: constructing a grammar (usually a context-free grammar or one that is close to being context-free) and then estimating it using statistical techniques. Simply put, our estimation techniques can be readily used for computational language acquisition which uses probabilistic grammars that capture salient natural language phenomena, as demonstrated in Chapter 8.

Early attempts at solving this problem of grammar induction were rather algorithmic in nature. These attempts include development of systems and specialized algorithms for computational language acquisition. We discuss more of this history in Section 7.1.

Grammar induction is an application that has merit other than its attractiveness for testing the estimation of probabilistic grammars. In addition to the scientific merit which grammar induction provides to fields such as linguistics, cognitive science and formal language theory (which we discuss in Section 7.1), grammar induction can advance state-of-the-art for unsupervised natural language parsers. Such parsers are a key building block in many widespread NLP applications, espe-

cially those which perform deep analysis of text, yet, they are hard to construct for languages where full training data (i.e. examples of sentences together with their syntactic trees) is not available. This is where grammar induction can be used to compensate for the absence of annotated data in order to construct a natural language parser.

Our choice of application is in fact narrower than the general domain of grammar induction. In our experiments, we focus on *dependency* grammar induction. Dependency grammar (Tesnière, 1959) refers to linguistic theories that describe syntax using directed trees (in the graph theory sense). In these trees, words are vertices and edges that denote syntactic relationships. Such grammars can be context-free or context-sensitive in power, and they can be made probabilistic (Gaifman, 1965). Dependency grammars have been found to be especially useful for natural language applications, and thus they are widely used in NLP for information extraction (Yangarber et al., 2000), machine translation (Ding and Palmer, 2005), question answering (Wang et al., 2007), and other applications (Johansson and Nugues, 2007; Das et al., 2010). We choose to focus our experimental evaluation on dependency grammars because of their ability to capture syntactic phenomena in an intuitive and useful manner for all of the applications mentioned above.

Natural language applications that use dependency grammar induction require data in order to learn and estimate the grammar. The data we use in our empirical evaluation are treebanks from various languages. Here, a treebank refers to a collection of sentences, usually from a very specific domain, such as newswire text, where these sentences are annotated with grammatical derivations. We use treebank data so that we can eventually *evaluate* our parser on small amounts of annotated data that are used as a test set. Our stated objective is to build parsers in the absence of annotated data, but for evaluation purposes, we use annotated treebank test data, a common practice in the field of grammar induction.

We experiment with treebanks for various languages (Bulgarian, Chinese, Czech, Danish, Dutch, English, Greek, Japanese, Portuguese, Slovene, Spanish, Swedish, Turkish). See Appendix D for details about the treebanks that we use.

Evaluation in natural language processing, as mentioned before, is an essential step that indicates whether a model or algorithm successfully performs a given task. Typically for supervised and unsupervised models, decoded output is compared to expert human-annotated gold standard analyses, providing an objective measure of quality for the learned model. Quality is then measured on new test data that is unseen during training, in order to test the generalization of the learned model. This is an attractive approach for evaluation of the quality of induced

grammars.[1]

With the specification of our evaluation methodology (more about this in Section 8.2) and the data that we use, there is still a missing piece to the puzzle. We need to specify the grammar that we use with our estimation techniques. We choose the dependency model with valence (Klein and Manning, 2004) for this end. The DMV is a head automaton grammar that recursively generates parts of speech, corresponding to the lexical units in a sentence. It starts by generating a root tag, and then generates children to the left and to the right of the root. The DMV then visits each child, and repeats the part-of-speech generation procedure recursively. For a given predicate, the DMV makes a parametric distinction between those children generated first and the rest of the children.

We note that the dependency model with valence, as a context-free grammar, induces *projective* dependency trees. (A tree is considered projective if all of its arcs are projective; an arc is considered projective if every word between its two end points can be reached from one of the arc's endpoints.) Projectivity is a property that dominates most trees in the data that we use, but there is a small percentage of non-projective arcs in some of the treebanks. Our estimation techniques are not limited to the projective setting, and can be used with non-projective probabilistic grammars such as that described in Cohen et al. (2011b).

We turn now to describing work related to the application of dependency grammar induction. We break this related work into the following sections:

- Section 7.1.1 describes previous engineering efforts of grammar induction models.
- Section 7.1.2 describes the relationship of computational language acquisition and grammar induction to the problem of understanding human language acquisition.
- Section 7.1.3 describes the relationship between dependency grammar induction and the field of grammatical inference, that gives an alternative treatment to the problem of computational language acquisition.

[1] We note that an alternative to using unseen data in the unsupervised setting is to measure the quality of the data we estimate the model from. However, this gives a weaker indication of the generalization of the model.

7.1 Related Work

We describe in this section related work, as mentioned above.

7.1.1 Grammar Induction

Research on grammar induction dates back to the 1950s-1960s (Solomonoff, 1958; Horning, 1969), but the current statistical setting in which most grammar induction currently takes place started with the introduction of the inside-outside algorithm for PCFGs (Baker, 1979). The inside-outside algorithm enables statistical inference for context-free grammars and serves as a basis for many grammar induction systems.

Two main approaches to grammar induction have been developed since then. The first approach includes a structural search of a grammar based on distributional clustering of sequences of units which are present in the surface forms. The second approach starts with the assumption that there is a fixed grammar, and proceeds with estimation of the parameters for this grammar. While the first approach mostly focuses on inducing bracketing or identifying constituents from surface forms, the second approach can be used to induce other types of syntactic structures, such as dependency structures. The first approach, the structural search, is also related to the field of grammatical inference, as we explain in Section 7.1.3.

While the focus of this dissertation is on the second approach, a brief survey of earlier work that uses the first approach is relevant, because it has served as the foundation for modern grammar induction. Early work about the first approach by Stolcke and Omohundro (1994) proposed a minimum description length framework based on two grammar generalization operators, merging and chunking, both of which are applied in an iterative manner until convergence to a grammar. Another algorithm designed for grammar induction was introduced by van Zaanen (2000). Their method is called "alignment based learning". Their method is based on the idea that two fragments of a sentence can be identified as one constituent of the same type if they can be substituted in their respective contexts. This approach is closely related to the principle of substitutability (Harris, 1951). In fact, alignment based learning "reverses" this principle in order to identify constituent types. Methods similar to alignment based learning have also been proposed, for example, by Adriaans et al. (2000).

More recent work that combines both the first approach and the second approach uses incremental parsing (Seginer, 2007) with heuristics proposed for estimation of a lexicon. The structures that Seginer's parser induces are hybrids of

dependencies and constituents. Ponvert et al. (2011) also uses cascaded finite state models for grammar induction based on raw text, where the end goal is to identify chunks in a text rather than identifying a complete syntactic tree.

Going back to the problem of using probabilistic grammars for grammar induction, it is important to note that over the course of grammar induction research, a variety of probabilistic grammars have been tested for grammar induction (Lari and Young, 1990; Pereira and Schabes, 1992; Carroll and Charniak, 1992; de Marcken, 1995; Chen, 1995; Klein and Manning, 2002, 2004; Kurihara and Sato, 2006, *inter alia*) without much initial success. Early experiments, such as those performed by Carroll and Charniak, were partially successful because of the complexity of the grammar induction problem that led to challenges such as local maxima. More promising results were achieved in later experiments such as those introduced in Klein and Manning (2002). These experiments demonstrated that natural language syntax can be induced using the expectation-maximization algorithm if the model is chosen carefully. In Klein and Manning (2002), for example, the estimated model is a generative one that includes all subsequences of part-of-speech tags in the data, which are either flagged as "constituents" or "distituents." This model was called the constituent-context model (CCM). Klein and Manning's method is also related to a method proposed in Clark (2000), which is more of a structural search approach that is designed to induce clusters of sentence fragments based on their distributional context.

Klein and Manning recognized that the CCM model could be further improved to obtain even higher performance on the task of bracketed parsing. Thus, in Klein and Manning (2004), the constituent-context model was augmented by a model for dependencies between the units of surface forms in order to improve the performance of the CCM model. These dependencies were induced using the dependency model with valence (DMV), which we also use in our experiments, in Chapter 8. We explain our reasons for using the DMV in more detail below.

The DMV enabled a long thread of research about dependency grammar induction and has been widely recognized as an effective probabilistic grammar for this end. The DMV has been used to test estimation algorithms such as Viterbi EM (Spitkovsky et al., 2010b), contrastive estimation (Smith and Eisner, 2005), algorithms which gradually introduce more data to the learning process (Spitkovsky et al., 2010a) and other modifications to the EM algorithm (Spitkovsky et al., 2011a); it has been used to test the efficacy of multilingual learning through dependency grammar induction (Ganchev et al., 2009; Berg-Kirkpatrick and Klein, 2010); it has been used as a base model that has inspired more complex lexicalized models (Headden et al., 2009). The DMV has also been used with various esti-

mation techniques that implement it as a base model with the goal of improving the DMV's performance. This goal is achieved by relying on other properties of language and text such as: dependencies between parameters in the model (Berg-Kirkpatrick et al., 2010), sparsity (Gillenwater et al., 2010), preference for short attachments (Smith and Eisner, 2006), punctuation (Spitkovsky et al., 2011b), and additional annotation offered by web pages (Spitkovsky et al., 2010c). In addition, the DMV has also been used for inferring grammatical structures from non-parallel corpora (Cohen et al., 2011a). It has also been used with concave models which are used for its initialization Gimpel and Smith (2011). Finally, it has also been modified to include information about semantic relations (Naseem and Barzilay, 2011).

There is some recent work that does not make use of the DMV for dependency grammar induction. Examples of these studies include the use of tree substitution grammars in a nonparametric Bayesian model based on the Pitman-Yor process (Blunsom and Cohn, 2010), and the use of universal linguistic knowledge to induce syntax trees in partially unsupervised manner (Naseem et al., 2010).

The reasons we choose the DMV as our base grammar for dependency grammar induction are two-fold. On one hand, the DMV is a widely recognized grammar, and therefore, when testing our models, we can easily separate the problem of *constructing a grammar* from the problem of *estimating* it, with the latter being the focus of this dissertation. Our results, therefore, reflect state-of-the-art in the estimation techniques, comparable to previous and latter techniques for estimating the DMV. On the other hand, as we see later in Chapter 8, the DMV actually gives moderate accuracy in the supervised setting. This implies that properly estimating the DMV can actually lead to high accuracy, making it an attractive model for natural language.

With that note about supervised learning in mind, we also note that the DMV is related to the head-outward model used by Collins et al. (1999) and Collins (2003) for supervised parsing; Collins' parser is one of the best performing parsers for English (but naturally, far more complicated than the vanilla DMV model as it is designed for the supervised setting).

To wrap up this summary about grammar induction, we should also mention that unsupervised parsing of natural language has also been tackled using a framework called unsupervised data oriented parsing (U-DOP) (Bod, 2006a,b). Like earlier experiments with grammar induction, this line of research focuses on inducing bracketings that represent syntax. U-DOP works by assigning all possible binary trees to a set of sentences, and then choosing the most probable tree according to counts of subtrees in this assignment.

7.1.2 Language Learnability

In addition to the advantage of being able to construct an unsupervised parser using grammar induction, grammar induction also impacts other fields such as linguistics and cognitive science. More specifically, when we think of grammar induction as computational language acquisition, it has a direct relationship to the problem of understanding *human* language acquisition. The "stimuli" given to a computer for computational language acquisition resemble the stimuli that children receive during their language acquisition phase. Yet, current grammar induction research has difficulty giving insight about human language acquisition – a machine learning technique that manages to induce syntax from raw text is not necessarily credible as a model for language acquisition. Instead, we propose, like others have, that grammar induction can help us gain insight about a variant of the language acquisition problem. This variant tackles the problem of *learnability* of language: in other words, it may help to develop credible models that can show the learnability of formal languages which subsume natural languages.

Indeed, this type of question was explored by Gold (1967), who was the first to provide results about the learnability of language in a formal way. Gold's results were pessimistic: his main result was a negative one, showing that, according to a model he called "identification in the limit" (IIL), any *supra-finite* class of languages, including the set of context-free grammars, is not learnable. With IIL, the learner is presented with examples from the language, and at each step, she returns a hypothesis about the language which the examples are taken from. Her goal is to eventually identify the correct language, and consistently maintain that identification in subsequent steps in which she is presented more examples.

This negative result stands in opposition to, for example, our result of the polynomial complexity of learning language in the unsupervised setting from Chapter 4. This could perhaps be explained by the fact that there is a problematic flaw in the IIL model. The IIL model assumes that examples are presented in an adversarial manner, an unlikely property in the process of language acquisition (Clark and Lappin, 2011). In fact, it has been shown that adversarial environments for language learning lead to impairments in a child's language acquisition (Curtiss, 1977). It is difficult to extend Gold's model to the context of a non-adversarial model (Goldman and Mathias, 1996). Other flaws in Gold's approach as a model for language acquisition are noted in Clark and Lappin (2010) and Clark and Lappin (2011).

Another flaw with Gold's main negative result is that it relies strictly on learning from positive examples (in a non-probabilistic setting). The probabilistic set-

ting (which we use in this dissertation) has been argued to offer compensation for the absence of negative data (Angluin, 1988). The frequency in which we observe patterns in language data can be used to create a gradient of likelihood for the language patterns, ranging from highly probable to improbable. Indeed, since the emergence of Gold's results, many learnability results have been achieved which show that large classes of grammars can be learned in the probabilistic setting (Angluin, 1988; Chater and Vitányi, 2007). In this context, our results from Chapter 4 complement and reinforce this idea: even though the sample complexity bounds in the unsupervised setting are larger than their counterpart in the supervised setting, they are still polynomial, and therefore the class of probabilistic grammars are considered learnable in the setting we describe.

7.1.3 Grammatical Inference

Learning natural language using a grammar is clearly not limited to just an estimation problem. One can also choose to infer the structure of the grammar itself, rather than assuming that it is known to the learner who focuses on its estimation. The problem of inferring the grammar itself is the focus of the grammatical inference field. The goal of this field is to develop algorithms and theory for inferring a grammar from examples, which can include both positive as well as negative examples.

The field of grammatical inference, in many cases, studies artificial data, i.e., strings that do not originate from human generated text. For example, with the Omphalos competition in 2004 (Starkie et al., 2004), the participants were requested to identify a context-free grammar from a synthetic set of positive instances. The field of grammatical inference is also, usually, more algorithmically-driven. Researchers design algorithms for various classes of languages (Angluin, 1988; Clark et al., 2010; Yoshinaka and Clark, 2010; Clark, 2010a, *interalia*), and prove the correctness of these algorithms. These developments can also shed some light on issues regarding the learnability of language (see Section 7.1.2). In natural language processing, however, general statistical learning algorithms are usually used to perform the learning, and principles, such as maximum likelihood estimation or large margin, guide researchers. Clark (2010b) discusses a more general setting that can be used in devising algorithms for grammatical inference.

In general, it is not trivial to completely tease apart the field of grammatical inference from the field of statistically-driven grammar induction which we focus on in this dissertation. For example, earlier work of structural search methods for the goal of grammar induction, as mentioned in Section 7.1.1, can be partially

thought of as relating more to grammatical inference than to current statistical grammar induction. Indeed, perhaps some of the main distinctions between these two communities include the types of data each works with, the class of algorithms they employ and the motivations behind the work (i.e. explaining language acquisition versus creating natural language parsers).

We conclude with a final note about the connection between grammatical inference and estimation of grammars. Perhaps there is a way to connect between the estimation of a grammar and learning its structure. For example, with context-free grammars, we can start with a large set of rules for the grammar, which includes practically all possible rules that use a set of nonterminals (or the set of symbols that represent rules) in the grammar. From that point, the goal of the learner would not just be to estimate parameters of the grammar, but rather to enforce *sparsity* as well: i.e. set the probabilities of many rules in the grammar to 0. This line of research is beyond the scope of this thesis, but perhaps some of the estimation methodology we develop can be extended to enforce sparsity on grammar rules. See Gillenwater et al. (2010) for a discussion of the role of sparsity in grammar induction. Other discussion of sparsity with parsing includes Mohri and Roark (2006). Discussion of sparsity in the Bayesian setting appears in Johnson (2007).

7.2 Summary

In this chapter we have detailed the main empirical setting with which we are going to experiment in Chapter 8, and surveyed related work. We note that our estimation techniques, although mostly applied to dependency grammar induction, are not limited to this setting. Indeed, our estimation techniques are appropriate for the estimation of any type of probabilistic grammars.

Chapter 8

Multilingual Grammar Induction

In this chapter, we provide results for an empirical evaluation of the estimation algorithms that we described in Chapters 5 and 6 to perform the task of dependency grammar induction (Chapter 7). Our focus is on controlled experiments that compare Bayesian and non-Bayesian baselines with the logistic normal prior estimation technique and the adaptor grammar estimation technique. We also describe a setting in which the estimation procedure along with the shared logistic normal distribution can be used for multilingual learning from non-parallel corpora, i.e. to learn the syntax of two (or more) languages at the same time by tying the parameters of the grammars of each language.

The rest of this chapter is organized as follows. We begin with a detailed explanation of the DMV model (Section 8.1). We then turn to describe the experimental setting and report the experiment results in Sections 8.3–8.6.

Following this, we analyze some of the results in Section 8.7 and give a summary in Section 8.8.

8.1 Dependency Model with Valence

Our experiments perform unsupervised induction of probabilistic dependency grammars using a model known as "dependency model with valence" (Klein and Manning, 2004). The model is a probabilistic split head automaton grammar (Alshawi and Buchsbaum, 1996) which renders inference cubic in the length of the sentence (Eisner, 1997). The language of the grammar is context-free, though our models are permissive and allow the derivation of any string in Γ^*. This is a major point of departure between theoretical work on grammatical inference and work on nat-

ural language text, particularly in the use of probabilistic grammars. Our goal is to induce a distribution over derivations so that the most likely derivations under the model most closely mimic those preferred by linguists.

"Valence" in "dependency model with *valence*", refers to the number of arguments controlled by the head of a phrase.[1] In the DMV, each word has a binomial distribution over the possibility that it has at least one child on the left (as well as on the right), and a geometric distribution over the number of further children (for each side).

Let $\boldsymbol{x} = \langle x_1, x_2, ..., x_n \rangle$ be a sentence (here, as in prior work, represented as a sequence of part-of-speech tags). x_0 is a special "wall" symbol, \$, on the left of every sentence. A tree $\mathbf{y}$ is defined by a pair of functions $\mathbf{y}_{\mathit{left}}$ and $\mathbf{y}_{\mathit{right}}$ (both $\{0, 1, 2, ..., n\} \rightarrow 2^{\{1,2,...,n\}}$) that map each word to its sets of left and right dependents, respectively. Here, the graph is constrained as a *projective* tree rooted at $x_0 = \$$: each word except \$ has a single parent, and there are no cycles or crossing dependencies. $\mathbf{y}_{\mathit{left}}(0)$ is taken to be empty, and $\mathbf{y}_{\mathit{right}}(0)$ contains the sentence's single head. Let $\mathbf{y}^{(i)}$ denote the subtree rooted at position i (i.e., $\mathbf{y}^{(i)}$ is a tree consisting of all descendents of x_i in the tree $\mathbf{y}$). The probability $P(\mathbf{y}^{(i)} \mid x_i, \boldsymbol{\theta})$ of generating this subtree, given its head word x_i, is defined recursively:

$$p(\mathbf{y}^{(i)} \mid x_i, \boldsymbol{\theta}) = \prod_{D \in \{\mathit{left}, \mathit{right}\}} \theta_{\mathrm{s}}(\mathrm{stop} \mid x_i, D, [\mathbf{y}_D(i) = \emptyset]) \tag{8.1}$$
$$\times \prod_{j \in \mathbf{y}_D(i)} \theta_{\mathrm{s}}(\neg\mathrm{stop} \mid x_i, D, \mathrm{first}_{\mathbf{y}}(j)) \times \theta_{\mathrm{c}}(x_j \mid x_i, D) \times p(\mathbf{y}^{(j)} \mid x_j, \boldsymbol{\theta}),$$

where $\mathrm{first}_{\mathbf{y}}(j)$ is a predicate defined as true iff x_j is the closest child (on either side) to its parent x_i. The probability of the entire tree is given by $p(\boldsymbol{x}, \mathbf{y} \mid \boldsymbol{\theta}) = p(\mathbf{y}^{(0)} \mid \$, \boldsymbol{\theta})$. The parameters $\boldsymbol{\theta}$ are the conditional multinomial distributions $\theta_{\mathrm{s}}(\cdot \mid \cdot, \cdot, \cdot)$ and $\theta_{\mathrm{c}}(\cdot \mid \cdot, \cdot)$. To follow the general setting of Equation 2.1, we index these distributions as $\boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_K$. Figure 8.1 shows a dependency tree and its probability under this model (Equation 8.1).

