The Dissertation Committee for Liang Sun
certifies that this is the approved version of the following dissertation:

# Discovering Latent Structures in Syntax Trees and Mixed-type Data

Committee:

Ned Dimitrov, Supervisor

Jason Baldridge

John Hasenbein

Aida Khajavirad

James Scott

# Discovering Latent Structures in Syntax Trees and Mixed-type Data

**by**

**Liang Sun, B.S., M.S.E**

**DISSERTATION**

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

**DOCTOR OF PHILOSOPHY**

THE UNIVERSITY OF TEXAS AT AUSTIN

May 2016

Dedicated to my parents Bin Sun and Qun Li for their endless love, support and encouragement, to my husband Bingqing Wang who always stood by my side sharing life's joys and challenges.

# Acknowledgments

Firstly, I would like to thank my supervisor Dr. Ned Dimitrov for the continuous support of my Ph.D study. He provided insightful discussions about the research. He guided me academically and emotionally through the rough road to finish this thesis.

Besides my supervisor, I would like to thank my advisors in Statistics and Natural Language Processing. I would like to thank Dr. James Scott, who led me to the world of Bayesian Statistics. He provided me many precious opportunities to apply statistical tools in many research fields. And his immense knowledge and insightful comments always guided me in research. I would like to thank Dr. Jason Baldridge, who led me to the world of Natural Language Processing. He showed me how to apply statistical tools to solve the problems in this fascinating field. Without his patience and motivation, it would not be possible for me to start research in NLP without prior knowledge in linguistics.

I would like to thank Dr. John Hasenbein and Dr. Aida Khajavirad for serving as members on my thesis committee. I am thankful for their time, continuous guidance and valuable comments and suggestions.

I would also like to thank many professors and fellow students who have been working with me on various projects. Many thanks go to Dr. Scott Moser, Dr. Carlos Carvalho, Dr. Jesse Windle, Jason Mielens, etc. And many thanks to professors and friends in Operations Research and Industrial Engineering group. It had been a joyful journey with all of them.

Last but not the least, I would like to thank my family: my parents and my husband for supporting me throughout writing this thesis and my life in general.

# Discovering Latent Structures in Syntax Trees and Mixed-type Data

Publication No. __________________

Liang Sun, Ph.D.
The University of Texas at Austin, 2016

Supervisor: Ned Dimitrov

Gibbs sampling is a widely applied algorithm to estimate parameters in statistical models. This thesis uses Gibbs sampling to resolve practical problems, especially on natural language processing and mixed type data. It includes three independent studies. The first study includes a Bayesian model for learning latent annotations. The technique is capable of parsing sentences in a wide variety of languages, producing results that are on-par with or surpass previous approaches in accuracy, and shows promising potential for parsing low-resource languages. The second study presents a method to automatically complete annotations from partially-annotated sentence data, with the help of Gibbs sampling. The algorithm significantly reduces the time required to annotate sentences for natural language processing, without a significant drop in annotation accuracy. The last study proposes a novel factor model for uncovering latent factors and exploring covariation among multiple outcomes of mixed types, including binary, count, and continuous data. Gibbs sampling is used to estimate model parameters. The algorithm successfully discovers correlation structures of mixed-type data in both simulated and real-word data.

# Table of Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

This thesis focuses on the use of statistical tools to resolve practical problems, especially in discovering hidden structure in natural language processing and mixed type data. The thesis includes three chapters. Chapter 2 introduces a Bayesian model and algorithms based on a Gibbs Sampler for learning latent annotations. The technique is capable of parsing sentences in a wide variety of languages and producing results that are on- par with or surpass previous approaches in accuracy. Particularly, the results demonstrate that low-resource language parsing can benefit substantially from the outlined Bayesian approach.

Chapter 3 further describes a method, based on a Gibbs Sampler, for completing annotations from partially-annotated sentence data. The completed annotations can be used for training a standard dependency parser. The experiments show that this strategy improves performance over not using partial annotations for a variety of languages. Moreover, performance competitive with state-of-the-art weakly-supervised parsers can be reached with just a few hours of partial annotation.

Chapter 4 describes a factor model for uncovering latent factors and exploring covariation among multiple outcomes of mixed types. The proposed factor

model is capable of discovering correlations among many types of variables, including binary, count, and continuous types. We create an algorithm to inference the model parameters, and demonstrate that the algorithm successfully recovers correlation structure on simulated data. The algorithm also provides valuable insights from political science data. Finally, we implement the algorithm as an R package, which enables R users to perform factor analysis of data with mixed-types in a fully automatic way.