Note that if all weights $\boldsymbol{\theta}$ are greater than zero, the model permits *any* dependency tree over *any* sentence in Γ^*. Hence the goal of grammar induction is to model the *distribution* of derivations, not to separate grammatical strings or derivations from ungrammatical ones.

[1] Here, we refer to "head of a phrase" in the linguistic sense—the word in a phrase that determines the syntactic category of a phrase.

Klein and Manning's (2004) dependency model with valence is widely recognized as an effective probabilistic grammar for dependency grammar induction. We refer the reader to Chapter 7 for a literature review of the use of the DMV model in various settings.

8.2 Evaluation

Before we detail our evaluation measure, a note about the decoding process is in order. As mentioned in Sections 5.4, we experiment with several decoding methods, including Viterbi decoding, minimum Bayes decoding and committee decoding (using the logistic normal prior). For minimum Bayes decoding, we need to specify the loss function that we use, cost. With our experiments, cost is a function that counts the number of words attached to a parent different than the one in the correct analysis. This means that our MBR decoder attempts to minimize the expected number of children attached to the wrong parents. MBR decoding in this case works as follows: using a set of parameters for a grammar and using the inside-outside algorithm, we compute the posterior probability of each dependency attachment (directed edge in the graph) that is present in the grammatical derivation for the sentence. Next, we find the tree with the largest score, with the score being the sum of the posterior probabilities of each edge present in the tree.

In our experiments, we use the "attachment accuracy" evaluation measure which is strictly related to MBR decoding. With the attachment accuracy measure, we calculate the fraction of parents that were correctly identified from gold standard data. This relationship between MBR decoding and attachment accuracy bears noteworthy implications on performance. As we see in the next chapter, MBR decoding indeed tends to function better than Viterbi decoding.

Attachment accuracy has been standardized as the main accuracy measure that the natural language processing community uses for evaluating dependency parsers, both in the supervised and unsupervised context. Yet, attachment accuracy has been criticized as being too fragile in the face of different annotation schemas and annotations which are linguistically controversial. We refer the reader to Schwartz et al. (2011) for a discussion of this issue. We leave it for future work to address these issues with the standard evaluation.

8.3 Experimental Setting

We turn now to describing the experimental setting in which we test the estimation methods of adaptor grammars and logistic normal priors. We note that the use of adaptor grammars for dependency grammar induction is novel in itself. Until now, adaptor grammars have been used mostly for segmentation (Johnson and Goldwater, 2009), entity recognition (Elsner et al., 2009) and modeling perspective Hardisty et al. (2010). More recently, a nonparametric Bayesian model has been devised for dependency grammar induction Blunsom and Cohn (2010), but it is not based on adaptor grammars, but instead of a model tailored specifically for dependency grammar induction.

We also consider an interesting setting for the shared logistic normal priors, which emerges naturally for multilingual learning. More specifically, we describe how one can use SLN to tie parameters across several grammars for different languages.

Our analysis starts with an extensive testing of adaptor grammars and logistic normal priors for the Penn treebank for English. Later in this chapter, we extend our experiments to more languages in other treebanks. A full listing of the treebanks that we use in this chapter is included in Appendix D.

The experimental report is organized as follows:

- (Section 8.4) Experiments with dependency grammar induction for English text using the logistic normal distribution and adaptor grammars.

- (Section 8.4.1) Experiments with text in additional languages using the logistic normal distribution and adaptor grammars.

- (Section 8.5) Experiments with the shared logistic normal distribution for tying parameters which correspond to the same coarse part-of-speech tag (English, Portuguese, and Turkish).

- (Section 8.6) Experiments with the shared logistic normal distribution in *bilingual* settings (English, Portuguese, and Turkish).

- (Section 8.7) Error analysis for the dependency structures induced by the logistic normal distribution and comparison between the models learned by adaptor grammars and the logistic normal distribution.

8.4 English Text

We begin our experiments with the *Wall Street Journal* Penn treebank (Marcus et al., 1993). Following the standard practice, sentences were stripped of words and punctuation, leaving just part-of-speech tags for the unsupervised induction of the dependency structure. We note that, in this setting, it is common to use gold standard part-of-speech tags as the input for the learning algorithm (Klein and Manning, 2004; Smith and Eisner, 2006; Spitkovsky et al., 2010b,a; Gillenwater et al., 2010, *inter alia*). We follow this practice as well.

We train our models on §2–21, tune them on §22 (without using annotations), and report the final results on §23. Unsupervised training for these data sets can be costly, as it requires iteratively running a cubic-time inside-outside dynamic programming algorithm, so we follow Klein and Manning (2004) in restricting the training set to sentences with a length of ten or fewer words. We also follow this constraint because short sentences are less structurally ambiguous and may therefore be easier to learn from.

To evaluate the performance of our models, we report the fraction of words whose predicted parent matches the gold standard annotation in the treebank.[2] This performance measure is known as *attachment accuracy*. We report attachment accuracy on three subsets of the test corpus: sentences of length ≤ 10 (typically reported in prior work and most similar to the training data set), length ≤ 20, and the full test corpus.

Logistic Normal Priors Setting With the logistic normal priors, we consider the three decoding methods mentioned in Section 5.4.1. For MBR decoding, we use the number of dependency attachment errors as the cost function. This means that at the point of decoding, we minimize the expected number of attachment errors according to the prediction of the estimated model. Because committee decoding is a randomized algorithm, we run it ten times on unseen data, and then average the dependency attachment accuracy.

Initialization is important for all conditions, because the likelihood, as well as our variational bound functions, are non-concave. For the multinomial values ($\boldsymbol{\theta}$), we use the harmonic initializer from Klein and Manning (2004). This method estimates $\boldsymbol{\theta}$ using soft counts on the training data where, in an n-length sentence,

[2]The Penn Treebank's phrase-structure annotations were converted to dependencies using the head rules of Yamada and Matsumoto, which are very similar to those proposed by Collins (2003). See `http://www.jaist.ac.jp/~h-yamada`.

(i) each word is counted as the sentence's head $\frac{1}{n}$ times, and (ii) each word x_i attaches to x_j proportional to $|i-j|^{-1}$, normalized to a single attachment per word. This initializer is used with MLE and Dirichlet-I (where "I" stands for model I from Figure 5.1). In the case of LN-I and LN-II, this initializer is used both to estimate $\boldsymbol{\mu}$ and to estimate variational parameters inside the E-step.

For learning with the logistic normal prior, we consider two initializations of the covariance matrices $\boldsymbol{\Sigma}_k$. The first is the $N_k \times N_k$ identity matrix. We then tried to bias the solution by injecting prior knowledge about the part-of-speech tags. To do this, we manually mapped the tag set (34 tags) to twelve disjoint tag "families." These are simply coarser tags: adjective, adverb, conjunction, foreign, interjection, noun, number, particle, preposition, pronoun, proper, verb. These coarse tags were chosen to loosely account for the part-of-speech tag sets of seven treebanks in different languages. The mapping from fine-grained tags to coarse tags is based on the annotation guidelines of the relevant treebank. This mapping into families provides the basis for an initialization of the covariance matrices for the dependency distributions: 1 on the diagonal, 0.5 between probabilities of possible child tags that belong to the same family, and 0 elsewhere. These results are denoted as "families" and are compared to the identity matrix as an initializer.

Adaptor Grammar Setting In order to use adaptor grammars with the DMV, we first have to reduce the DMV to a PCFG. We follow the reduction presented in Smith (2006). Figure 8.2 presents this reduction in detail. With adaptor grammars, it is important to determine which nonterminals are adapted in the grammar. Ideally, we could try to have all nonterminals adapted in the DMV (represented as a context-free grammar), and let the learning algorithm learn which strings are important to cache, and for which strings we should use regular PCFG expansion. However, such an adaptor grammar model is extremely large, especially when using our variational inference algorithm, thus it would not fit into computer memory. For this reason, we carefully select the nonterminals that we choose to adapt.

More specifically, we choose to adapt nonterminals for the part-of-speech tag categories which are most prominent in any language: nouns and verbs. We therefore have two experimental settings with adaptor grammars, one that adapts noun POS tags and the other that adapts verb POS tags. Because of memory limitations, we cannot adapt nouns and verbs together.

While the reason for adapting only a subset of nonterminals can be merely of practical convenience, there are other reasons to adapt specifically the set of

nouns and verbs. Most part-of-speech tag in a given sentence can be thought of as revolving around the main parts of the sentence, the noun phrases and the verbial phrases. This phrases are indeed dominated by the noun nonterminals and verb nonterminals. It is important to note that when adapting these nonterminals according to the reduction in Figure 8.2, we are in essence "caching" subtrees which consist of whole left constituents, whole right constituents, or whole constituents altogether. For example, the non-terminal $N[\mathrm{NN}]$, when adapted, leads to caching whole constituents dominated by the NN part-of-speech tag.

Because of the recursive structure of the PCFG in Figure 8.2, adapting the nonterminals corresponding to a certain part-of-speech tag implies that we cache mostly whole constituents headed by this part-of-speech tag (or partial constituents headed by this part-of-speech). In the case of the nonterminals R_{c_0}, R_c, L_{c_0} and L_c, the cached strings also include dependents of some other part-of-speech tags as well, because of the binary structure of their hand-side. Note that this means we are also caching strings which do not necessarily represent a full noun or verb phrase. We discuss this further in Section 8.4.1.

To decide which POS tags denote nouns and which denote verbs, we use the same mapping for POS tags that we use with the logistic normal priors (mentioned above). We use the preprocessing step defined in §6.2.4 along with a uniform grammar and take the top 15,000 (or 8,000, if memory restrictions force us to do so) strings[3] for each nonterminal of a noun or verb constituent. For decoding, we use the MBR and Viterbi decoding mechanisms described in Chapter 6.

Baselines We compared several models where learning is accomplished using (variational) EM: MLE, standard maximum-likelihood estimation using EM; Dirichlet-I, a common baseline in the literature which uses a Dirichlet prior together with variational EM; LN-I (LN-II), a model with the logistic normal distribution using model I (model II); AG-Noun (AG-Verb), adaptor grammars with adapted noun nonterminals (adapted verb nonterminals). In all cases, we either run the (variational) EM algorithm until convergence of the log-likelihood (or its bound) or until the log-likelihood on an unannotated development set of sentences ceases to increase.

We note that in the full test set, attaching each word to the word on its right ("Attach-Right") achieves about 30% accuracy, and attaching each word to the word on its left ("Attach-Left") achieves about 20% accuracy.

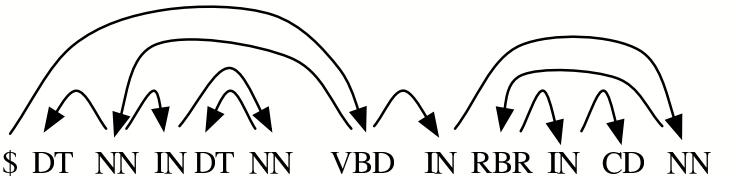
[3]When adating noun nonterminals with the Japanese treebank, we take only the top 150 strings, because of the large number of noun part-of-speech tags in this treebank.

Report Table 8.1 shows the experimental results. Note that there are two logistic normal variants which consistently achieve lower performance than their counterparts: use of model II (versus model I) and use of committee decoding rather than Viterbi or MBR decoding. This suggests that the covariance matrices play a useful role during the learning process, but that they are not informative when performing decoding, since they are not used in Viterbi and MBR decoding. Interestingly, Smith and Eisner (2006) report a similar result for *structurally biased* DMV—a model that includes a parameter which controls the length of the decoded dependencies. Their bias parameter is useful only during the learning process, never during decoding. In general, the logistic normal distribution with model I substantially outperforms the baselines. It is interesting to note that LN-I outperforms Dirichlet-I and MLE even when identity covariance matrices are used for initialization. As a matter of fact, even when permitting diagonal covariance matrices in our model, there is a significant improvement in performance compared to Dirichlet. This is because such covariance matrices permit modeling of variance in the parameters, while the Dirichlet prior does not permit that.

The next thing to notice is that the logistic normal priors perform substantially better than adaptor grammars do for English. This finding is related strictly to the data that we use with English. As we see later, adaptor grammars actually function better, on average, than the logistic normal prior, when considering more languages. It is also interesting to note that adaptor grammars function better when adapting verbs (in comparison to adapting nouns). This is consistent with our findings for other languages, where adapting verbs always performs better than adapting nouns.

It is also interesting to note that adaptor grammars behave similarly with MBR decoding and Viterbi decoding. With the word segmentation experiments in Chapter 6, the difference between MBR and Viterbi decoding was much more substantial.

In preparation for our next set of experiments, we note that when we tested model II with the logistic normal prior and with committee decoding on other languages, the decrease in performance was consistent. For the rest of the experiments, we report only MBR (and possibly Viterbi) decoding results using model I. The reason for the underperformance of model II could be a result of the small number of parameters that are defined by the model. This small set of parameters cannot fully capture the nuances across sentences in the data.



$$\mathbf{y} =$$

$$\boldsymbol{x} = \langle \$ \text{ DT NN IN DT NN VBD IN RBR IN CD NN} \rangle$$

$$\begin{aligned}
p(\boldsymbol{x}, \mathbf{y} \mid \boldsymbol{\theta}) &= \theta_{\text{c}}(\text{VBD} \mid \$, \mathsf{r}) \times p(\mathbf{y}^{(1)} \mid \text{VBD}, \boldsymbol{\theta}) \\
p(\mathbf{y}^{(1)} \mid \text{VBD}, \boldsymbol{\theta}) &= \theta_{\text{s}}(\neg\text{stop} \mid \text{VBD}, \mathsf{l}, \mathsf{f}) \times \theta_{\text{c}}(\text{NN} \mid \text{VBD}, \mathsf{l}) \times p(\mathbf{y}^{(2)} \mid \text{NN}, \boldsymbol{\theta}) \\
&\quad \times \theta_{\text{s}}(\text{stop} \mid \text{VBD}, \mathsf{l}, \mathsf{t}) \times \theta_{\text{s}}(\neg\text{stop} \mid \text{VBD}, \mathsf{r}, \mathsf{f}) \times \theta_{\text{c}}(\text{IN} \mid \text{VBD}, \mathsf{r}) \\
&\quad \times p(\mathbf{y}^{(4)} \mid \text{IN}, \boldsymbol{\theta}) \times \theta_{\text{s}}(\text{stop} \mid \text{VBD}, \mathsf{r}, \mathsf{t}) \\
p(\mathbf{y}^{(2)} \mid \text{NN}, \boldsymbol{\theta}) &= \theta_{\text{s}}(\neg\text{stop} \mid \text{NN}, \mathsf{l}, \mathsf{f}) \times \theta_{\text{c}}(\text{DT} \mid \text{NN}, \mathsf{l}) \times \theta_{\text{s}}(\text{stop} \mid \text{DT}, \mathsf{r}, \mathsf{f}) \\
&\quad \times \theta_{\text{s}}(\text{stop} \mid \text{DT}, \mathsf{l}, \mathsf{f}) \theta_{\text{c}}(\text{IN} \mid \text{NN}, \mathsf{r}) \times p(\mathbf{y}^{(3)} \mid \text{IN}, \boldsymbol{\theta}) \\
&\quad \times \theta_{\text{s}}(\text{stop} \mid \text{IN}, \mathsf{l}, \mathsf{f}) \times \theta_{\text{s}}(\text{stop} \mid \text{NN}, \mathsf{l}, \mathsf{t}) \times \theta_{\text{s}}(\text{stop} \mid \text{NN}, \mathsf{r}, \mathsf{t}) \\
p(\mathbf{y}^{(3)} \mid \text{IN}, \boldsymbol{\theta}) &= \theta_{\text{s}}(\neg\text{stop} \mid \text{IN}, \mathsf{r}, \mathsf{f}) \times \theta_{\text{c}}(\text{NN} \mid \text{IN}, \mathsf{r}) \times \theta_{\text{c}}(\text{DT} \mid \text{NN}, \mathsf{l}) \\
&\quad \times \theta_{\text{s}}(\text{stop} \mid \text{DT}, \mathsf{r}, \mathsf{f}) \times \theta_{\text{s}}(\text{stop} \mid \text{DT}, \mathsf{l}, \mathsf{f}) \\
&\quad \times \theta_{\text{s}}(\text{stop} \mid \text{NN}, \mathsf{r}, \mathsf{f}) \times \theta_{\text{s}}(\text{stop} \mid \text{NN}, \mathsf{l}, \mathsf{t}) \\
p(\mathbf{y}^{(4)} \mid \text{IN}, \boldsymbol{\theta}) &= \theta_{\text{s}}(\text{stop} \mid \text{IN}, \mathsf{l}, \mathsf{f}) \times \theta_{\text{s}}(\neg\text{stop} \mid \text{IN}, \mathsf{r}, \mathsf{f}) \times \theta_{\text{c}}(\text{NN} \mid \text{IN}, \mathsf{r}) \\
&\quad \times \theta_{\text{s}}(\text{stop} \mid \text{NN}, \mathsf{r}, \mathsf{f}) \times \theta_{\text{s}}(\neg\text{stop} \mid \text{NN}, \mathsf{l}, \mathsf{f}) \times \theta_{\text{c}}(\text{RBR} \mid \text{NN}, \mathsf{r}) \\
&\quad \times \theta_{\text{s}}(\text{stop} \mid \text{RBR}, \mathsf{l}, \mathsf{f}) \times p(\mathbf{y}^{(5)} \mid \text{RBR}, \boldsymbol{\theta}) \\
p(\mathbf{y}^{(5)} \mid \text{RBR}, \boldsymbol{\theta}) &= \theta_{\text{s}}(\neg\text{stop} \mid \text{RBR}, \mathsf{r}, \mathsf{f}) \times \theta_{\text{c}}(\text{IN} \mid \text{RBR}, \mathsf{r}) \times \theta_{\text{c}}(\text{CD} \mid \text{IN}, \mathsf{r}) \\
&\quad \times \theta_{\text{s}}(\text{stop} \mid \text{IN}, \mathsf{l}, \mathsf{f}) \times \theta_{\text{s}}(\text{stop} \mid \text{IN}, \mathsf{r}, \mathsf{t}) \times \theta_{\text{s}}(\text{stop} \mid \text{CD}, \mathsf{r}, \mathsf{f}) \\
&\quad \times \theta_{\text{s}}(\text{stop} \mid \text{CD}, \mathsf{l}, \mathsf{f})
\end{aligned}$$

Figure 8.1: An example of a dependency tree (derivation $\mathbf{y}$). and its probability. The part-of-speech tags NN, VBD, DT, CD, RBR, and IN denote noun, past-tense verb, determiner, number, comparative adverb, and preposition, respectively, following Penn Treebank conventions. We break the probability of the tree down into recursive parts, one per head word, marked in blue (lighter). l, r, t, and f denote left, right, true, and false, respectively (see Equation 8.1).

DMV weight	context-free rule
$\theta_{\mathrm{c}}(t \mid \$, \mathsf{r})$	$S \rightarrow N[t]$
$\theta_{\mathrm{s}}(\mathrm{stop} \mid t, \mathsf{r}, \mathsf{f})$	$N[x] \rightarrow L_0[x]$
$\theta_{\mathrm{s}}(\neg\mathrm{stop} \mid t, \mathsf{r}, \mathsf{f})$	$N[x] \rightarrow R_{c_0}[x]$
$\theta_{\mathrm{c}}(t \mid \mathsf{r}, t')$	$R_{c_0}[t] \rightarrow R[t]N[t']$
$\theta_{\mathrm{s}}(\mathrm{stop} \mid t, \mathsf{r}, \mathsf{t})$	$R[t] \rightarrow L_0[t]$
$\theta_{\mathrm{s}}(\neg\mathrm{stop} \mid t, \mathsf{r}, \mathsf{t})$	$R[t] \rightarrow R_c[t]$
$\theta_{\mathrm{c}}(t \mid \mathsf{r}, t')$	$R_c[t] \rightarrow R[t]N[t']$
$\theta_{\mathrm{s}}(\mathrm{stop} \mid \mathsf{l}, \mathsf{f})$	$L_0[t] \rightarrow t$
$\theta_{\mathrm{s}}(\neg\mathrm{stop} \mid \mathsf{l}, \mathsf{f})$	$L_0[t] \rightarrow L_{c_0}[t]$
$\theta_{\mathrm{c}}(t \mid \mathsf{l}, t')$	$L_{c_0}[t] \rightarrow N[t']L[t]$
$\theta_{\mathrm{s}}(\mathrm{stop} \mid \mathsf{l}, \mathsf{t})$	$L[t] \rightarrow t$
$\theta_{\mathrm{s}}(\neg\mathrm{stop} \mid \mathsf{l}, \mathsf{t})$	$L[t] \rightarrow L_c[t]$
$\theta_{\mathrm{c}}(t \mid \mathsf{l}, t')$	$L_c[t] \rightarrow N[t']L[t]$

Figure 8.2: A representation of the DMV using a probabilistic context-free grammar. All nonterminals (except for the starting symbol S) are decorated with part-of-speech tags, and instantiated for each part-of-speech tag. See Section 8.1 for a description of the weights.

	attachment accuracy (%)								
	Viterbi decoding			MBR decoding			Committee decoding		
	≤ 10	≤ 20	all	≤ 10	≤ 20	all	≤ 10	≤ 20	all
MLE	45.8	39.1	34.2	46.1	39.9	35.9		$*$	
Dirichlet-I	45.9	39.4	34.9	46.1	40.6	36.9		$*$	
LN-I, $\boldsymbol{\Sigma}_k^{(0)} = \mathbf{I}$	56.5	42.9	36.6	58.4	45.2	39.5	**56.4**$_{\pm.001}$	**42.3**$_{\pm.001}$	**36.2**$_{\pm.001}$
LN-I, families	**59.3**	**45.1**	**39.0**	**59.4**	**45.9**	**40.5**	56.3$_{\pm.01}$	41.3$_{\pm.01}$	34.9$_{\pm.005}$
LN-II, $\boldsymbol{\Sigma}_k^{(0)} = \mathbf{I}$	26.1	24.0	22.8	27.9	26.1	25.3	22.0$_{\pm.02}$	20.1$_{\pm.02}$	19.1$_{\pm.02}$
LN-II, families	24.9	21.0	19.2	26.3	22.8	21.5	26.6$_{\pm.003}$	22.7$_{\pm.003}$	20.8$_{\pm.0006}$
AG-Noun	26.6	23.5	22.3	27.7	24.9	24.2		$*$	
AG-Verb	45.5	37.3	31.6	45.9	38.0	33.0		$*$	

Table 8.1: Attachment accuracy of different learning methods on unseen test data from the Penn Treebank at varying levels of difficulty imposed through a length filter. MLE is a reproduction of an earlier result using EM (Klein and Manning, 2004). LN-I and LN-II denote use of the logistic normal with model I and model II (Figure 5.1), respectively. Committee decoding includes ten averaged runs. The numbers in small font denote variance. Results in bold denote the best results in a column. Training is done on sentences of length ≤ 10, though testing is done on longer sentences as well.

8.4.1 Additional Languages

We turn now to a description of experiments, parallel to those we performed with English, for other languages. The languages that we experiment with are Bulgarian, Chinese, Czech, Danish, Dutch, English, Greek, Japanese, Portuguese, Slovene, Spanish, Swedish and Turkish. Appendix D details treebank sizes, tagset sizes and other information about the treebanks.

As in the case for English, sentences were stripped of words and punctuation, leaving just part-of-speech tags for the unsupervised induction of the dependency structure. All learning algorithms were run on sentences with a length of ten words or less.

Table 8.2 gives results of using the logistic normal priors and adaptor grammars compared to the baselines mentioned in Section 8.4. Just like with English, we note that the results for adaptor grammars are not very different for Viterbi and MBR decoding, unlike the case with word segmentation (Section 6.3). In several cases, adaptor grammars significantly improve performance over classic EM and variational EM that use a Dirichlet prior. It seems that this mainly happens when the *verb* nonterminals are adapted and not when the noun nonterminals are adapted. This is consistent with the behavior mentioned in Section 8.4. The consistent improvements of performance when adapting verb nonterminals and not noun nonterminals is surprising at first. One may expect that adapting noun nonterminals would work better, because noun pharses, in many cases, act as non-compositional phrases (such as multi-word expressions) and therefore we would want to cache them (see Johnson (2010) for explanation). Verb phrases, on the other hand, are usually compositional phrases, so we would want to use the PCFG rules to expand them. There could be several explanations to explain this result. First, note that our analysis is done on sentences which consist only *parts of speech* and not words. Therefore, in many cases, verb pharse patterns occur frequently enough that caching them as non-compositional structure can help. Another point to note (as mentioned in Section 8.4) is that because of the way the DMV PCFG is structured, we are caching strings that contain partial constituents from other part-of-speech tags as well (because a nonterminal headed by a verb part-of-speech tag can dominate other partial constituents as well). This means that verb nonterminals dominate longer substrings (than noun nonterminals) that occur frequently in the corpus, because verb nonterminals tend to occur higher in dependency trees. The conclusion that we arrive to is that caching longer substrings for the task of *part-of-speech* dependency grammar induction is actually desirable. These strings, because of the unlexicalized setting, behave somewhat

like non-compositional pharses.

It also seems that adaptor grammars behave better for treebanks for which not much data is available, while the logistic normal prior tends to work better when there are relatively large amounts of data available. It is not completely clear why this is the case. With the logistic normal distribution we require estimation of a large number of parameters, and this could hinder performance in the case of small amounts of data. However, a similar argument can be made about adaptor grammars – where we require estimation of the parameters of all grammatons.

More detailed results for the logistic normal are given in Figure 8.3, for Portuguese, Japanese, Czech, Chinese and Turkish. Note that for Portuguese, the difference is much smaller between the EM baselines and logistic normal variational EM when only short sentences are considered, but there is a wider gap for longer sentences; the LN models appear to generalize better to longer sentences. For Turkish, no method outperforms Attach-Right, but there is still a big gap between variational EM using the logistic normal and other EM baselines. The case is similar for Japanese, though the logistic normal does outperform the Attach-Right baselines for shorter sentences in this case. For Czech, it seems like Dirichlet and EM do somewhat better than the logistic normal prior, but performance of all four methods is close. It is conceivable that the approximation inherent in a projective syntax representation for the Czech sentences (whose gold-standard analyses have a relatively large fraction of nonprojective dependencies) interacts with different models in different ways.[4]

In general, the covariance matrices learned when initializing with the identity covariance matrix are rather sparse, but there is a high degree of variability across the diagonal (for the variance values learned). For the DMV, when using an identity initializer, diagonal matrices represent the local optimum that is reached by the variational EM algorithm. When initializing the covariance matrices with the tag family initializer, the learned matrices are still rather sparse, but they have a larger number of significant correlations (for Portuguese, for example, using a t-test to test the significance of the correlation, we found that 0.3% of the values in the covariance matrices had significant correlation).[5]

[4]We note that we also experimented with other languages, including Hebrew and Arabic. We do not include these results, because in these cases all methods, including MLE, Dirichlet-I and LN-I performed poorly (though Dirichlet-I and MLE sometimes does better than LN-I). We believe that for these languages, the DMV is probably not the appropriate model. Developing better grammatical models for these languages is beyond the scope of this paper.

[5]However, it is interesting to note that most of the elements of the covariance matrices were not exactly zero. For example, 90% of the values in the covariance matrices were larger (in absolute

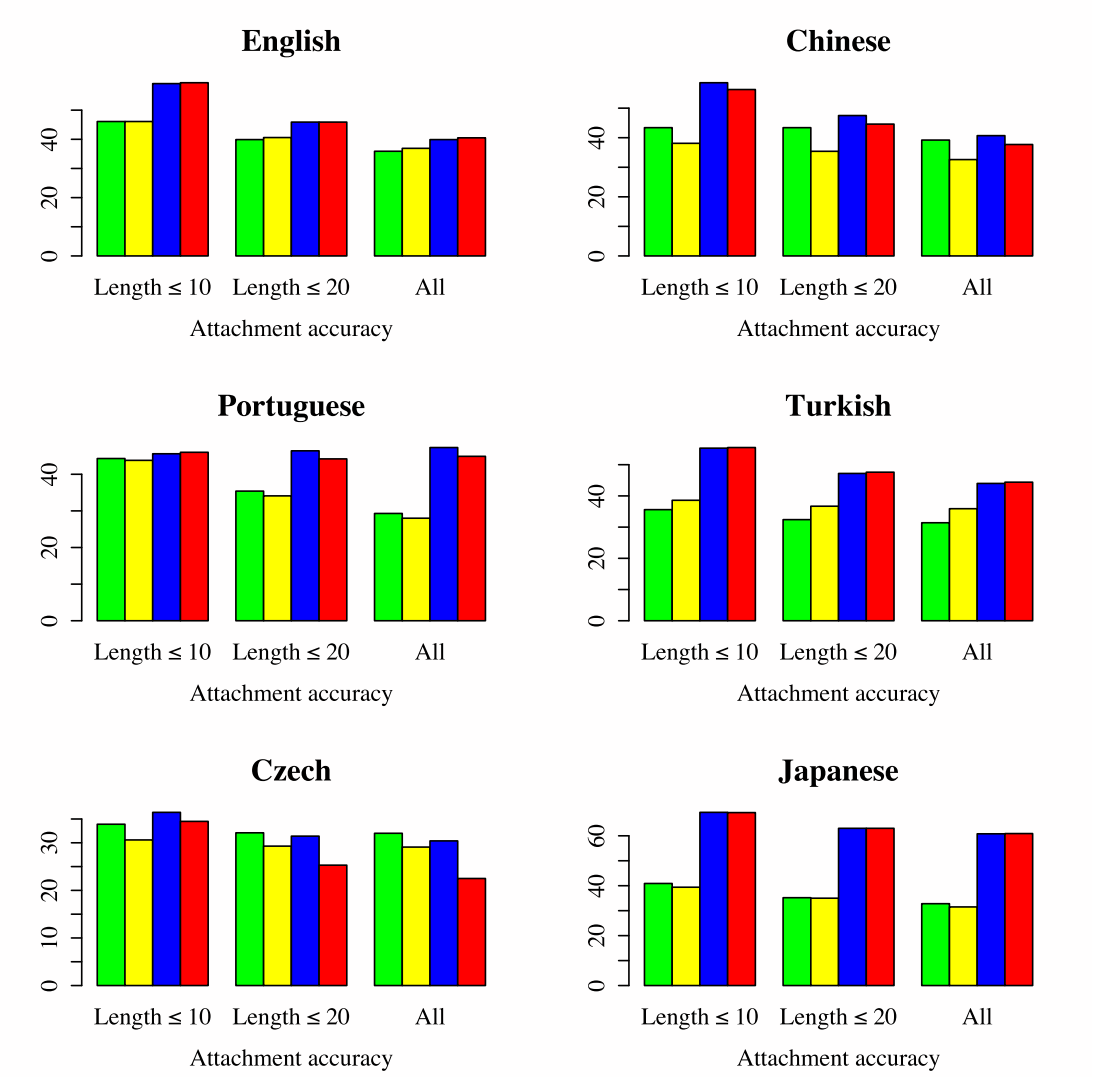


Figure 8.3: Attachment accuracy results for English (equivalent to Table 8.1), Chinese, Portuguese, Turkish, Czech and Japanese. The decoding mechanism used is MBR. Legend for the baselines: MLE (green, first column in each block); Dirichlet-I (yellow, second column); Legend for the methods in this paper: LN-I, $\boldsymbol{\Sigma}_k^{(0)} = \mathbf{I}$ (blue, third column), and LN-I, families initializer (red, fourth column).

value) than 2.3×10^{-6}.

Method	Pt	Tr	Bg	Jp	El	Sv	Es	Sl	Nl	Da	Avg
AG (Verb), MBR	45.7	39.2	52.0	66.4	59.0	**54.3**	62.3	35.3	**47.7**	**54.5**	**51.6**
AG (Noun), MBR	45.2	27.5	20.4	26.1	42.2	30.1	37.4	23.8	29.4	24.5	30.66
AG (Verb), Viterbi	45.4	31.1	53.9	65.8	**59.5**	51.0	**62.6**	34.3	47.2	50.4	47.5
AG (Noun), Viterbi	42.2	25.5	19.3	26.1	42.2	29.6	38.5	23.6	30.0	24.8	30.18
LN-I	**46.0**	**55.3**	44.0	**70.9**	45.0	42.6	31.1	21.8	28.8	42.2	42.7
EM	42.5	35.6	**54.3**	43.0	41.0	42.3	38.1	**37.0**	38.6	41.4	41.3
Dirichlet EM	43.8	38.6	47.9	41.1	50.2	43.1	21.0	34.2	39.4	40.5	39.9

Table 8.2: Attachment accuracy for various languages with sentence length ≤ 10. AG (Verb) denotes use of adaptor grammar with adaptation of verbial nonterminals, and AG (Noun) denotes use of adaptor grammars with adaptation of nominal nonterminals. Viterbi denotes Viterbi decoding and MBR denotes maximum Bayes risk decoding. LN-I denotes the use of the logistic normal prior (with model I). EM is the baseline for using the EM algorithm, while Dirichlet EM is the baseline for using empirical Bayes with variational EM for the Dirichlet prior. The languages reported in this table are: Portuguese (Pt), Turkish (Tr), Bulgarian (Bg), Japanese (Jp), Greek (El), Swedish (Sv), Spanish (Es), Slovene (Sl), Dutch (Nl) and Danish (Da). Results in bold are best results in each column.

8.5 SLN with Nouns, Verbs, and Adjectives

We now turn to experiments where the partition structure for the logistic normal prior permits covariance of parameters across multinomials, making use of the expressive power of the shared logistic normal distribution. We use a few simple heuristics to decide which partition structure $\mathcal{S}$ to apply. Our heuristics mainly rely on the centrality of content words: nouns, verbs, and adjectives. For example, in the English treebank, the most common attachment errors (with the LN prior) occur with a parent that is a noun (25.9%) or a verb (16.9%). The fact that the most common errors occur with these attachments is a result of nouns and verbs being the most common parents in the majority of the data sets that we experimented with.

Following this observation, we compare four different settings in our experiments (all SLN settings include one normal expert for each multinomial on its own, which is equivalent to the regular LN setting):

- TIEV: We add normal experts that tie all probabilities which correspond to a verbial parent (*any* verbial parent, using the coarse tags of Cohen et al., 2008). Let $\boldsymbol{V}$ be the set of part-of-speech tags that belong to the verb category. For each direction D (left or right), the set of multinomials of the form $\theta_{\mathrm{c}}(\cdot \mid v, D)$, for $v \in \boldsymbol{V}$, all share a normal expert. For each direction D and each boolean value B of the predicate $\mathrm{first}_{\mathbf{y}}(\cdot)$, the set of multinomials $\theta_{\mathrm{s}}(\cdot \mid v, D, B)$ for $v \in \boldsymbol{V}$ share a normal expert.
- TIEN: This is the same as TIEV, only for nominal parents.
- TIEV&N: Tie both verbs and nouns (in separate partitions). This is equivalent to taking the union of the partition structures for the above two settings.
- TIEA: This is the same as TIEV, only for adjectivial parents.

			English			Portuguese			Turkish		
			≤ 10	≤ 20	all	≤ 10	≤ 20	all	≤ 10	≤ 20	all
		MLE	46.1	39.9	35.9	44.3	35.4	29.3	35.6	32.4	31.4
		Dirichlet-I	46.1	40.6	36.9	43.8	34.1	28.0	38.6	36.7	35.9
		$\boldsymbol{\Sigma}_k^{(0)} = \mathbf{I}$	59.1	45.9	40.5	45.6	**45.9**	**46.5**	55.3	47.2	44.0
		families	59.4	45.9	40.5	45.9	44.0	44.4	55.5	47.6	44.4
Trained with	English	TIEV	60.2	46.2	40.0	45.4	43.7	44.5	† 56.5	**48.7**	45.5
		TIEN	60.2	46.7	40.9	45.7	44.3	45.0	51.1	43.7	41.2
		TIEV&N	61.3	47.4	41.4	46.3	44.6	45.1	55.9	48.2	45.2
		TIEA	59.9	45.8	39.6	45.4	43.8	44.6	49.8	43.2	40.8
	Portuguese	TIEV	62.1	48.1	42.2	45.2	42.3	42.3	**56.7**	† 48.6	45.1
		TIEN	60.7	46.9	40.9	45.7	42.8	42.9	33.2	29.8	28.7
		TIEV&N	61.4	47.8	42.0	46.3	44.6	45.1	**56.7**	49.2	**46.0**
		TIEA	62.1	47.8	41.8	45.2	42.7	42.7	31.5	28.4	27.5
	Turkish	TIEV	**62.5**	**48.3**	**42.4**	45.4	43.2	43.7	55.2	47.3	44.0
		TIEN	61.0	47.2	41.2	45.9	43.9	44.4	45.1	39.8	37.8
		TIEV&N	† 62.3	**48.3**	† 42.3	**46.7**	44.3	44.6	55.7	**48.7**	45.5
		TIEA	† 62.3	48.0	42.1	45.1	43.2	43.7	38.6	34.0	32.5

Table 8.3: Attachment accuracy of different monolingual tying models and bilingual tying models with varying levels of difficulty imposed through a length filter (Sections 8.5 and 8.6). Monolingual results (Section 8.5) are described when the languages in both the column and the row are identical (blocks on the diagonal). Results for MLE and Dirichlet-I are identical to Figure 8.3. Results for $\boldsymbol{\Sigma}_k^{(0)} = \mathbf{I}$ and families are identical to Table 8.1 and Figure 8.3. Each block contains the results obtained from tying one language with the other, specifying performance for the column language. Results in bold denote the best results in a column, and † marks figures that are not significantly worse (binomial sign test, $p < 0.05$).

Since learning a model with parameter tying can be computationally intensive, we first run the inference algorithm without parameter tying, and then add parameter tying to the rest of the inference algorithm's execution until convergence.

For the covariance matrices, we follow the setting described in Section 8.4. For each treebank, we divide the tags into twelve disjoint tag families. The covariance matrices for all dependency distributions were initialized with 1 on the diagonal, 0.5 between tags which belong to the same family, and 0 otherwise.

Results with MBR decoding are given in the blocks on the diagonal of Table 8.3, where the languages in the columns and rows are identical. For English, there are small improvements when adding the expressive power of SLN. The best results are achieved when tying both nouns and verbs together. Portuguese shows small benefits on shorter sentences, and when compared to the families-initialized in the LN-I model, but not in the stronger identity-initialized LN-I model. For Turkish, tying across multinomials hurts performance.

8.6 Bilingual Experiments

Leveraging linguistic information from one language for the task of disambiguating another language has received considerable attention (Dagan, 1991; Yarowsky et al., 2001; Hwa et al., 2005; Smith and Smith, 2004; Snyder and Barzilay, 2008; Burkett and Klein, 2008). Usually such a setting requires a parallel corpus or other annotated data which ties between those two languages. One notable exception is Haghighi et al. (2008), where bilingual lexicons were learned from non-parallel monolingual corpora.

Our bilingual experiments use the data for English, Portuguese, and Turkish (two at a time), which are not parallel corpora, to train parsers for two languages at a time, jointly. Sharing information between two models is accomplished by softly tying grammar weights in the two hidden grammars.

For each pair of languages, we first merge the models for these two languages by taking a union of the multinomial families from each and the corresponding prior parameters. We then add a normal expert which ties together between the parts of speech in the respective partition structures for both grammars. Parts of speech are matched through the coarse tag set. For example, with TIEV, let $\boldsymbol{V} = \boldsymbol{V}_{\text{Eng}} \cup \boldsymbol{V}_{\text{Por}}$ be the set of part-of-speech tags which belong to the verb category for either the English or Portuguese treebank (to take an example). We then tie parameters for all part-of-speech tags in $\boldsymbol{V}$. We tested this joint model for each of TIEV, TIEN, TIEV&N, and TIEA. After running the inference algorithm

which learns the two models jointly, we use unseen data to test each learned model separately.