# Chapter 2

# Parsing low-resource languages using Gibbs sampling for Probabilistic Context-Free Grammars (PCFGs) with latent annotations

## 2.1 Background

This chapter discusses a Bayesian approach to the problem of constituency parsing. A constituency parse tree represents the constituent structure of the sentence, specifically the way it breaks into sub-phrases. An example of parse tree is shown in Figure 2.1. Parse trees have come to play a vital role in modern Natural Languages Processing (NLP). The ability to reconstruct parse tree structure enables other tasks like machine translation, question answering, etc.

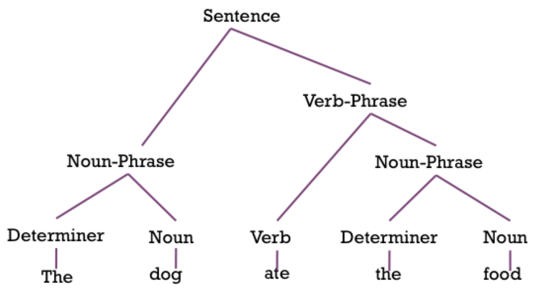


Figure 2.1: A parse tree for the sentence *The dog ate the food*.

The Probabilistic Context-Free Grammar (PCFG) model has been one of the most important formal models in syntactic parsing (Charniak, 2000; Collins, 2003).

The basic PCFG model is briefly introduced here. A Context-Free-Grammar (CFG) G in Chomsky normal form (Chomsky, 1956) is a 4-tuple $(T, N, S, R)$, which consists of:

1. $T$ is a finite set of terminal symbols, the symbols that appear in the final strings.
2. $N$ is a finite set of non-terminal symbols, that are expanded into other symbols.
3. $S$ is a special non-terminal called the start symbol, for example, S.
4. and $R$ is a finite set of production rules of the form $A \to BC$ or $A \to w$, where $A, B, C \in N$ and $w \in T$.

A CFG can be extended into a PCFG by associating production rule with a probability $\theta_{r \in R}$ (Booth and Thompson, 1973). Because all non-terminals must expand, the probabilities for all the rules that expand the same non-terminal must sum to one.

Extensive work has been done in search of an automatic and accurate language parser for low resource languages. A low resource language is one where not much data, either labeled or unlabeled, is available. A primary approach to improve the performance of parsers is to use some form of cross-lingual bootstrapping. For instance, Kuhn (2004b) used multiple languages to induce a monolingual grammar to perform parsing in a low resource language. Similarly, Hwa et al. (2005) used a parallel Chinese/English corpus and an English dependency grammar to create an annotated Chinese corpus and train a Chinese dependency grammar. These approaches are based on a parallel corpus, which is difficult to acquire, or not available at all, for low-resource languages.

In addition to bootstrapping, a second method to address low resource languages is using linguistic universals. For example, Kuhn (2004a) studied various tasks using Q'anjob'al and identified some of the difficulties in handling low-resource languages in general. Bender et al. (2002) seeded newly developed grammars by making use of universal grammars. These approaches, based on linguistic universals, lacks the accuracy of grammars learned from data.

Recently, Probabilistic Context-Free Grammars with latent annotations ( PCFG-LA) (Matsuzaki et al., 2005; Petrov et al., 2006) have been proven to be an effective model for syntactic parsing. Specifically, this approach requires less training material, making it a suitable tool for parsing low-resource languages (Liang et al., 2009; Shindo et al., 2012). Previous PCFG-LA work focuses on algorithms for parameter estimation, including expectation-maximization (EM) (Matsuzaki et al., 2005; Petrov et al., 2006), spectral learning (Cohen et al., 2012, 2013), and variational inference (Liang et al., 2009; Wang and Blunsom, 2013). But all previous work uses a standard Viterbi parse to label a new sentence (Matsuzaki et al., 2005). The work presented in this thesis provides both a new estimation method, as well as a new method for labeling new sentences.

The main contributions of this thesis to this problem are: 1) a novel algorithm for labeling new sentences using a PCFG-LA, based on a Gibbs sampler 2) a novel algorithm for estimating PCFG-LA parameters from training data 3) experiments that demonstrate a parser resulting from the first two contributions is on par with state-of-the-art parsers. Specifically, our algorithms provide excellent results in low resource settings – both artificially limited in the case of English, and naturally limited in the case of Italian, Malagasy, and Kinyarwanda. Furthermore, the novel methods do not depend on supporting materials such as parallel corpora, linguistic universals, or language-specific modifications. This independence of supporting materials make the algorithmic approach a valuable tool for parsing low-resource languages.

This work is published in Sun et al. (2014). The main contributions of this thesis are the models and algorithms presented. The main contribution of the coauthor, Jason Mielens, is data acquisition and performing experiments.