We repeat the generative story specifically for the bilingual setting, using the example of TIEV. For each language, there are normal experts for all part-of-speech tags, for the basic DMV. In addition, there are normal experts, for each language, which combine all part-of-speech tags together that belong to the verb category. Finally, there are normal experts, for the two languages, that combine all part-of-speech tags together that belong to the verb category in either language. For each sentence in the corpus, the following two steps are conducted just as before (model I): the normal experts are sampled from the SLN distribution and combined into multinomials to parameterize the DMV; a grammar derivation is sampled from the resulting DMV.

Table 8.3 presents the results for these experiments (in the blocks which are not on the diagonal). English grammar induction shows moderate gains when tied with Portuguese and strong gains when combined with Turkish. Cohen and Smith (2009) reported qualitatively similar results when English was tied with Chinese. For Portuguese, there is not much gain based on tying it with other languages, though doing so does improve the performance of the other two languages. In general, the table shows that with the proper selection of a pair of languages and multinomials for tying, we can usually get improvement over the LN baselines and the technique does not hurt performance (cf. Turkish grammar induction with SLN, on its own). We note that selection of multinomials for tying encodes prior knowledge about the languages. This knowledge simply requires being able to map fine-grained, treebank-specific part-of-speech tags to coarse categories. In addition, bilingual learning with SLN does not require bitext parsing at any point, which is an expensive operation. The runtime of the variational E-step for a sentence $\boldsymbol{x}$ is still cubic in the length of $\boldsymbol{x}$, just as in EM, thus as a result, the runtime of the variational E-step in the multilingual case is the same as it would be if we added an equivalent amount of data in the monolingual case.

8.7 Error Analysis

We now include some error analysis of the results we get for the English treebank using the logistic normal distribution and adaptor grammars. We choose to focus on English for the logistic normal distribution and Spanish for adaptor grammars. These two languages have high performance with respect to each method that we use.

8.7.1 Confusion Matrices

	Noun	Conjunction	Foreign	Verb	Adjective	Number	Pronoun	Interjection	Particle	Adverb	Preposition	Proper
Noun	3190	114	1	2577	52	186	396	3	1381	13	2041	205
Conjunction	1	0	0	4	0	0	0	0	0	0	0	0
Foreign	0	0	3	2	0	0	1	0	1	0	1	0
Verb	2769	179	5	3693	124	138	774	3	888	88	604	108
Adjective	260	19	0	281	15	9	39	0	87	7	40	1
Number	62	1	0	51	4	35	26	0	30	0	37	0
Pronoun	149	3	0	128	11	12	76	0	10	7	18	24
Interjection	0	0	0	0	0	0	0	1	0	0	0	0
Particle	223	8	0	438	5	14	18	0	58	2	111	13
Adverb	158	11	3	284	29	11	40	1	11	5	19	2
Preposition	545	121	3	3152	62	91	240	0	265	20	949	70
Proper	387	63	2	532	26	41	91	3	122	11	428	579

Table 8.4: A confusion matrix that shows the kinds of errors we get when using the logistic normal prior. Each cell in the table corresponds to a count of the number of times that the part-of-speech tag heading the column was predicted as an incorrect parent, rather than the part-of-speech tag heading the row. Across the diagonals, counts are given for errors in prediction of a parent where it has the same part-of-speech, but it is predicted for the wrong position in the sentence. Note that we reduced the original tagset in the Penn treebank to a coarse part-of-speech tagset for this table.

	Noun	Conjunction	Foreign	Verb	Adjective	Number	Pronoun	Interjection	Particle	Adverb	Preposition	Proper
Noun	0.6 / 1454	0 / 0	1 / 1	0.4 / 1554	1.0 / 51	0.9 / 13	0.9 / 33	1 / 1	1.0 / 444	1 / 116	1.0 / 3488	0.9 / 146
Conjunction	0.4 / 206	0 / 0	0 / 0	0.9 / 445	0.5 / 32	1 / 19	0.3 / 2	0 / 0	1 / 10	1 / 19	1 / 47	1.0 / 235
Foreign	1 / 2	0 / 0	0.5 / 3	0.8 / 3	1 / 2	0 / 0	0 / 0	0 / 0	0 / 0	1 / 3	0.5 / 1	1 / 1
Verb	0.6 / 1016	1 / 1	0.5 / 2	0.7 / 2791	0.6 / 106	0.3 / 44	0.7 / 383	0 / 0	0.7 / 33	0.4 / 52	0.8 / 720	0.4 / 147
Adjective	0.4 / 1202	0 / 0	0 / 0	0.5 / 277	1.0 / 80	1 / 20	0.7 / 8	0 / 0	0.9 / 31	0.9 / 18	0.8 / 97	1.0 / 88
Number	0.6 / 570	1 / 1	0 / 0	0.7 / 153	1 / 25	0.6 / 59	0 / 0	0 / 0	0.7 / 42	1 / 91	0.6 / 203	1 / 135
Pronoun	0.5 / 373	1 / 3	0 / 0	0.1 / 129	1 / 13	1 / 6	1 / 6	0 / 0	0.9 / 29	0.9 / 8	1 / 75	1 / 95
Interjection	1 / 1	0 / 0	0 / 0	0.3 / 1	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	1 / 1	0 / 0	0 / 0
Particle	0.4 / 2056	0 / 0	1 / 1	0.7 / 971	1.0 / 77	1.0 / 32	0 / 0	0 / 0	1 / 168	1.0 / 42	0.4 / 38	1.0 / 416
Adverb	0.9 / 228	0 / 0	0 / 0	0.4 / 568	0.6 / 157	0.8 / 10	0.2 / 2	0 / 0	0.9 / 16	1.0 / 112	1.0 / 160	1 / 14
Preposition	0.9 / 2118	0 / 0	0 / 0	0.8 / 2120	1.0 / 200	0.7 / 34	1 / 1	0 / 0	1 / 42	1 / 104	1.0 / 143	1 / 152
Proper	0.9 / 933	0 / 0	1 / 1	0.4 / 361	1 / 15	1 / 9	1 / 3	0 / 0	0.1 / 75	1 / 8	0.6 / 546	0.3 / 856

Table 8.5: A confusion matrix that shows which dependencies we most often get wrong when using the logistic normal prior. In each cell in the table, with x/y, we have that y corresponds to the total count of links of the form $c \rightarrow r$ in the parsed data, where r is the part-of-speech tag heading the row and c is the part-of-speech tag heading the column. We also have that x corresponds to the fraction of times that we wrongly predicted the link $c \rightarrow r$, out of the total number of times it appeared. Note that we reduced the original tagset in the Penn treebank to a coarse part-of-speech tagset for this table.

Table 8.4 and Table 8.5 give confusion matrices for various errors when using the logistic normal distribution for English. Most of the errors, as can be seen from Table 8.4 occur as a result of confusion between several nouns in the sentence (or several verbs). This means that the parser was able to correctly predict that the parent should be a noun (or a verb), but chose the wrong noun (or verb) in the sentence as the parent. Another type of confusion happens between verbs, nouns and prepositions. It seems that verbs are often predicted as parents when a noun or a preposition should instead be the parent. Interestingly enough, it is also often the case that prepositions are incorrectly predicted as parents, where nouns are the correct parents, but this confusion does not happen as much with verbs and prepositions.

The large confusion numbers that occur with prepositions may partially be explained by the head rules that we chose to use (Yamada and Matsumoto rules; see earlier note). It could be that our parser learns a different annotation scheme than that used by these rules. The choice of these head rules for prepositions has been under debate in the computational linguistics community. It is not clear whether a preposition word should head a prepositional phrase (as in the Yamada and Mastumoto rules), or whether other words in the phrase (such as the noun) should act as the heads. It is important to note, however, that the high count of confusion between nouns, verbs and prepositions could also occur because of their high frequency in the English treebank.

When inspecting Table 8.5, we see that the logistic normal prior tends to identify verb noun relationships moderately well. However, it also tends to attach other part-of-speech tags to the verb (and to nouns). It also seems that the PP-attachment problem is quite hard for the unsupervised learner. For noun $\rightarrow$ preposition dependencies, the unsupervised learner errs in 90% of the cases. For verb $\rightarrow$ preposition dependencies, the unsupervised learner errs in 0.8% of the cases.

Table 8.6 and Table 8.7 give an analoguous analysis for using adaptor grammars with Spanish.

	Noun	Conjunction	Verb	Adjective	Punctuation	Number	Pronoun	Interjection	Adverb	Particle	.	Preposition	Unknown
Noun	162	26	194	64	0	2	4	0	126	22	0	0	0
Conjunction	10	4	26	10	0	0	1	0	4	2	0	0	0
Verb	214	70	414	93	0	1	20	0	99	71	0	0	0
Adjective	139	11	87	26	0	0	0	0	55	10	0	0	0
Punctuation	0	0	0	0	0	0	0	0	0	0	0	0	0
Number	2	2	10	3	0	0	0	0	4	1	0	0	0
Pronoun	13	3	28	7	0	0	1	0	11	3	0	0	0
Interjection	0	0	0	0	0	0	0	0	0	0	0	0	0
Adverb	91	17	161	17	0	1	0	0	112	64	0	0	0
Particle	8	0	10	2	0	0	0	0	9	0	0	0	0
.	0	0	0	0	0	0	0	0	0	0	0	0	0
Preposition	0	0	0	0	0	0	0	0	0	0	0	0	0
Unknown	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 8.6: A confusion matrix that shows the kinds of errors we get when using adaptor grammars for Spanish. Each cell in the table corresponds to a count of the number of times that the part-of-speech tag heading the column was predicted as an incorrect parent, rather than the part-of-speech tag heading the row. Across the diagonals, counts are given for errors in prediction of a parent where it has the same part-of-speech, but it is predicted for the wrong position in the sentence. Note that we reduced the original tagset in the Spanish treebank to a coarse part-of-speech tagset for this table.

	Noun	Conjunction	Verb	Adjective	Punctuation	Number	Pronoun	Interjection	Adverb	Particle	.	Preposition	Unknown
Noun	1.0 / 119	0.9 / 18	0.5 / 190	1.0 / 91	0 / 0	0 / 0	0.8 / 5	0 / 0	0.2 / 154	1 / 1	0 / 0	0 / 0	0 / 0
Conjunction	0.8 / 62	1 / 10	0.8 / 169	0.9 / 18	0 / 0	1 / 1	0.9 / 6	0 / 0	1.0 / 26	1 / 2	0 / 0	0 / 0	0 / 0
Verb	0.7 / 120	1.0 / 20	0.8 / 236	0.5 / 30	0 / 0	0.4 / 4	0.9 / 14	0 / 0	0.9 / 96	1 / 1	0 / 0	0 / 0	0 / 0
Adjective	0.7 / 148	1 / 1	0.8 / 54	1 / 42	0 / 0	1 / 1	1 / 6	0 / 0	1.0 / 69	0 / 0	0 / 0	0 / 0	0 / 0
Punctuation	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0
Number	0.6 / 4	0 / 0	0.8 / 9	0.8 / 3	0 / 0	1 / 1	0 / 0	0 / 0	0.5 / 12	0 / 0	0 / 0	0 / 0	0 / 0
Pronoun	0.8 / 5	1 / 1	0.4 / 83	1 / 1	0 / 0	0 / 0	1 / 2	0 / 0	0.6 / 25	0.5 / 1	0 / 0	0 / 0	0 / 0
Interjection	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0
Adverb	0.3 / 117	1 / 6	0.5 / 228	0.9 / 82	0 / 0	1 / 5	0.9 / 17	0 / 0	0.9 / 69	1 / 2	0 / 0	0 / 0	0 / 0
Particle	0.0 / 25	1 / 1	0.9 / 13	0.8 / 61	0 / 0	1 / 10	0.9 / 16	0 / 0	0.7 / 11	0.8 / 22	0 / 0	0 / 0	0 / 0
.	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0
Preposition	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0
Unknown	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	1 / 1	0 / 0	0 / 0	0 / 0	0 / 0

Table 8.7: A confusion matrix that shows which dependencies we most often get wrong when using adaptor grammars. In each cell in the table, with x/y, we have that y corresponds to the total count of links of the form $c \rightarrow r$ in the parsed data, where r is the part-of-speech tag heading the row and c is the part-of-speech tag heading the column. We also have that x corresponds to the fraction of times that we wrongly predicted the link $c \rightarrow r$, out of the total number of times it appeared. Note that we reduced the original tagset in the Spanish treebank to a coarse part-of-speech tagset for this table.

8.7.2 Dependency Properties

When inspecting Figure 8.4, we note that a tendency exists for the logistic normal parser to output trees which are more shallow than the trees that appear in the treebank – their depth tends to be of smaller degree. One might expect that in the case of having more shallow trees the dependency lengths would in general be of a higher degree (because, for example, a single parent would have to be the parent of children farther away from its position in the sentence), but surprisingly enough, this does not happen. When inspecting Figure 8.4, we see that the distribution of the lengths of dependency links for parsed data and gold-standard data is actually similar.

8.7.3 Probability Values Set During Learning

Both estimation methods for the logistic normal prior family and adaptor grammars eventually yield a point estimate for a given grammar. These point estimates for the grammar can be used to extract information about the most salient features that the models learn – in other words, the dependency affinities between part-of-speech tags.

In this section, we consider such affinities for two languages: English and Spanish. While with English, the logistic normal considerably outperforms adaptor grammars, the situation is the opposite for Spanish (Table 8.2). Figure 8.5 and Figure 8.6 describe the affinity matrices (right and left dependencies) for these languages as compared with the matrices from a maximum likelihood estimate obtained from annotated data. More specifically, the figures describe heatmaps that illustrate the difference between two quantities: the probability of emitting a certain part-of-speech tag as the child of another part-of-speech tag according to the learned models, and the probability of emitting a certain part-of-speech tag as the child of another part-of-speech tag according to annotated data ML estimate.

Interestingly enough, at first glance, the heatmaps for both adaptor grammars and for the logistic normal seem very similar, despite the great differences in performance. In both cases, most of the learned dependency probabilities (i.e. values for the attachment parameters) are close to the probabilities learned with annotated data, with some outliers that have a higher difference. However, when we consider, for example, the matrices for the English treebank, we see that the logistic normal tends to give higher probabilities than the supervised MLE solution more often than adaptor grammars tend to. This is especially true for dependencies with a left direction. The situation is even more clear for Spanish, for which

Adaptor grammars tend to perform much better than the logistic normal prior. Adaptor grammars tend to give higher probabilities than the supervised MLE solution more often than the logistic normal does. For the right direction, for example, it seems like most of the probabilities that the adaptor grammar solution gives are *higher* than the supervised MLE solution, while the opposite happens for the logistic normal.

We hypothesize that in order to obtain a better parser, it is preferable to give higher rather than a lower probability to salient dependencies. This perhaps may complement the view that sparsity can help grammar induction (Chapter 7): when the solutions are sparse, more probability mass may be allocated to the salient dependencies.

8.8 Summary

In this chapter, we have presented the main empirical results for dependency grammar induction while using the estimation techniques that we present in this dissertation. Our results indicate that the logistic normal prior and adaptor grammars did better on average than several baselines, including EM and variational EM with a Dirichlet prior. We have also described an empirical setting in which the shared logistic normal distribution was used for bilingual learning from non-parallel corpora. Finally, we have also compared and contrasted the behavior of adaptor grammars and logistic normal priors for English and Spanish.

It is important to note that there has been a large body of work on dependency grammar induction since the mid 2000s, and most notably, since 2009. This work is mentioned in Chapter 7. In many cases, as reported by Schwartz et al. (2011), the use of different datasets and different head rules precludes a direct comparison of our performance to previous work. If we disregard these differences for a moment and directly compare numbers, our results are state-of-the-art for certain languages or very close to state-of-the-art, while for other languages they are not. It is important to note that our methods in certain cases are orthogonal to the methods presented in the abovementioned work. For example, Headden et al. (2009) presented a different model than the DMV model, which was partially lexicalized. In principle, we could use our estimation methods to estimate the grammar presented in Headden et al., but we leave this for future work. We also note that adaptor grammars are not limited to use with a selection of a Dirichlet distribution as a prior for the grammar rules. Our variational inference algorithm, for example, can be extended with some effort to use with the logistic normal prior rather than

the Dirichlet.

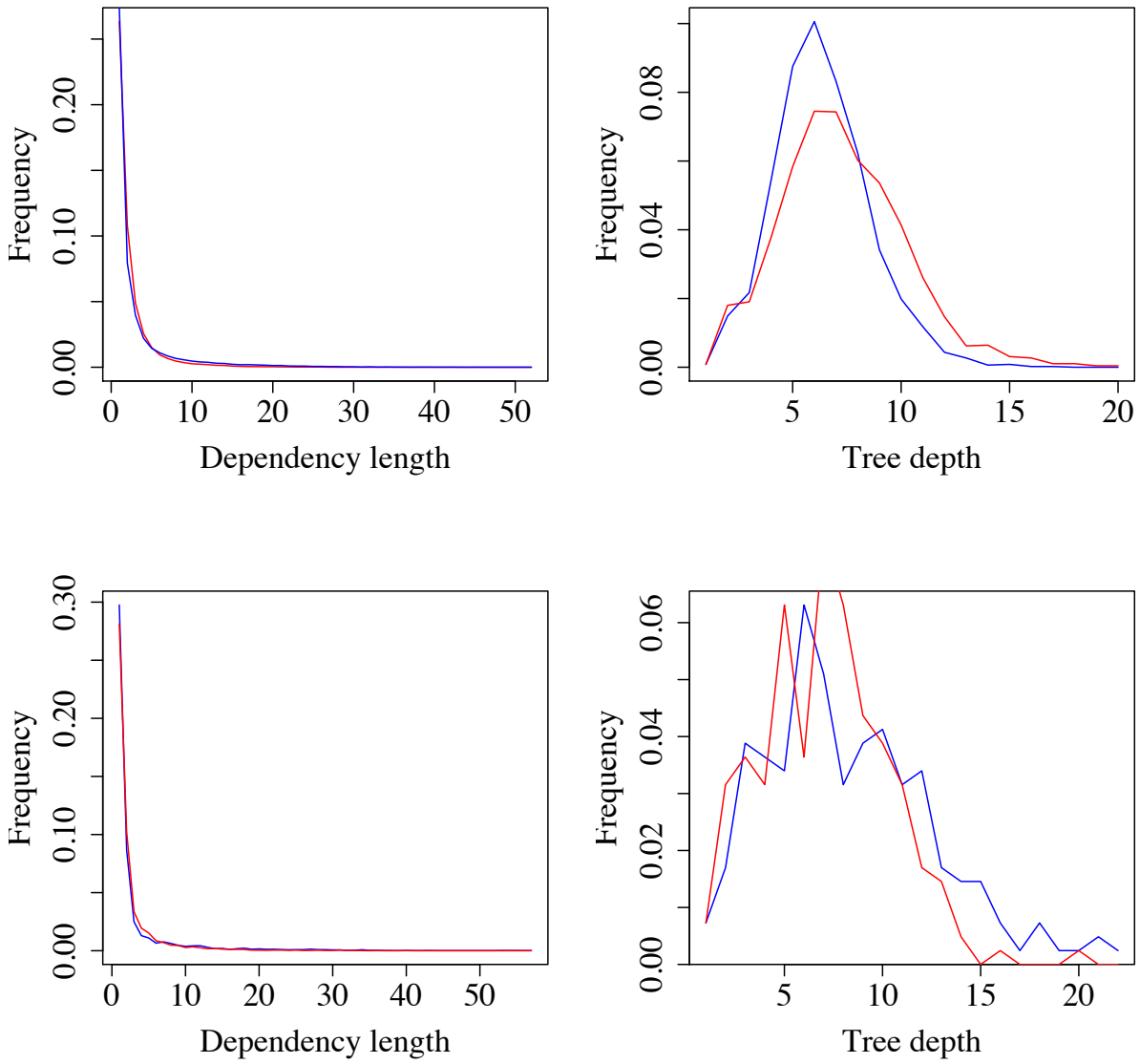


Figure 8.4: Distribution of lengths and depths of dependency links in gold standard data (red) and when using the shared logistic normal distribution or adaptor grammars (blue). Top left: Length distribution plot for the LN model for English. Top right: Depth distribution plot for the LN model for English. Bottom left: Length distribution plot for the AG model for Spanish. Bottom right: Depth distribution plot for the LN model for Spanish.