## 2.2 Gibbs sampling for PCFGs

Johnson et al. (2007) built a Bayesian model for PCFG, and developed a Gibbs Sampler for inferencing rule probabilities. This work extends the model and algorithms for parsing PCFG-LA, see Section 2.3. For better understanding of the models and algorithms in Section 2.3, we summarize the Bayesian PCFG and Gibbs sampler defined by Johnson et al. (2007).

The main purpose of this section is to review the Gibbs sampling steps outlined in Johnson et al. (2007). A smaller, secondary purpose is to highlight the fact that these steps can be used for labeling a new sentence. Johnson et al. (2007) use the Gibbs sampling steps solely for estimating the unknown rule probabilities.

The inputs of the Gibbs sampling algorithm are a corpus of unlabeled sentences and a prior for the rule probabilities. The prior can be an uninformed prior, or an informed prior derived from a corpus of labeled sentence. To use the algorithm for labeling a new sentence, one could output the parse tree that appears most often during the sampling process. This labeling technique works particularly well for sentences where relatively few parse trees are possible.

### 2.2.1 Bayesian PCFG

Given an input corpus of sentences $\boldsymbol{w}$=$(w^{(1)}, \cdots, w^{(n)})$, we introduce two quantities that require estimation: 1) a latent variable $\boldsymbol{t}$=$(t^{(1)}, \cdots, t^{(n)})$, representing one parse tree for each sentence and 2) a vector of rule probabilities, one for each production rule in the PCFG, $\boldsymbol{\theta} =< \theta_r >_{r \in R}$. The joint posterior distribution of $\boldsymbol{t}$ and $\boldsymbol{\theta}$ conditioned on $\boldsymbol{w}$ is:

$$\begin{aligned} p(\boldsymbol{t}, \boldsymbol{\theta} \mid \boldsymbol{w}) &\propto p(\boldsymbol{\theta}) p(\boldsymbol{w} \mid \boldsymbol{t}) p(\boldsymbol{t} \mid \boldsymbol{\theta}) \\ &= p(\boldsymbol{\theta}) (\prod_{i=1}^{n} p(w^{(i)} \mid t^{(i)}) p(t^{(i)} \mid \boldsymbol{\theta})) \\ &= p(\boldsymbol{\theta}) (\prod_{i=1}^{n} p(w^{(i)} \mid t^{(i)}) \prod_{r \in R} \theta_r^{f_r(t^{(i)})}) \end{aligned} \tag{2.1}$$

Here $f_r(t)$ is the number of occurrences of rule $r$ in the derivation of $\boldsymbol{t}$; $p(w^{(i)} \mid t^{(i)}) = 1$ if the terminals of $t^{(i)}$ are the sequence $w^{(i)}$, and 0 otherwise. The above expression assumes independence of: the individual sentences, given their parse trees; the individual parse trees, given production rule probabilities; and of the individual rules within a parse tree.

Define $p(\boldsymbol{\theta} \mid \boldsymbol{\alpha})$ as prior on $\boldsymbol{\theta}$ – a product of Dirichlet distributions, parameterized by a vector $\boldsymbol{\alpha}$, $p(\boldsymbol{\theta} \mid \boldsymbol{\alpha}) = \prod_{A \in N} Dir(\theta_A \mid \alpha_A)$. Here each non-terminal $A \in N$ is associated with one Dirichlet distribution parameterized by $\alpha_A$, so that each production expanding $A$: $A \to \beta \in R$ has a corresponding Dirichlet parameter $\alpha_{A \to \beta}$.

This Dirichlet parameterization allows for easy updates to the conditional posterior. Specifically, the Dirichlet distribution is conjugate to the Multinomial distribution, which we used in (2.1) to model the likelihood a tree given rule probabilities. The conjugate prior update produces posteriors on the parameters $\theta_A$ as follows:

$$\begin{aligned} p_G(\boldsymbol{\theta} \mid \boldsymbol{t}, \boldsymbol{\alpha}) &\propto p_G(\boldsymbol{t} \mid \boldsymbol{\theta}) p(\boldsymbol{\theta} \mid \boldsymbol{\alpha}) \\ &\propto \left(\prod_{r \in R} \theta_r^{f_r(\boldsymbol{t})}\right)\left(\prod_{r \in R} \theta_r^{\alpha_r - 1}\right) \\ &= \prod_{r \in R} \theta_r^{f_r(\boldsymbol{t}) + \alpha_r - 1} \end{aligned} \tag{2.2}$$

which is still a Dirichlet distribution with updated parameter $f_r(\boldsymbol{t}) + \alpha_r$ for each rule $r \in R$.