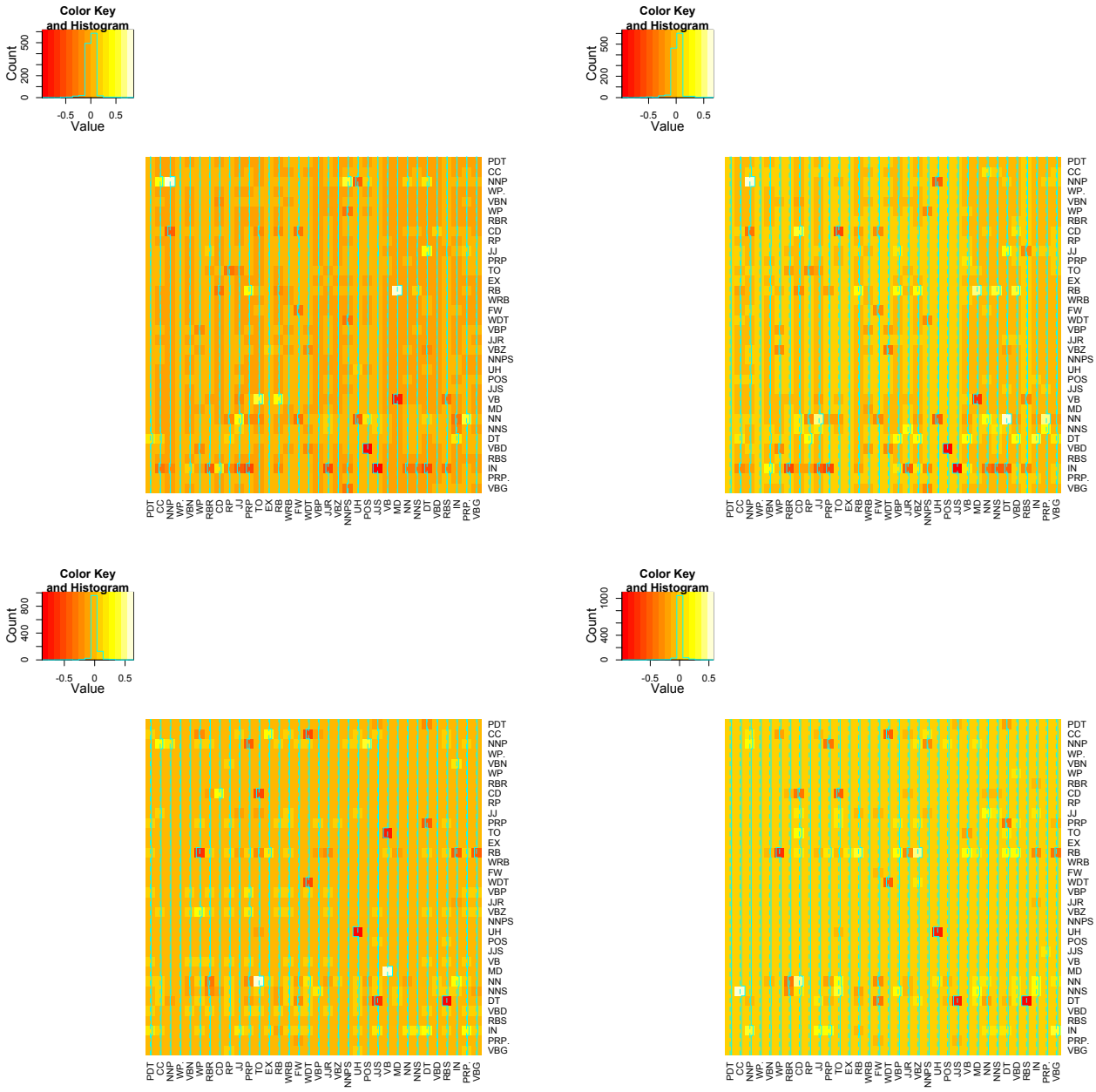


Figure 8.5: Dependency affinity heatmaps for the English data for the logistic normal distribution and adaptor grammars as compared to the supervised maximum likelihood solution. Each square in the heatmap denotes a dependency where the part-of-speech tag heading the row is the parent and the part-of-speech tag heading the column is the child. Intensity refers to the difference between the probability given by an LN or AG solution and the probability given by a supervised MLE solution. Top-left: differences between right dependencies using the logistic normal prior. Top-right: differences between right dependencies using adaptor grammars. Bottom-left: differences between left dependencies using the logistic normal prior. Bottom-right: differences between left dependencies using adaptor grammars.

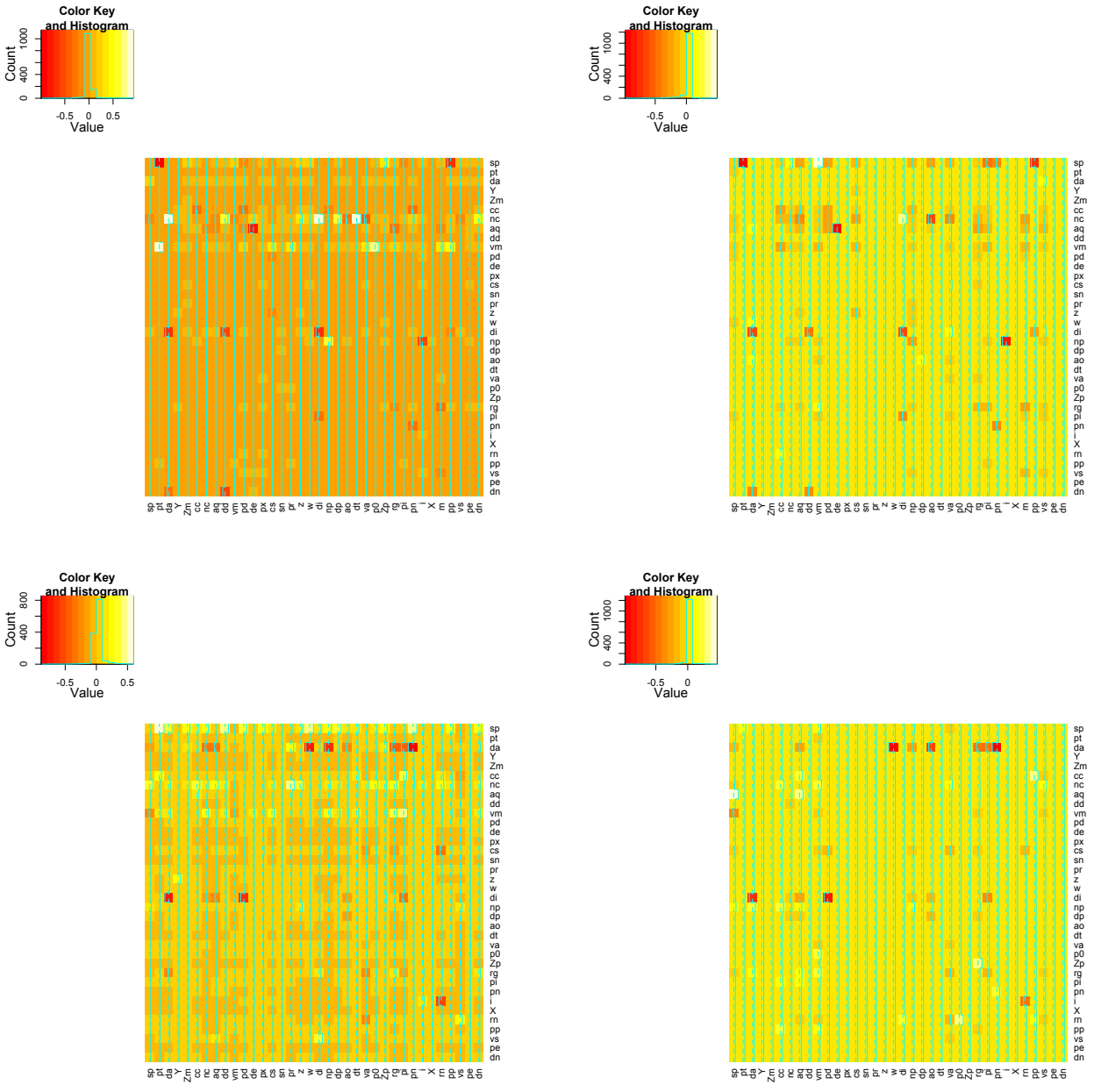


Figure 8.6: Dependency affinity heatmaps for the Spanish data for the logistic normal distribution and adaptor grammars as compared to the supervised maximum likelihood solution. Each square in the heatmap denotes a dependency where the part-of-speech tag heading the row is the parent and the part-of-speech tag heading the column is the child. Intensity refers to the difference between the probability given by an LN or AG solution and the probability given by a supervised MLE solution. Top-left: differences between right dependencies using the logistic normal prior. Top-right: differences between right dependencies using adaptor grammars. Bottom-left: differences between left dependencies using the logistic normal prior. Bottom-right: differences between left dependencies using adaptor grammars.

Chapter 9

Summary and Future Work

In this dissertation we have presented a study of the computational properties of estimating probabilistic grammars in the unsupervised setting. We showed that probabilistic grammars are learnable under the maximum likelihood criterion, though inference is hard. We showed how to estimate grammars in the Bayesian setting, achieving high performance despite the computational challenges.

Our application was grammar induction. To the best of our knowledge, we are the first to use a Bayesian framework for grammar induction with (or without) a non-conjugate prior.

9.1 Future Directions

We conclude with a few possible future directions and open problems.

9.1.1 Learning-Theoretic Analysis of Probabilistic Grammars

We describe two open problems from Chapter 4.

Sample Complexity Bounds with Semi-supervised Learning Our bounds focus on the supervised case and the unsupervised case. There is a trivial extension to the semi-supervised case. Consider the objective function to be the sum of the likelihood for the labeled data together with the marginalized likelihood of the unlabeled data (this sum could be a weighted sum). Then, use the sample complexity bounds for each summand, to derive a sample complexity bound on this sum.

It would be more interesting to extend our results to frameworks such as the one described by Balcan and Blum (2010). In that case, our discussion of sample complexity would attempt to see identify how unannotated data can reduce the space of candidate probabilistic grammars to a smaller set, after which we can use the annotated data to estimate the final grammar. This reduction of the space is accomplished through a notion of compatibility, a type of fitness that the learner believes the estimated grammar should have given the distribution that generates the data. The key challenge in the case of probabilistic grammars would be to properly define this compatibility notion such that it fits the log-loss. If this is achieved, then similar machinery to that described in this paper (with proper approximations) can be followed to derive semi-supervised sample complexity bounds for probabilistic grammars.

Sharper Bounds for the Pseudo-dimension of Probabilistic Grammars The pseudo-dimension of a probabilistic grammar with the log-loss is bounded by the number of parameters in the grammar, because the logarithm of a distribution generated by a probabilistic grammar is a linear function. However, typically the set of counts for the feature vectors of a probabilistic grammar resides in a subspace of a dimension which is smaller than the full dimension specified by the number of parameters. The reason for this is that there are usually relationships (which are often linear) between the elements in the feature counts. For example, with hidden Markov models, the total feature count for emissions should equal the total feature count for transitions. With PCFGs, the total number of times that nonterminal rules fire equals the total number of times that features with that nonerminal in the righthand side fired, again reducing the pseudo-dimension. An open problem that remains is characterization of the exact value pseudo-dimension for a given grammar, determined by consideration of various properties of that grammar. We conjecture, however, that a lower bound on the pseudo-dimension would be rather close to the full dimension of the grammar (the number of parameters).

It is interesting to note that there has been some work to identify the VC dimension and pseudo-dimension for certain type of grammars. Bane et al. (2010), for example, calculated the VC dimension for constraint-based grammars. Ishigami and Tani (1993) and Ishigami and Tani (1997) computed the VC dimension for finite state automata with various properties.

9.1.2 Partition Structure and Covariance Matrices

The partition structure in Chapter 5 is fixed. It originates in prior knowledge about the part-of-speech tags and how they reduce to coarser part-of-speech tags. One possible future direction is to learn this partition structure automatically. The partition structure is considered a hyperparameter for the shared logistic normal distribution, and just like we estimate these parameters, it might be possible to "estimate" the partition structure as well.

Another interesting direction to explore is working in a complete Bayesian setting, and not empirical Bayes. It seems like the estimation of the mean values and covariance matrices for the shared logistic normal distribution is crucial to get good performance. Perhaps instead of estimating these mean values and covariance matrices, it is possible to place an additional prior (hyperprior) over these hyperparameters. More specifically, we can use the inverse-Wishart distribution as a prior for the covariance matrices, as it is conjugate to the normal distribution with respect to the covariance parameters.

9.1.3 Extensions to the Nonparametric Setting

There are other approaches to use nonparametric modeling with grammars, other than adaptor grammars. For example, Liang et al. (2007) and Finkel et al. (2007) used a Dirichlet process to split the states in a grammar into more fine-grained set of states. These approaches are orthogonal to the use of Dirichlet process with adaptor grammars. It would be interesting to combine both of these approaches, to yield a state-split adaptor grammar. The variational inference framework that we give in this paper is sufficiently flexible to contain both of these approaches in a unified framework.

Another work related to adaptor grammars is that of *fragment grammars*. Fragment grammars (O'Donnell et al., 2009) are also nonparametric models which are similar to adaptor grammars. The main distinction is that with fragment grammars the model "grows" a tree not necessarily up to the leaves, like is the case with adaptor grammars. Fragment grammars are currently used with a sampler which is similar to the sampler devised by Johnson et al. (2006). Our variational inference framework again can be extended to handle fragment grammars as well.

Last, we leave for future work a connection in this dissertation that could be further exploited: using logistic normal priors as a prior for the base context-free grammar in an adaptor grammar. Our variational inference frameworks with the logistic normal prior and adaptor grammars permits a modular extension to this

setting.

9.1.4 Connecting Better Between the Estimation Techniques and the Theoretical Analysis

In the first part of this thesis, we presented a theoretical analysis of the estimation problem of probabilistic grammars, both from a learning-theoretic perspective and from a computational complexity perspective. This analysis is set up for maximum likelihood estimation.

In the second part of this thesis, we presented two empirical Bayes estimation techniques. Empirical Bayes has a strong connection to maximum likelihood estimation, but still, it could be fruitful to explore further how the learning-theoretic framework we presented can be extended to this setting. As far as the computational complexity analysis, we believe that the results about the hardness of maximizing the MLE objective function should extend (with efforts) to the empirical Bayes setting we presented in this thesis.

Still, we note that the empirical Bayes techniques that we presented in this thesis focus on identifying a *point estimate* for probabilistic grammars. This implies that the learning-theoretic framework we presented extends trivially when eventually evaluating the regular log likelihood from the point estimate of the grammar, instead of the full likelihood which includes the Bayesian prior.

9.1.5 New Models and New Applications

Probabilistic grammars is a wide family of statistical models. Probabilistic grammars other than the dependency model with valence have been devised for dependency grammar induction, such as the one in Headden et al. (2009). It would be interesting to apply the estimation techniques to such models and see whether performance gains emerge both from better modeling as well as from better estimation. Testing the usefulness of the estimation methods we presented in this thesis with lexicalized data is interesting as well. Unsupervised part-of-speech tagging can be used to tag raw text, after which we can use our estimation methods. Similar approach has been taken in Cohen et al. (2011a).

More generally, probabilistic grammars can be useful outside of natural language processing. They have applications in computer vision (Lin et al., 2009), computational biology (Sakakibara et al., 1994) and more recently, in human activity analysis (Guerra and Aloimonos, 2005). It could be very fruitful to apply

the tools which were developed in this dissertation to these fields.

Bibliography

Abe, N., Takeuchi, J., and Warmuth, M. (1990). Polynomial learnability of probabilistic concepts with respect to the Kullback-Leiber divergence. In *ACM Conference on Computational Learning Theory*.

Abe, N. and Warmuth, M. (1992). On the computational complexity of approximating distributions by probabilistic automata. *Machine Learning*, 2:205–260.

Abney, S. (2007). *Semisupervised Learning for Computational Linguistics*. CRC Press.

Adriaans, P. W., Trautwein, M., and Vervoort, M. (2000). Towards high speed grammar induction on large text corpora. In *SOFSEM: Proceedings of the 27th Conference on Current Trends in Theory and Practice of Informatics*.

Afonso, S., Bick, E., Haber, R., and Santos, D. (2002). Floresta sinta(c)tica: a treebank for Portuguese. In *Proceedings of LREC*.

Ahmed, A. and Xing, E. (2007). On tight approximate inference of the logistic normal topic admixture model. In *Proceedings of AISTATS*.

Aitchison, J. (1986). *The Statistical Analysis of Compositional Data*. Chapman and Hall, London.

Aloise, D., Deshpande, A., Hansen, P., and Popat, P. (2009). NP-hardness of Euclidean sum-of-squares clustering. *Machine Learning*, 75(2):245–248.

Alshawi, H. and Buchsbaum, A. L. (1996). Head automata and bilingual tiling: Translation with minimal representations. In *Proceedings of ACL*.

Angluin, D. (1988). Learning regular sets from queries and counterexamples. *Information and Computation*, pages 87–106.

Anthony, M. and Bartlett, P. L. (1999). *Neural Network Learning: Theoretical Foundations*. Cambridge University Press.

Antoniak, C. (1974). Mixtures of Dirichlet processes with applications to Bayesian nonparametric problems. *The Annals of Statistics*, 2(6):1152–1174.

Atalay, N. B., Oflazer, K., and Say, B. (2003). The annotation process in the Turkish treebank. In *Proceedings of LINC*.

Baker, J. (1979). Trainable grammars for speech recognition. In *The 97th meeting of the Acoustical Society of America*.

Balcan, M. and Blum, A. (2010). A discriminative model for semi-supervised learning. *Journal of the ACM*, 57(3).

Bane, M., Riggle, J., and Sonderegger, M. (2010). The VC dimension of constraint-based grammars. *Lingua*, 120(5):1194–1208.

Banerjee, A. (2006). On Bayesian bounds. In *Proceedings of ICML*.

Bartlett, P., Bousquet, O., and Mendelson, S. (2005). Local Rademacher complexities. *Annals of Statistics*, 33(4):1497–1537.

Berg-Kirkpatrick, T., Bouchard-Cote, A., DeNero, J., and Klein, D. (2010). Painless unsupervised learning with features. In *Proceedings of NAACL*.

Berg-Kirkpatrick, T. and Klein, D. (2010). Phylogenetic grammar induction. In *Proceedings of ACL*.

Berger, J. O. (2010). *Statistical Decision Theory and Bayesian Analysis*. Springer.

Bernstein-Ratner, N. (1987). The phonology of parent child speech. *Children's Language*, 6.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

Blei, D. and Jordan, M. (2005). Variational inference for Dirichlet process mixtures. *Journal of Bayesian Analysis*, 1(1):121–144.

Blei, D. M. and Lafferty, J. D. (2006). Correlated topic models. In *Proceedings of NIPS*.

Blei, D. M., Ng, A., and Jordan, M. (2003). Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Blunsom, P. and Cohn, T. (2010). Unsupervised induction of tree substitution grammars for dependency parsing. In *Proceedings of EMNLP*.

Bod, R. (2006a). An all-subtrees approach to unsupervised parsing. In *Proceedings of COLING*.

Bod, R. (2006b). Unsupervised parsing with U-DOP. In *CoNLL-X*.

Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge Press.

Boyd-Graber, J. and Blei, D. M. (2010). Syntactic topic models. In *ArXiv*.

Brent, M. and Cartwright, T. (1996). Distributional regularity and phonotactic constraints are useful for segmentation. *Cognition*, 6:93–125.

Brown, P. F., Pietra, V. J. D., deSouza, P. V., Lai, J. C., and Mercer, R. L. (1990). Class-based n-gram models of natural language. *Computational Linguistics*.

Burkett, D. and Klein, D. (2008). Two languages are better than one (for syntactic parsing). In *Proceedings of EMNLP*.

Carlin, B. P. and Louis, T. A. (2000). *Bayes and Empirical Bayes Methods for Data Analysis*. Chapman & Hall/CRC.

Carrasco, R. (1997). Accurate computation of the relative entropy between stochastic regular grammars. *Theoretical Informatics and Applications*, 31(5):437–444.

Carroll, G. and Charniak, E. (1992). Two experiments on learning probabilistic dependency grammars from corpora. Technical report, Brown University.

Casacuberta, F. and de la Higuera, C. (2000). Computational complexity of problems on probabilistic grammars and transducers. In *Proceedings of ICGI*.

Charniak, E. (1996). *Statistical Language Learning*. MIT Press.

Charniak, E. (1997). Statistical parsing with a context-free grammar and word statistics. In *Proceedings of National Conference on Artificial Intelligence*.

Charniak, E. and Johnson, M. (2005). Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of ACL*.

Chater, N. and Vitányi, P. (2007). 'ideal learning' of natural language: Positive results about learning from positive evidence. *Journal of Mathematical Psychology*, 51(3):135–163.

Chen, S. F. (1995). Bayesian grammar induction for language modeling. In *Proceedings of ACL*, pages 228–235.

Chi, Z. (1999). Statistical properties of probabilistic context-free grammars. *Computational Linguistics*, 25(1):131–160.

Chiang, D. (2005). A hierarchical phrase-based model for statistical machine translation. In *Proceedings of ACL*.

Choi, Y. and Cardie, C. (2007). Structured local training and biased potential functions for conditional random fields with application to coreference resolution. In *Proceedings of HLT-NAACL*.

Civit, M. and Martí, M. (2004). Building cast3lb: A Spanish treebank. *Research on Language & Computation*, 2(4):549–574.

Clark, A. (2000). Inducing syntactic categories by context distribution clustering. In *Proceedings of Conference on Computational Natural Language Learning*, pages 91–94, Lisbon, Portugal.

Clark, A. (2010a). Learning context free grammars with the syntactic concept lattice. In Sempere, J. and Garcia, P., editors, *Grammatical Inference: Theoretical Results and Applications. Proceedings of the International Colloquium on Grammatical Inference*, number 6339 in Lecture Notes in Computer Science, pages 38–51. Springer.

Clark, A. (2010b). Towards general algorithms for grammatical inference. In *Proceedings of ALT*, pages 11–30. Springer, Canberra, Australia. Invited Paper.

Clark, A., Eyraud, R., and Habrard, A. (2010). Using contextual representations to efficiently learn context-free languages. *Journal of Machine Learning Research*, 11:2707–2744.

Clark, A. and Lappin, S. (2010). Unsupervised learning and grammar induction. In Clark, A., Fox, C., and Lappin, S., editors, *The Handbook of Computational Linguistics and Natural Language Processing*, pages 197–220. Wiley-Blackwell.

Clark, A. and Lappin, S. (2011). Computational learning theory and language acquisition. In Kempson, R., Asher, N., and Fernando, T., editors, *Handbook of Philosophy of Linguistics*. Elsevier. forthcoming.

Clark, A. and Thollard, F. (2004). PAC-learnability of probabilistic deterministic finite state automata. *Journal of Machine Learning Research*, 5:473–497.

Clark, S. and Curran, J. R. (2007). Wide-coverage efficient statistical parsing with ccg and log-linear models. *Computational Linguistics*, 33(4):493–552.

Cohen, S. B., Blei, D. M., and Smith, N. A. (2010). Variational inference for adaptor grammars. In *Proceedings of NAACL*.

Cohen, S. B., Das, D., and Smith, N. A. (2011a). Unsupervised structure prediction with non-parallel multilingual guidance. In *Proceedings of EMNLP*.

Cohen, S. B., Gimpel, K., and Smith, N. A. (2008). Logistic normal priors for unsupervised probabilistic grammar induction. In *NIPS*.

Cohen, S. B., Gómez-Rodríguez, C., and Satta, G. (2011b). Exact inference for generative probabilistic non-projective dependency parsing. In *Proceedings of EMNLP*.

Cohen, S. B. and Smith, N. A. (2007). Joint morphological and syntactic disambiguation. In *Proceedings of EMNLP-CoNLL*.

Cohen, S. B. and Smith, N. A. (2009). Shared logistic normal distributions for soft parameter tying in unsupervised grammar induction. In *Proceedings of HLT-NAACL*.

Cohen, S. B. and Smith, N. A. (2010a). Covariance in unsupervised learning of probabilistic grammars. *Journal of Machine Learning Research*, 11:3017–3051.

Cohen, S. B. and Smith, N. A. (2010b). Empirical risk minimization with approximations of probabilistic grammars. In *NIPS*.

Cohen, S. B. and Smith, N. A. (2010c). Viterbi training for PCFGs: Hardness results and competitiveness of uniform initialization. In *Proceedings of ACL*.

Cohen, S. B. and Smith, N. A. (2011). Empirical risk minimization for probabilistic grammars: Sample complexity and hardness of learning. *Computational Linguistics (in review)*.

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

Collins, M., Hajič, J., Ramshaw, L., and Tillmann, C. (1999). A statistical parser for Czech. In *Proceedings of ACL*, pages 505–518.

Corazza, A. and Satta, G. (2006). Cross-entropy and estimation of probabilistic context-free grammars. In *Proceedings of NAACL*.

Curtiss, S. (1977). *Genie: A Psycholinguistic Study of Modern-day Wild Child*. 1977.

Dagan, I. (1991). Two languages are more informative than one. In *Proceedings of ACL*.

Das, D., Schneider, N., Chen, D., and Smith, N. A. (2010). Probabilistic frame-semantic parsing. In *Proceedings of ACL*.

Dasgupta, S. (1997). The sample complexity of learning fixed-structure bayesian networks. *Machine Learning*, 29(2-3):165–180.

Day, W. H. E. (1983). Computationally difficult parsimony problems in phylogenetic systematics. *Journal of Theoretical Biology*, 103.

de Marcken, C. (1995). On the unsupervised induction of phrase structure grammars. In *Proceedings of SIGDAT*.

Dean, J. and Ghemawat, S. (2004). MapReduce: Simplified data processing on large clusters. In *Proceedings of OSDI*.

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

DeNero, J. and Klein, D. (2008). The complexity of phrase alignment problems. In *Proceedings of ACL*.

Ding, Y. and Palmer, M. (2005). Machine translation using probabilistic synchronous dependency insertion grammars. In *Proceedings of ACL*.

Dyer, C. (2010). *A Formal Model of Ambiguity and its Applications in Machine Translation*. PhD thesis, University of Maryland.

Džeroski, S., Erjavec, T., Ledinek, N., Pajas, P., Žabokrtsky, Z., and Žele, A. (2006). Towards a Slovene dependency treebank. In *Proceedings of LREC*.

Eisner, J. (1997). Bilexical grammars and a cubic-time probabilistic parser. In *Proceedings of IWPT*.

Elsner, M., Charniak, E., and Johnson, M. (2009). Structured generative models for unsupervised named-entity clustering. In *Proceedings of HLT-NAACL*.

Finkel, J. R., Grenager, T., and Manning, C. D. (2007). The infinite tree. In *Proceedings of ACL*.

Gaifman, H. (1965). Dependency systems and phrase-structure systems. *Information and Control*, 8.

Ganchev, K. (2010). *Posterior Regularization for Learning with Side Information and Weak Supervision*. PhD thesis, University of Pennsylvania.

Ganchev, K., Gillenwater, J., and Taskar, B. (2009). Dependency grammar induction via bitext projection constraints. In *Proceedings of ACL*.

Gildea, D. (2010). Optimal parsing strategies for linear context-free rewriting systems. In *Proceedings of NAACL*.

Gillenwater, J., Ganchev, K., Graça, J., Pereira, F., and Taskar, B. (2010). Sparsity in dependency grammar induction. In *Proceedings of ACL*.

Gimpel, K. and Smith, N. A. (2011). Concavity and initialization for unsupervised dependency grammar induction. Technical report, Carnegie Mellon University.

Gold, E. M. (1967). Language identification in the limit. *Information and Control*, 10(5):447–474.

Goldman, S. A. and Mathias, H. D. (1996). Teaching a smarter learner. *Journal of Computer and System Sciences*, 52(2):255–267.

Goldwater, S. and Griffiths, T. L. (2007). A fully Bayesian approach to unsupervised part-of-speech tagging. In *Proceedings of ACL*.

Goldwater, S. and Johnson, M. (2005). Bias in learning syllable structure. In *Proceedings of CoNLL*.

Gómez-Rodríguez, C. and Satta, G. (2009). An optimal-time binarization algorithm for linear context-free rewriting systems with fan-out two. In *Proceedings of ACL-IJCNLP*.

Goodman, J. (1996). Parsing algorithms and metrics. In *Proceedings of ACL*.

Goodman, J. (1998). *Parsing Inside-out*. PhD thesis, Harvard University.

Grenander, U. (1981). *Abstract Inference*. Wiley, New York.

Guerra, G. and Aloimonos, Y. (2005). Discovering a language for human activity. In *AAAI Workshop on Anticipation in Cognitive Systems*.

Haghighi, A., Liang, P., Berg-Kirkpatrick, T., and Klein, D. (2008). Learning bilingual lexicons from monolingual corpora. In *Proceedings of ACL*.

Hajič, J., Böhmová, A., Hajičová, E., and Hldaká, B. V. (2000). The Prague dependency treebank: A three-level annotation scenario. *Treebanks: Building and Using Parsed Corpora*, pages 103–127.

Hardisty, E., Boyd-Graber, J., and Resnik, P. (2010). Modeling perspective using adaptor grammars. In *Proceedings of EMNLP*.

Harris, Z. S. (1951). *Methods in Structural Linguistics*. University of Chicago Press.

Haussler, D. (1992). Decision-theoretic generalizations of the PAC model for neural net and other learning applications. *Information and Computation*, 100:78–150.

Headden, W. P., Johnson, M., and McClosky, D. (2009). Improving unsupervised dependency parsing with richer contexts and smoothing. In *Proceedings of NAACL-HLT*.

Herbert, R. (1956). An empirical bayes approach to statistics. In *Proceedings of the Third Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Contributions to the Theory of Statistics*.

Hinton, G. E. (1999). Products of experts. In *Proceedings of ICANN*.

Hockenmaier, J. and Steedman, M. (2002). Generative models for statistical parsing with combinatory categorial grammar. In *Proceedings of ACL*.

Horning, J. J. (1969). *A Study of Grammatical Inference*. PhD thesis, Stanford University.

Hwa, R., Resnik, P., Weinberg, A., Cabezas, C., and Kolak, O. (2005). Bootstrapping parsers via syntactic projection across parallel texts. *Journal of Natural Language Engineering*, 11(3):311–25.

Ishigami, Y. and Tani, S. (1993). The VC-dimensions of finite automata with n states. In *Proceedings of ALT*.

Ishigami, Y. and Tani, S. (1997). VC-dimensions of finite automata and commutative finite automata with k letters and n states. *Applied Mathematics*, 74(3):229–240.

Johansson, R. and Nugues, P. (2007). LTH: Semantic structure extraction using nonprojective dependency trees. In *Proceedings of SemEval*.

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

Johnson, M. (2007). Why doesn't EM find good HMM POS-taggers? In *Proceedings EMNLP-CoNLL*.

Johnson, M. (2008a). Unsupervised word segmentation for Sesotho using adaptor grammars. In *Proceedings of the Tenth Meeting of ACL Special Interest Group on Computational Morphology and Phonology*.

Johnson, M. (2008b). Using adaptor grammars to identify synergies in the unsupervised acquisition of linguistic structure. In *Proceedings of ACL*.

Johnson, M. (2010). PCFGs, topic models, adaptor grammars and learning topical collocations and the structure of proper nouns. In *Proceedings of ACL*.

Johnson, M. and Goldwater, S. (2009). Improving nonparameteric Bayesian inference experiments on unsupervised word segmentation with adaptor grammars. In *Proceedings of HLT-NAACL*.

Johnson, M., Griffiths, T. L., and Goldwater, S. (2006). Adaptor grammars: A framework for specifying compositional nonparameteric Bayesian models. In *NIPS*.

Johnson, M., Griffiths, T. L., and Goldwater, S. (2007). Bayesian inference for PCFGs via Markov chain Monte Carlo. In *Proceedings of NAACL*.

Jordan, M. I., Ghahramani, Z., Jaakola, T. S., and Saul, L. K. (1999). An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233.

Joshi, A. (2003). Tree adjoining grammars. In Mitkov, R., editor, *The Oxford Handbook of Computational Linguistics*, pages 483–501. Oxford University Press.

Kaji, Y., Nakanisi, R., Seki, H., and Kasami, T. (1992). The universal recognition problems for multiple context-free grammars and for linear context-free rewriting systems. *IEICE Transactions on Information and Systems*, E75-D(1):78–88.

Kawata, Y. and Bartels, J. (2000). Stylebook for the Japanese treebank in VERBMOBIL. Technical Report Verbmobil-Report 240, Seminar für Sprachwissenschaft, Univerisität Tübingen.

Kearns, M. and Valiant, L. (1989). Cryptographic limitations on learning boolean formulae and finite automata. In *Proceedings of the 21st ACM Symposium on the Theory of Computing*, pages 433–444.

Klein, D. and Manning, C. (2001). Natural language grammar induction using a constituent-context model. In *Advances in NIPS*.

Klein, D. and Manning, C. D. (2002). A generative constituent-context model for improved grammar induction. In *Proceedings of ACL*.

Klein, D. and Manning, C. D. (2003a). A* parsing: Fast exact Viterbi parse selection. In *Proceedings of HLT-NAACL*, pages 119–126.

Klein, D. and Manning, C. D. (2003b). Accurate unlexicalized parsing. In *Proceedings of ACL*, pages 423–430.

Klein, D. and Manning, C. D. (2004). Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of ACL*.

Knight, K. (1999). Decoding complexity in word-replacement translation models. *Computational Linguistics*, 25(4):607–615.

Koltchinskii, V. (2006). Local Rademacher complexities and oracle inequalities in risk minimization. *The Annals of Statistics*, 34(6):2593–2656.

Kroeger, P. (2005). *Analyzing Grammar: An Introduction*. Cambridge University Press.

Kromann, M., Mikkelsen, L., and Lynge, S. (2003). Danish Dependency Treebank. In *Proceedings of the Workshop on Treebanks and Linguistic Theories*.

Kurihara, K. and Sato, T. (2006). Variational Bayesian grammar induction for natural language. In *Proceedings of ICGI*.

Kurihara, K., Welling, M., and Vlassis, N. A. (2006). Accelerated variational Dirichlet process mixtures. In *NIPS*.

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

Leermakers, R. (1989). How to cover a grammar. In *Proceedings of ACL*.

Liang, P., Petrov, S., Jordan, M., and Klein, D. (2007). The infinite PCFG using hierarchical Dirichlet processes. In *Proceedings of EMNLP*.

Lin, L., Wu, T., Porway, J., and Xu, Z. (2009). A stochastic graph grammar for compositional object representation and recognition. *Pattern Recognition*, 8.

Lloyd, S. P. (1982). Least squares quantization in PCM. In *IEEE Transactions on Information Theory*.

Lyngsø, R. B. and Pederson, C. N. S. (2002). The consensus string problem and the complexity of comparing hidden Markov models. *Journal of Computing and System Science*, 65(3):545–569.

Mahajan, M., Nimbhorkar, P., and Varadarajan, K. (2009). The planar k-means problem is NP-hard. In *Proceedings of International Workshop on Algorithms and Computation*.

Manning, C. and Schütze, H. (1999). *Foundations of Statistical Natural Language Processing*. MIT Press.

Manning, C. D. (2003). Probabilistic syntax. *Probabilistic Linguistics*.

Marcus, M. P., Santorini, B., and Marcinkiewicz, M. A. (1993). Building a large annotated corpus of English: The Penn treebank. *Computational Linguistics*, 19:313–330.

Massart, P. (2000). Some applications of concentration inequalities to statistics. *Ann. Fac. Sci. Toulouse (IX)*, pages 245–303.

McAllester, D. (2003). PAC-Bayesian model averaging. *Machine Learning Journal*, 5:5–21.

McCallum, A., Mann, G., and Druck, G. (2007). Generalized expectation criteria. Technical Report #2007-60, University of Massachusetts Amherst.

McClosky, D., Charniak, E., and Johnson, M. (2006a). Effective self-training for parsing. In *Proceedings of HLT-NAACL*.

McClosky, D., Charniak, E., and Johnson, M. (2006b). Reranking and self-training for parser adaptation. In *Proceedings of COLING-ACL*.

McDonald, R. and Satta, G. (2007). On the complexity of non-projective data-driven dependency parsing. In *Proceedings of IWPT*.

Mimno, D., Wallach, H., and McCallum, A. (2008). Gibbs sampling for logistic normal topic models with graph-based priors. In *In Proceedings of NIPS Workshop on Analyzing Graphs*.

Mochihashi, D., Yamada, T., and Ueda, N. (2009). Bayesian unsupervised word segmentation with nested pitman-yor language modeling. In *Proceedings of ACL*.

Mohri, M. and Roark, B. (2006). Probabilistic context-free grammar induction based on structural zeros. In *Proceedings of HLT-NAACL*.

Naseem, T. and Barzilay, R. (2011). Using semantic cues to learn syntax. In *Proceedings of AAAI*.

Naseem, T., Chen, H., Barzilay, R., and Johnson, M. (2010). Using universal linguistic knowledge to guide grammar induction. In *Proceedings of EMNLP*.

Neal, R. M. and Hinton, G. E. (1998). A view of the EM algorithm that justifies incremental, sparse, and other variants. In *Learning and Graphical Models*, pages 355–368. Kluwer Academic Publishers.

Newman, D., Asuncion, A., Smyth, P., and Welling, M. (2009). Distributed algorithms for topic models. *Journal of Machine Learning Research*, 10:1801–1828.

Nijholt, A. (1980). *Context-Free Grammars: Covers, Normal Forms, and Parsing*, volume 93. Springer-Verlag, Berlin, Germany.

Nivre, J., Nilsson, J., and Hall, J. (2006). Talbanken05: A Swedish Treebank with Phrase Structure and Dependency Annotation. In *Proceedings of LREC*.

O'Donnell, T., Goodman, N. D., and Tenenbaum, J. B. (2009). Fragment grammars: Exploring computation and reuse in language. Technical Report MIT-CSAIL-TR-2009-13, Massachusetts Institute of Technology.

Oflazer, K., Say, B., Hakkani-Tür, D. Z., and Tür, G. (2003). Building a Turkish treebank. In Abeille, A., editor, *Building and Exploiting Syntactically-Annotated Corpora*. Kluwer. Available at `http://www.ii.metu.edu.tr/~corpus/treebank.html`.

Palmer, N. and Goldberg, P. W. (2007). Pac-learnability of probabilistic deterministic finite state automata in terms of variation distance. In *Proceedings of ALT*.

Pereira, F. C. N. and Schabes, Y. (1992). Inside-outside reestimation from partially bracketed corpora. In *Proceedings of ACL*, pages 128–135.

Pitman, J. (2002). *Combinatorial Stochastic Processes*. Lecture Notes for St. Flour Summer School. Springer-Verlag, New York, NY.

Pitman, J. and Yor, M. (1997). The two-parameter Poisson-Dirichlet distribution derived from a stable subordinator. *Annals of Probability*, 25(2):855–900.

Pollard, D. (1984). *Convergence of Stochastic Processes*. New York: Springer-Verlag.

Ponvert, E., Baldridge, J., and Erk, K. (2011). Simple unsupervised grammar induction from raw text with cascaded finite state models. In *Proceedings of ACL*.

Post, M. and Gildea, D. (2009). Bayesian learning of a tree substitution grammar. In *Proceedings of ACL*.

Prokopidis, P., Desypri, E., Koutsombogera, M., Papageorgiou, H., and Piperidis, S. (2005). Theoretical and practical issues in the construction of a Greek dependency treebank. In *Proceedings of the Workshop on Treebanks and Linguistic Theories*.

Raiffa, H. and Schlaifer, R. (1961). *Applied Statistical Decision Theory*. Wiley-Interscience.

Rambow, O. and Satta, G. (1994). A rewriting system for free word order syntax that is non-local and mildly context sensitive. *In C. Martin-Vide ed., Current Issues in Mathematical Linguistics*, 65:121–130.

Robert, C. P. and Casella, G. (2005). *Monte Carlo Statistical Methods*. Springer.

Ron, D. (1995). *Automata Learning and Its Applications*. PhD thesis, Hebrew University of Jerusalem.

Ron, D., Singer, Y., and Tishby, N. (1995). On the learnability and usage of acyclic probabilistic finite automata. In *Proceedings of COLT*.

Sakakibara, Y., Brown, M., Hughey, R., Mian, S., Sjölander, K., Underwood, R. C., and Haussler, D. (1994). Stochastic context-free grammars for tRNA modeling. *Nucleic Acids Research*, 22.

Satta, G. (1992). Recognition of linear context-free rewriting systems. In *Proceedings of ACL*.

Schwartz, R., Abend, O., Reichart, R., and Rappoport, A. (2011). Neutralizing linguistically problematic annotations in unsupervised dependency parsing evaluation. In *Proceedings of ACL*.

Seeger, M. (2002). Pac-bayesian generalization bounds for gaussian processes. *Journal of Machine Learning Research*, 3:233–269.

Seginer, Y. (2007). Fast unsupervised incremental parsing. In *Proceedings of ACL*.

Sethuraman, J. (1994). A constructive definition of Dirichlet priors. *Statistica Sinica*, 4:639–650.

Shalev-Shwartz, S., Shamir, O., Sridharan, K., and Srebro, N. (2009). Learnability and stability in the general learning setting. In *Proceedings of COLT*.

Sima'an, K. (1996). Computational complexity of probabilistic disambiguation by means of tree-grammars. In *In Proceedings of COLING*.

Simov, K., Osenova, P., Kolkovska, S., Balabanova, E., Doikoff, D., Ivanova, K., Simov, A., and Kouylekov, M. (2002). Building a Linguistically Interpreted Corpus of Bulgarian: the BulTreeBank. In *Proceedings of LREC*.

Sipser, M. (2006). *Introduction to the Theory of Computation, Second Edition*. Thomson Course Technology.

Smith, D. A. and Smith, N. A. (2004). Bilingual parsing with factored estimation: Using English to parse Korean. In *Proceedings of EMNLP*, pages 49–56.

Smith, N. A. (2006). *Novel Estimation Methods for Unsupervised Discovery of Latent Structure in Natural Language Text*. PhD thesis, Johns Hopkins University.

Smith, N. A. and Eisner, J. (2005). Contrastive estimation: Training log-linear models on unlabeled data. In *Proceedings of ACL*, pages 354–362.

Smith, N. A. and Eisner, J. (2006). Annealing structural bias in multilingual weighted grammar induction. In *Proceedings of COLING-ACL*.

Snyder, B. and Barzilay, R. (2008). Unsupervised multilingual learning for morphological segmentation. In *Proceedings of ACL*.

Solomonoff, R. (1958). The mechanization of linguistic learning. In *Second International Congress on Cybernetics*.

Spitkovsky, V., Alshawi, H., and Jurafsky, D. (2010a). From baby steps to leapfrog: How "less is more" in unsupervised dependency parsing. In *Proceedings of NAACL*.

Spitkovsky, V. I., Alshawi, H., and Jurafsky, D. (2011a). Lateen EM: Unsupervised training with multiple objectives, applied to dependency grammar induction. In *Proceedings of EMNLP*.

Spitkovsky, V. I., Alshawi, H., and Jurafsky, D. (2011b). Punctuation: Making a point in unsupervised dependency parsing. In *Proceedings of CoNLL*.

Spitkovsky, V. I., Alshawi, H., Jurafsky, D., and Manning, C. D. (2010b). Viterbi training improves unsupervised dependency parsing. In *Proceedings of CoNLL*.

Spitkovsky, V. I., Jurafsky, D., and Alshawi, H. (2010c). Profiting from mark-up: Hyper-text annotations for guided parsing. In *Proceedings of ACL*.

Starkie, B., Coste, F., and van Zaanen, M. (2004). The Omphalos context-free grammar learning competition. In *International Colloquium on Grammatical Inference*, volume 3264, pages 16–27.

Stolcke, A. and Omohundro, S. M. (1994). Inducing probabilistic grammars by Bayesian model merging. In *Proceedings of ICGI*.

Talagrand, M. (1994). Sharper bounds for Gaussian and empirical processes. *Annals of Probability*, 22:28–76.

Tesnière, L. (1959). *Élément de Syntaxe Structurale*. Klincksieck.

Toutanova, K. and Johnson, M. (2007). A Bayesian LDA-based model for semi-supervised part-of-speech tagging. In *Proceedings of NIPS*.

Tsybakov, A. (2004). Optimal aggregation of classifiers in statistical learning. *The Annals of Statistics*, 32(1):135–166.

Udupa, R. and Maji, K. (2006). Computational complexity of statistical machine translation. In *Proceedings of EACL*.

Van der Beek, L., Bouma, G., Malouf, R., and Van Noord, G. (2002). The Alpino dependency treebank. *Language and Computers*, 45(1):8–22.

van Zaanen, M. (2000). ABL: Alignment-based learning. In *Proceedings of COLING*.

Vapnik, V. N. (1998). *Statistical Learning Theory*. Wiley-Interscience.

Wainwright, M. J. and Jordan, M. I. (2008). Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1:1–305.

Wang, C. and Blei, D. M. (2009). Variational inference for the nested Chinese restaurant process. In *NIPS*.

Wang, M., Smith, N. A., and Mitamura, T. (2007). What is the Jeopardy model? a quasi-synchronous grammar for question answering. In *Proceedings of EMNLP*.

Wu, D. (1997). Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–404.

Xue, N., Xia, F., Chiou, F.-D., and Palmer, M. (2004). The Penn Chinese Treebank: Phrase structure annotation of a large corpus. *Natural Language Engineering*, 10(4):1–30.

Yangarber, R., Grishman, R., Tapanainen, P., and Huttunen, S. (2000). Automatic acquisition of domain knowledge for information extraction. In *Proceedings of COLING*.

Yarowsky, D., Ngai, G., and Wicentoswki, R. (2001). Inducing multilingual text analysis tools via robust projection across aligned corpora. In *Proceedings of HLT*.

Yoshinaka, R. and Clark, A. (2010). Polynomial time learning of some multiple context-free languages with a minimally adequate teacher. In *Proceedings of the 15th Conference on Formal Grammar*, Copenhagen, Denmark.

Part III

Appendices

Appendix A

Proofs for Chapter 4

We include in this appendix proofs for several results in the paper.

Utility Lemma 4.2 *Let* $a_i \in [0,1]$, $i \in \{1,\ldots,N\}$ *such that* $\sum_i a_i = 1$. *Define* $b_1 = a_1$, $c_1 = 1 - a_1$, $b_i = \left(\frac{a_i}{a_{i-1}}\right)\left(\frac{b_{i-1}}{c_{i-1}}\right)$ *and* $c_i = 1 - b_i$ *for* $i \geq 2$. *Then* $a_i = \left(\prod_{j=1}^{i-1} c_j\right) b_i$.

Proof Proof by induction on $i \in \{1,\ldots,N\}$. Clearly, the statement holds for $i = 1$. Assume it holds for arbitrary $i < N$. Then:

$$a_{i+1} = \left(\frac{a_i}{a_i}\right) a_{i+1} = \left(\left(\prod_{j=1}^{i-1} c_j\right) b_i\right) \frac{a_{i+1}}{a_i} = \left(\left(\prod_{j=1}^{i-1} c_j\right) b_i\right) \frac{c_i b_{i+1}}{b_i}$$
$$= \left(\prod_{j=1}^{i} c_j\right) b_{i+1}$$

and this completes the proof. □

Lemma 4.6 *Denote by* $\mathcal{Z}_{\epsilon,n}$ *the set* $\bigcup_{f\in\mathcal{F}}\{\mathbf{y} \mid C_n(f)(\mathbf{y}) - f(\mathbf{y}) \geq \epsilon\}$. *Denote by* $A_{\epsilon,n}$ *the event "one of* $y_i \in D$ *is in* $\mathcal{Z}_{\epsilon,n}$." *Then if* $\mathcal{F}_n$ *properly approximates* $\mathcal{F}$ *then:*

$$\mathbb{E}\left[\mathbb{E}_{\tilde{p}_n}[g_n] - \mathbb{E}_{\tilde{p}_n}[f_n^*]\right] \tag{A.0}$$
$$\leq \left|\mathbb{E}\left[\mathbb{E}_{\tilde{p}_n}[C_n(f_n^*)] \mid A_{\epsilon,n}\right]\right| p(A_{\epsilon,n}) + \left|\mathbb{E}\left[\mathbb{E}_{\tilde{p}_n}[f_n^*] \mid A_{\epsilon,n}\right]\right| p(A_{\epsilon,n}) + \epsilon_{\text{tail}}(n)$$

where the expectations are taken with respect to the dataset D.

Proof Consider the following:

$$\begin{aligned}
&\mathbb{E}[\mathbb{E}_{\tilde{p}_n}[g_n] - \mathbb{E}_{\tilde{p}_n}[f_n^*]] \\
&= \mathbb{E}[\mathbb{E}_{\tilde{p}_n}[g_n]] - \mathbb{E}_{\tilde{p}_n}[C_n(f_n^*)] + \mathbb{E}_{\tilde{p}_n}[C_n(f_n^*)] - \mathbb{E}_{\tilde{p}_n}[f_n^*]] \\
&= \mathbb{E}[\mathbb{E}_{\tilde{p}_n}[g_n] - \mathbb{E}_{\tilde{p}_n}[C_n(f_n^*)]] + \mathbb{E}[\mathbb{E}_{\tilde{p}_n}[C_n(f_n^*)] - \mathbb{E}_{\tilde{p}_n}[f_n^*]]
\end{aligned}$$

Note first that $\mathbb{E}[\mathbb{E}_{\tilde{p}_n}[g_n] - \mathbb{E}_{\tilde{p}_n}[C_n(f_n^*)]] \leq 0$, by the definition of g_n as the minimizer of the empirical risk. We next bound $\mathbb{E}[\mathbb{E}_{\tilde{p}_n}[C_n(f_n^*)] - \mathbb{E}_{\tilde{p}_n}[f_n^*]]$. We know from the requirement of proper approximation that we have:

$$\begin{aligned}
&\mathbb{E}[\mathbb{E}_{\tilde{p}_n}[C_n(f_n^*)] - \mathbb{E}_{\tilde{p}_n}[f_n^*]] \\
&= \mathbb{E}[\mathbb{E}_{\tilde{p}_n}[C_n(f_n^*)] - \mathbb{E}_{\tilde{p}_n}[f_n^*] \mid A_{\epsilon,n}]p(A_{\epsilon,n}) + \\
&\mathbb{E}[\mathbb{E}_{\tilde{p}_n}[C_n(f_n^*)] - \mathbb{E}_{\tilde{p}_n}[f_n^*] \mid \neg A_{\epsilon,n}](1 - p(A_{\epsilon,n})) \\
&\leq |\mathbb{E}[\mathbb{E}_{\tilde{p}_n}[C_n(f_n^*)] \mid A_{\epsilon,n}]|p(A_{\epsilon,n}) + |\mathbb{E}[\mathbb{E}_{\tilde{p}_n}[f_n^*] \mid A_{\epsilon,n}]|p(A_{\epsilon,n}) + \epsilon_{\text{tail}}(n)
\end{aligned}$$

and that equals the right side of Equation A. □

Proposition 4.4 *Let $p \in \mathcal{P}(\alpha, L, r, q, B, \mathbf{G})$ and let $\mathcal{F}_m$ as defined above. There exists a constant $\beta = \beta(L, q, s, N) > 0$ such that $\mathcal{F}_m$ has the boundedness property with $K_m = sN \log^3 m$ and $\epsilon_{\text{bound}}(m) = m^{-\beta \log m}$.*

Proof Let $f \in \mathcal{F}_m$. Let $\mathcal{Z}(m) = \{\mathbf{y} \mid |\mathbf{y}| \leq \log^2 m\}$. Then, for all $\mathbf{y} \in \mathcal{Z}(m)$ we have $|f(\mathbf{y})| = -\sum_{i,k} \psi_{k,i}(\mathbf{y}) \log \theta_{k,i} \leq \sum_{i,k} \psi_{k,i}(\mathbf{y})(s \log m) \leq sN \log^3 m = K_m$, where the first inequality follows from $f \in \mathcal{F}_m$ ($\theta_{k,i} \geq m^{-s}$) and the second from $|\mathbf{y}| \leq \log^2 m$. In addition, from the requirements on p we have:

$$\mathbb{E}\left[|f| \times \mathbb{I}\{|f| \geq K_m\}\right] \leq \left(sN \log^3 m\right) \times \left(\sum_{k > \log^2 m} L\Lambda(k) r^k k\right) \leq \left(\kappa \log^3 m\right) \times \left(q^{\log^2 m}\right)$$

for $\kappa = \dfrac{sNL}{(1-q)^2}$. Finally, for $\beta(L, q, s, N) \triangleq \log \kappa + 1 + \log \frac{1}{q} = \beta > 0$ and if $m > 1$ then $\left(\kappa \log^3 m\right)\left(q^{\log^2 m}\right) \leq m^{-\beta \log m}$. □

Utility Lemma A.1 *(From (Dasgupta, 1997).) Let $a \in [0, 1]$ and let $b = a$ if $a \in [\gamma, 1 - \gamma]$, $b = \gamma$ if $a \leq \gamma$ and $b = 1 - \gamma$ if $a \geq 1 - \gamma$. Then for any $\epsilon \leq 1/2$ such that $\gamma \leq \epsilon/(1 + \epsilon)$ we have $\log a/b \leq \epsilon$.*

Proposition 4.5 *Let $p \in \mathcal{P}(\alpha, L, r, q, B, \mathbf{G})$ and let $\mathcal{F}_m$ as defined above. There exists an M such that for any $m > M$ we have:*

$$p\left(\bigcup_{f\in\mathcal{F}}\{\mathbf{y} \mid C_m(f)(\mathbf{y}) - f(\mathbf{y}) \geq \epsilon_{\text{tail}}(m)\}\right) \leq \epsilon_{\text{tail}}(m)$$

for $\epsilon_{\text{tail}}(m) = \dfrac{N\log^2 m}{m^s - 1}$ *and* $C_m(f) = T(f, m^{-s})$.

Proof Let $\mathcal{Z}(m)$ be the set of derivations of size bigger than $\log^2 m$. Let $f \in \mathcal{F}$. Define $f' = T(f, m^{-s})$. For any $\mathbf{y} \notin \mathcal{Z}(m)$ we have that:

$$\begin{aligned} f'(\mathbf{y}) - f(\mathbf{y}) &= -\sum_{k=1}^{K}\left(\psi_{k,1}(\mathbf{y})\log\theta_{k,1} + \psi_{k,2}(\mathbf{y})\log\theta_{k,2} - \psi_{k,1}(\mathbf{y})\log\theta'_{k,1} - \psi_{k,1}(\mathbf{y})\log\theta'_{k,2}\right) \\ &\leq \sum_{k=1}^{K}\log^2 m\left(\max\{0, \log(\theta'_{k,1}/\theta_{k,1})\} + \max\{0, \log(\theta'_{k,2}/\theta_{k,2})\}\right) \end{aligned} \tag{A.1}$$

Without loss of generality, assume $\epsilon_{\text{tail}}(n)/N\log^2 m \leq 1/2$. Let $\gamma = \dfrac{\epsilon_{\text{tail}}(m)/N\log^2 m}{1 + \epsilon_{\text{tail}}(m)/N\log^2 m} = 1/m^s$. From Utility Lemma A.1 we have that $\log(\theta'_{k,i}/\theta_{k,i}) \leq \epsilon_{\text{tail}}(m)/N\log m$. Plug this into Equation A.1 ($N = 2K$) to get that for all $\mathbf{y} \notin \mathcal{Z}(m)$ we have $f'(\mathbf{y}) - f(\mathbf{y}) \leq \epsilon_{\text{tail}}(m)$. It remains to show that the measure $p(\mathcal{Z}(m)) \leq \epsilon_{\text{tail}}(m)$. Note that $\sum_{\mathbf{y}\in\mathcal{Z}(m)} p(\mathbf{y}) \leq \sum_{k>\log^2 m} L\Lambda(k)r^k \leq L\sum_{k>\log^2 m} q^k = Lq^{\log^2 m}/(1-q) < \epsilon_{\text{tail}}(m)$ for $m > M$ where M is a fixed constant chosen appropriately. □

Proposition A.2 *There exists a $\beta'(L, q, s, N) > 0$ such that $\mathcal{F}'_m$ has the boundedness property with $K_m = sN\log^3 m$ and $\epsilon_{\text{bound}}(m) = m^{-\beta'\log m}$.*

Proof From the requirement of p, we know that for any $\boldsymbol{x}$ we have a $\mathbf{y}$ such that $s(\mathbf{y}) = \boldsymbol{x}$ and $|\mathbf{y}| \leq \alpha|\boldsymbol{x}|$. Therefore, if we let $\mathcal{X}(m) = \{\boldsymbol{x} \mid |\boldsymbol{x}| \leq \log^2 m/\alpha\}$, then we have for any $f \in \mathcal{F}'_m$ and $x \in \mathcal{X}(m)$ that $f(\boldsymbol{x}) \leq sN\log^3 m = K_m$ (similarly to the proof of Proposition 4.1). Denote by $f_1(\boldsymbol{x}, \mathbf{y})$ the function in $\mathcal{F}_m$ such that $f(\boldsymbol{x}) = -\log\sum_{\mathbf{y}}\exp(-f_1(\boldsymbol{x}, \mathbf{y}))$.

In addition, from the requirements on p and the definition of K_m we have:

$$\begin{aligned}\mathbb{E}\left[|f| \times \mathbb{I}\left\{|f| \geq K_m\right\}\right] &= \sum_{\boldsymbol{x}} p(\boldsymbol{x}) f(\boldsymbol{x}) \mathbb{I}\left\{f \geq K_m\right\} \\ &= \sum_{x:|\boldsymbol{x}|>\log^2 m/\alpha} p(\boldsymbol{x}) f(\boldsymbol{x}) \\ &\leq \sum_{x:|\boldsymbol{x}|>\log^2 m/\alpha} p(\boldsymbol{x}) f_1(\boldsymbol{x}, \mathbf{y}(\boldsymbol{x}))\end{aligned}$$

where $\mathbf{y}(\boldsymbol{x})$ is some derivation for $\boldsymbol{x}$. We have:

$$\begin{aligned}\sum_{x:|\boldsymbol{x}|>\log^2 m/\alpha} p(\boldsymbol{x}) f_1(\boldsymbol{x}, \mathbf{y}(\boldsymbol{x})) &\leq \sum_{x:|\boldsymbol{x}|\geq\log^2 m/\alpha} \sum_{\mathbf{y}\in D_x(\mathbf{G})} p(\boldsymbol{x}, \mathbf{y}) f_1(x, \mathbf{y}(\boldsymbol{x})) \\ &\leq sN \log m \sum_{x:|\boldsymbol{x}|>\log^2 m/\alpha} \sum_{z} p(\boldsymbol{x}, \mathbf{y}) |\mathbf{y}(\boldsymbol{x})| \\ &\leq sN \log m \sum_{k>\log^2 m} \Lambda(k) r^k k \\ &\leq sN \log m \sum_{k>\log^2 m} q^k k \leq \kappa \times (\log m) \times (q^{log^2 m})\end{aligned}$$

for some constant $\kappa > 0$. Finally, for some $\beta'(L, q, s, N) = \beta' > 0$ and some constant M, if $m > M$ then $\kappa \log m \left(q^{\log^2 m}\right) \leq m^{-\beta' \log m}$. □

Utility Lemma 4.11 *For $a_i, b_i \geq 0$, if $-\log \sum_i a_i + \log \sum_i b_i \geq \epsilon$ then there exists an i such that $-\log a_i + \log b_i \geq \epsilon$.*

Proof Assume $-\log a_i + \log b_i < \epsilon$ for all i. Then, $b_i/a_i < e^{\epsilon}$, therefore $\sum_i b_i / \sum_i a_i < e^{\epsilon}$, therefore $-\log \sum_i a_i + \log \sum_i b_i < \epsilon$ which is a contradiction to $-\log \sum_i a_i + \log \sum_i b_i \geq \epsilon$. □

The next lemma is the main concentarion of measure result that we use. Its proof requires some simple modification to the proof given for Theorem 24 in (Pollard, 1984, pages 30–31).

Lemma 4.8 *Let $\mathcal{F}_n$ be a permissible class of functions such that for every $f \in \mathcal{F}_n$ we have $\mathbb{E}[|f| \times \mathbb{I}\left\{|f| \leq K_n\right\}] \leq \epsilon_{\text{bound}}(n)$. Let $\mathcal{F}_{\text{truncated},n} = \{f \times \mathbb{I}\left\{f \leq K_n\right\} \mid$*

$f \in \mathcal{F}_m\}$*, i.e., the set of functions from* $\mathcal{F}_n$ *after being truncated by* K_n*. Then for* $\epsilon > 0$ *we have,*

$$p\left(\sup_{f\in\mathcal{F}_n} |\mathbb{E}_{\tilde{p}_n}[f] - \mathbb{E}_p[f]| > 2\epsilon\right) \leq 8\mathcal{N}(\epsilon/8, \mathcal{F}_{\text{truncated},n}) \exp\left(-\frac{1}{128} n\epsilon^2/K_n^2\right) + \epsilon_{\text{bound}}(n)/\epsilon$$

provided $n \geq K_n^2/4\epsilon^2$ *and* $\epsilon_{\text{bound}}(n) < \epsilon$*.*

Proof First note that

$$\begin{aligned}\sup_{f\in\mathcal{F}_n} |\mathbb{E}_{\tilde{p}_n}[f] - \mathbb{E}_p[f]| \leq & \sup_{f\in\mathcal{F}_n} |\mathbb{E}_{\tilde{p}_n}[f\mathbb{I}\{|f| \leq K_n\}] - \mathbb{E}_p[f\mathbb{I}\{|f| \leq K_n\}]| \\ & + \sup_{f\in\mathcal{F}_n} \mathbb{E}_{\tilde{p}_n}[|f|(|f| \leq K_n)] + \sup_{f\in\mathcal{F}_n} \mathbb{E}_p[|f|(|f| \leq K_n)]\end{aligned}$$

We have $\sup_{f\in\mathcal{F}_n} \mathbb{E}_p[|f|(|f| \leq K_n)] \leq \epsilon_{\text{bound}}(n) < \epsilon$ and also, from Markov inequality, we have:

$$P(\sup_{f\in\mathcal{F}_n} \mathbb{E}_{\tilde{p}_n}[|f|(|f| \leq K_n)] > \epsilon) \leq \epsilon_{\text{bound}}(n)/\epsilon$$

At this point, we can follow the proof of Theorem 24 in Pollard (1984), and its extension in pages 30–31 to get Lemma 5.1, using the shifted set of functions $\mathcal{F}_{\text{truncated},n}$. □

Appendix B

Minimizing Log-Loss for Probabilistic Grammars

Central to our algorithms for minimizing the log-loss from Chapter 4 (both in the supervised case and the unsupervised case) is a convex optimization problem of the form:

$$\begin{aligned}
&\min_{\boldsymbol{\theta}} \sum_{k=1}^{K} c_{k,1} \log \theta_{k,1} + c_{k,2} \log \theta_{k,2} \\
&\text{s.t.} \forall k \in \{1, \ldots, K\} : \\
&\quad \theta_{k,1} + \theta_{k,2} = 1 \\
&\quad \gamma \leq \theta_{k,1} \leq 1 - \gamma \\
&\quad \gamma \leq \theta_{k,2} \leq 1 - \gamma
\end{aligned}$$

for constants $c_{k,i}$ which depend on $\tilde{p}_n$ or some other intermediate distribution in the case of the expectation-maximization algorithm and γ which is a margin determined by the number of samples. This minimization problem can be decomposed into several optimization problems, one for each k, each having the following form:

$$\max_{\boldsymbol{\beta}} c_1\beta_1 + c_2\beta_2 \tag{B.1}$$

$$\text{s.t.} \exp(\beta_1) + \exp(\beta_2) = 1 \tag{B.2}$$

$$\gamma \leq \beta_1 \leq 1 - \gamma \tag{B.3}$$

$$\gamma \leq \beta_2 \leq 1 - \gamma \tag{B.4}$$

where $c_i \geq 0$ and $1/2 > \gamma \geq 0$. Ignore for a moment the constraints $\gamma \leq \beta_i \leq 1 - \gamma$. In that case, this can be thought of as a regular maximum likelihood estimation problem, so $\beta_i = c_i/(c_1 + c_2)$. We give a derivation of this result in this simple case for completion. We use Lagranian multipliers to solve this problem. Let $F(\beta 1, \beta 2) = c_1\beta_1 + c_2\beta_2$. Define the Lagrangian:

$$\begin{aligned} g(\lambda) &= \inf_{\boldsymbol{\beta}} L(\lambda, \boldsymbol{\beta}) \\ &= \inf_{\boldsymbol{\beta}} c_1\beta_1 + c_2\beta_2 + \lambda(\exp(\beta_1) + \exp(\beta_2) - 1) \end{aligned}$$

Taking the derivative of the term we minimize in the Lagrangian, we have:

$$\frac{\partial L}{\partial \beta_i} = c_i + \lambda \exp(\beta_i)$$

Setting the derivatives to 0 for minimization, we have:

$$g(\lambda) = c_1 \log(-c_1/\lambda) + c_2 \log(-c_2/\lambda) + \lambda(-c_1/\lambda - c_2/\lambda - 1) \quad \text{(B.5)}$$

$g(\lambda)$ is the objective function of the dual problem of Equation B.1–Equation B.2. We would like to minimize Equation B.5 with respect to λ. The derivative of $g(\lambda)$ is:

$$\frac{\partial g}{\partial \lambda} = -c_1/\lambda - c_2/\lambda - 1$$

hence when equating the derivative of $g(\lambda)$ to 0, we get $\lambda = -(c_1 + c_2)$, and therefore the solution is $\beta_i^* = \log(c_i/(c_1 + c_2))$. We need to verify that the solution to the dual problem indeed gets the optimal value for the primal. Since the primal problem is convex, it is sufficient to verify that the Karush-Kuhn-Tucker conditions hold (Boyd and Vandenberghe, 2004). Indeed, we have:

$$\begin{aligned} \frac{\partial F}{\partial \beta_i}(\boldsymbol{\beta}^*) + \lambda \frac{\partial h}{\partial \beta_i}(\boldsymbol{\beta}^*) &= c_i - (c_1 + c_2) \times \frac{c_i}{c_1 + c_2} \\ &= 0 \end{aligned}$$

where $h(\boldsymbol{\beta}) \triangleq \exp(\beta) + \exp(\beta) - 1$ stands for the equality constraint. The rest of the KKT conditions trivially hold, therefore $\boldsymbol{\beta}^*$ is the optimal solution for Equations B.1-B.2.

Note that if $1-\gamma < c_i/(c_1+c_2) < \gamma$, then this is the solution even when again adding the constraints in Equation B.3 and Equation B.4. When $c_1/(c_1+c_2) < \gamma$, then the solution is $\beta_1^* = \gamma$ and $\beta_2^* = 1-\gamma$. Similarly, when $c_2/(c_1+c_2) < \gamma$ then the solution is $\beta_2^* = \gamma$ and $\beta_1^* = 1-\gamma$. We describe why this is true for the first case. The second case follows very similarly. Assume $c_1/(c_1+c_2) < \gamma$. We want to show that for any choice of $\beta \in [0,1]$ such that $\beta > \gamma$ we have:

$$c_1 \log \gamma + c_2 \log(1-\gamma) \geq c_1 \log \beta + c_2 \log(1-\beta)$$

Divide both sides of the inequality by $c_1 + c_2$ and we get that we need to show that

$$\frac{c_1}{c_1+c_2} \log(\gamma/\beta) + \frac{c_2}{c_1+c_2} \log\left(\frac{1-\gamma}{1-\beta}\right) \geq 0$$

Since we have $\beta > \gamma$, and we also have $c_1/(c_1+c_2) < \gamma$, it is sufficient to show that

$$\gamma \log(\gamma/\beta) + (1-\gamma) \log\left(\frac{1-\gamma}{1-\beta}\right) \geq 0 \tag{B.6}$$

Equation B.6 is precisely the definition of the KL-divergence between the distribution of a coin with probability γ of heads and the distribution of a coin with probability β of heads, and therefore the right side in Equation B.6 is positive, and we get what we need.

Appendix C

Counterexample to Tsybakov Noise (Proofs)

We turn now to give proofs for the Tsybakov noise result in Chapter 4.

Lemma C.1 *$A = A_{\mathbf{G}}(\boldsymbol{\theta})$ is positive semi-definite for any probabilistic grammar $\langle \mathbf{G}, \boldsymbol{\theta} \rangle$.*

Proof Let $d_{k,i}$ be a collection of constants. Define the random variable:

$$R(\mathbf{y}) = \sum_{i,k} \frac{d_{k,i}}{\mathbb{E}[\psi_{k,i}]} \psi_{k,i}(\mathbf{y})$$

We have that:

$$\mathbb{E}[R^2] = \sum_{i,i'} \sum_{k,k'} A_{(k,i),(k',i')} d_{k,i} d_{k',i'}$$

which is always larger or equal to 0. Therefore, A is positive semi-definite. □

Lemma C.2 *Let $0 < \mu < 1/2$, $c_1, c_2 \geq 0$. Let $\kappa, C > 0$. Also, assume that $c_1 \leq c_2$. For any $\epsilon > 0$, define:*

$$a = \mu \left(\exp\left(\frac{C\epsilon^{1/\kappa} + \epsilon/2}{c_1} \right) \right) = \alpha_1 \mu$$

$$b = \mu \left(\exp\left(\frac{-C\epsilon^{1/\kappa} + \epsilon/2}{c_2} \right) \right) = \alpha_2 \mu$$

$$t(\epsilon) = c_1 \left(\frac{1-\mu}{1-a} \right) + c_2 \left(\frac{1-\mu}{1-b} \right) - (c_1 + c_2) \exp(\epsilon/2)$$

Then, for small enough ϵ, we have $t(\epsilon) \leq 0$.

Proof We have that $t(\epsilon) \leq 0$ if:

$$\begin{aligned} ac_2 + bc_1 &\geq -\frac{(c_1 + c_2)(1-a)(1-b)}{1-\mu} \exp(\epsilon/2) + c_1 + c_2 \\ &= (c_1 + c_2)\left(1 - \frac{(1-a)(1-b)}{(1-\mu)\exp(-\epsilon/2)}\right) \end{aligned} \tag{C.1}$$

First, show that:

$$\frac{(1-a)(1-b)}{(1-\mu)\exp(-\epsilon/2)} \geq 1 - \mu \tag{C.2}$$

which happens if (after substituting $a = \alpha_1\mu$, $b = \alpha_2\mu$):

$$\mu \leq (\alpha_1 + \alpha_2 - 2)/(1 - \alpha_1\alpha_2)$$

Note we have $\alpha_1\alpha_2 > 1$ because $c_1 \leq c_2$. In addition, we have $\alpha_1 + \alpha_2 - 2 \geq 0$ for small enough ϵ (can be shown by taking the derivative, with respect to ϵ of $\alpha_1 + \alpha_2 - 2$, which is always positive for small enough ϵ, and in addition, noticing that the value of $\alpha_1 + \alpha_2 - 2$ is 0 when $\epsilon = 0$.) Therefore, Equation C.2 is true.

Substituting Equation C.2 in Equation C.1, we have that $t(\epsilon) \leq 0$ if:

$$ac_2 + bc_1 \geq (c_1 + c_2)\mu$$

which is equivalent to:

$$c_2\alpha_1 + c_1\alpha_2 \geq c_1 + c_2 \tag{C.3}$$

Taking again the derivative of the left side of Equation C.3, we have that it is an increasing function of ϵ (if $c_1 \leq c_2$), and in addition at $\epsilon = 0$ it obtains the value $c_1 + c_2$. Therefore, Equation C.3 holds, and therefore $t(\epsilon) \leq 0$ for small enough ϵ. □

Theorem 4.17 *Let $\mathbf{G}$ be a grammar with $K \geq 2$ and degree 2. Assume that p is $\langle \mathbf{G}, \boldsymbol{\theta}^* \rangle$ for some $\boldsymbol{\theta}^*$, such that $\theta^*_{1,1} = \theta^*_{2,1} = \mu$ and that $c_1 \leq c_2$. If $A_{\mathbf{G}}(\boldsymbol{\theta}^*)$ is positive definite, then p does not satisfy the Tsybakov noise condition for any (C, κ), where $C > 0$ and $\kappa \geq 1$.*

Proof Define λ to be the eigenvalue of $A_{\mathbf{G}}(\boldsymbol{\theta})$ with the smallest value (λ is positive). Also, define $\mathbf{v}(\boldsymbol{\theta})$ to be a vector indexed by k, i such that

$$v_{k,i}(\theta) = \mathbb{E}[\psi_{k,i}] \log \frac{\theta^*_{k,i}}{\theta_{k,i}}.$$

Simple algebra shows that for any $h \in \mathcal{H}(\mathbf{G})$ (and the fact that $p \in \mathcal{H}(\mathbf{G})$), we have:

$$\mathcal{E}_p(h) = d_{\mathrm{KL}}(p\|h) = \sum_{k=1}^{K} \left(\mathbb{E}_p[\psi_{k,1}] \log \frac{\theta^*_{k,1}}{\theta_{k,1}} + \mathbb{E}_p[\psi_{k,1}] \log \left(\frac{1-\theta^*_{k,1}}{1-\theta_{k,1}} \right) \right)$$

For a $C > 0$ and $\kappa \geq 1$, define $\alpha = C\epsilon^{1/\kappa}$. Let $\epsilon < \alpha$. First, we construct an h such that $d_{\mathrm{KL}}(p\|h) < \epsilon + \epsilon/2$ but $\mathrm{dist}(p, h) > C\epsilon^{1/k}$ as $\epsilon \to 0$. The construction follows. Parametrize h by $\boldsymbol{\theta}$ such that $\boldsymbol{\theta}$ is identical to $\boldsymbol{\theta}^*$ except for $k = 1, 2$, in which case we have:

$$\theta_{1,1} = \theta^*_{1,1} \left(\exp\left(\frac{\alpha + \epsilon/2}{c_1} \right) \right) = \mu \left(\exp\left(\frac{\alpha + \epsilon/2}{c_1} \right) \right) \tag{C.4}$$

$$\theta_{2,1} = \theta^*_{2,1} \left(\exp\left(\frac{-\alpha + \epsilon/2}{c_2} \right) \right) = \mu \left(\exp\left(\frac{-\alpha + \epsilon/2}{c_2} \right) \right) \tag{C.5}$$

Note that $\mu \leq \theta_{1,1} \leq 1/2$ and $\theta_{2,1} < \mu$. Then, we have that:

$$\begin{aligned} d_{\mathrm{KL}}(p\|h) &= \sum_{k=1}^{K} \left(\mathbb{E}_p[\psi_{k,1}] \log \frac{\theta^*_{k,1}}{\theta_{k,1}} + \mathbb{E}_p[\psi_{k,1}] \log \left(\frac{1-\theta^*_{k,1}}{1-\theta_{k,1}} \right) \right) \\ &= \epsilon + c_1 \log \frac{1-\theta^*_{k,1}}{1-\theta_{1,1}} + c_2 \log \frac{1-\theta^*_{k,2}}{1-\theta_{2,1}} \\ &= \epsilon + c_1 \log \frac{1-\mu}{1-\theta_{1,1}} + c_2 \log \frac{1-\mu}{1-\theta_{2,1}} \end{aligned}$$

We also have:

$$c_1 \log \frac{1-\mu}{1-\theta_{1,1}} + c_2 \log \frac{1-\mu}{1-\theta_{2,1}} \leq 0 \tag{C.6}$$

if

$$c_1 \times \frac{1-\mu}{1-\theta_{1,1}} + c_2 \times \frac{1-\mu}{1-\theta_{2,1}} \leq c_1 + c_2 \tag{C.7}$$

(this can be shown by dividing Equation C.6 by $c_1 + c_2$ and then using the concavity of the logarithm function.) From Lemma C.2, we have that Equation C.7 holds. Therefore,

$$d_{\mathrm{KL}}(p \| h) \leq 2\epsilon$$

Now, consider the following, which can be shown through algebraic manipulation:

$$\mathrm{dist}(p, h) = \mathbb{E}\left[\left(\log \frac{p}{h}\right)^2\right] = \sum_{k,k'} \sum_{i,i'} \mathbb{E}[\psi_{k,i} \times \psi_{k',i'}] \left(\log \frac{\theta^*_{k,i}}{\theta_{k,i}}\right) \left(\log \frac{\theta^*_{k',i'}}{\theta_{k',i'}}\right)$$

Then, additional algebraic simplification shows that:

$$\mathbb{E}\left[\left(\log \frac{p}{h}\right)^2\right] = \mathbf{v}(\theta) A \mathbf{v}(\theta)^\top$$

A fact from linear algebra states that:

$$\mathbf{v}(\theta) A \mathbf{v}(\theta)^\top \geq \lambda ||\mathbf{v}(\theta)||_2^2$$

where λ is the smallest eigenvalue in A. From the construction of θ and Equation C.4–C.5, we have that $||\mathbf{v}(\theta)||_2^2 > \alpha^2$. Therefore,

$$\mathbb{E}\left[\left(\log \frac{p}{h}\right)^2\right] \geq \lambda \alpha^2$$

stwhich means $\mathrm{dist}(p, h) \geq \sqrt{\lambda} C \epsilon^{1/\kappa}$. Therefore, p does not satisfy the Tsybakov noise condition with parameters (D, κ) for any $D > 0$. □

Appendix D

Details of Treebanks Used

We detail in Table D.1 information about datasets used in this dissertation.

Language	Tag set	Training		Development		Test		Source	Baselines	
	size	tokens	sent.	tokens	sent.	tokens	sent.		A-R	A-L
Bulgarian	54	40,609	5,890	15,737	1,283	5,032	398	Simov et al. (2002)	18.1	38.6
Chinese	34	27,357	4,775	5,824	350	7,007	348	Xue et al. (2004)	32.9	9.7
Czech	47	67,756	10,674	32,647	2,535	33,147	2,535	Hajič et al. (2000)	24.4	28.3
Danish	25	11,794	1,747	8,455	519	8,257	519	Kromann et al. (2003)	13.2	47.9
Dutch	12	38,565	5,650	19,549	1,335	14,327	1,335	Van der Beek et al. (2002)	28.8	25.8
English	34	55,340	7,179	35,021	1,700	49,363	2,416	Marcus et al. (1993)	30.2	20.4
Greek	38	3,123	476	5,814	270	5,673	270	Prokopidis et al. (2005)	31.7	19.5
Japanese	80	39,121	10,330	14,666	1,700	13,648	1,700	Kawata and Bartels (2000)	67.3	13.8
Portuguese	22	15,976	2,477	14,558	907	5,009	288	Afonso et al. (2002)	25.9	31.1
Slovene	29	4,539	659	2,729	153	2,211	153	Džeroski et al. (2006)	24.4	26.6
Spanish	47	3,849	538	7,013	330	6,597	330	Civit and Martí (2004)	24.8	30.0
Swedish	41	29,468	4,060	14,994	1,100	19,573	1,097	Nivre et al. (2006)	24.2	30.9
Turkish	31	18,873	3,416	7,812	500	6,288	623	Atalay et al. (2003); Oflazer et al. (2003)	61.4	3.9

Table D.1: Information about the treebanks used in this dissertation. "Tag set size" stands for the size of the part-of-speech tag set. Train, development and test columns show the number of tokens and number of sentences in each data set. The training set consists of sentences of length ten or less, as described in the text. The development set and the test set do not have any length restriction. The development set includes unannotated set of sentences from the respective language. A-R (A-L) stands for Attach-Right (Attach-Left), which are attachment accuracy baselines on the test set for all sentences. See text for details.