### 2.2.2 Gibbs sampler

To sample the joint posterior $p(\boldsymbol{t}, \boldsymbol{\theta} \mid \boldsymbol{w})$, Johnson et al. presents a Gibbs sampler to sample production probabilities $\boldsymbol{\theta}$ and then trees $\boldsymbol{t}$ iteratively from these conditional distributions:

$$p(\boldsymbol{\theta} \mid \boldsymbol{t}, \boldsymbol{w}, \boldsymbol{\alpha}) = \prod_{A \in N} Dir(\theta_A \mid f_A(\boldsymbol{t}) + \alpha_A) \tag{2.3}$$

$$p(\boldsymbol{t} \mid \boldsymbol{\theta}, \boldsymbol{w}, \boldsymbol{\alpha}) = \prod_{i=1}^{n} p(t^{(i)} \mid w^{(i)}, \boldsymbol{\theta}) \tag{2.4}$$

```
Require: $A$ is parent node of a binary rule; $w_{i,k}$ is a span of words: $i+1<k$
  function TREESAMPLER($A, i, k$)
      for $i < j < k$ and pair of child nodes of $A$:$B, C$ do
          $P(j, B, C) = \frac{\theta_{A \to BC} \cdot p_{B,i,j} \cdot p_{C,j,k}}{p_{A,i,k}}$
      end for
  Sample $j*, B*, C*$ from multinomial distribution for $(j, B, C)$ with probabilities calculated above
      return $j*, B*, C*$
  end function
```

Algorithm 1: Sampling split position and rule to expand parent node. The probabilities $p_{A,i,j}$ for all $A \in N$ and $0 \leq i < j \leq l$ are pre-computed using the recursive rules in Equation 5 and 6.

*Step 1: Sample Rule Probabilities.* Given trees $\boldsymbol{t} = (t^{(1)}, \cdots, t^{(n)})$ for sentences $1, \cdots, n$ and prior $\boldsymbol{\alpha}$, the production probabilities $\theta_A$ for each nonterminal $A \in N$ are sampled from a Dirichlet distribution with parameters $f_A(\boldsymbol{t}) + \alpha_A$. $f_A(\boldsymbol{t})$ is a vector, and each component of $f_A(\boldsymbol{t})$ is the number of occurrences of one rule expanding nonterminal $A$ in all trees.

*Step 2: Sample Tree Structures.* There exists an efficient sampling scheme to sample trees from $p(t^{(i)} \mid w^{(i)}, \boldsymbol{\theta})$ introduced in previous work (Goodman, 1998; Finkel et al., 2006; Johnson et al., 2007). To describe this algorithm, consider a particular sentence $w$ and its associated tree $t$. There are two parts to the algorithm: The first constructs an inside table as in the Inside-Outside algorithm for PCFGs (Lary and Young, 1990); The second selects the tree by recursively sampling productions from top to bottom.

Consider a sentence $w$ made of terminals $(w_0, \cdots, w_l)$, with sub-spans $w_{i,k} = (w_{i+1}, \cdots, w_k)$. Given $\boldsymbol{\theta}$, for each nonterminal $A$ and each word span

$w_{i,k} : 0 \leq i < k \leq l$, compute the probability, $p_{A,i,k}$, that words $i$ through $k$ are produced by $A$. The table is computed recursively by

$$p_{A,k-1,k} = \theta_{A \rightarrow w_k} \tag{2.5}$$

$$p_{A,i,k} = \sum_{A \rightarrow BC \in R} \sum_{i<j<k} \theta_{A \rightarrow BC} \cdot p_{B,i,j} \cdot p_{C,j,k} \tag{2.6}$$

for all $A, B, C \in N$ and $0 \leq i < j < k \leq l$.

The resulting inside probabilities are then used to generate trees from the distribution of all valid trees of the sentence. The tree is generated from top to bottom recursively with the function $TreeSampler$ defined in Algorithm 1.

## 2.3 PCFG with latent annotations

In this section, we extend the Gibbs sampling algorithm to PCFG-LA. This is the first work that uses algorithms based on Gibbs sampling for parsing new sentences with PCFG-LA. PCFG-LAs have been shown to be a very effective model for phrase structure parsing (Matsuzaki et al., 2005). Most related work focuses on inference methods, while using Viterbi parse to parse new sentences.

When training data, i.e. parsed trees, are available, rule frequencies can be directly extracted and used as priors for the Gibbs sampler for PCFG. However, the same is not true for PCFG-LA. Because the data does not include the latent annotation label, there is no direct way to get a prior for PCFG-LAs. We develop a novel algorithm to assign latent annotations to trees that have no latent annotations. The output of the algorithm allows us to then use rule counts to construct a prior for a Gibbs sampler for PCFG-LA.

### 2.3.1 PCFG-LA

A Context-Free Grammar with latent annotations (CFG-LA) (Matsuzaki et al., 2005; Petrov et al., 2006) refines the non-terminals of a CFG. Define a CFG-LA model as a tuple $(T, N, H, S, R)$, which consists of:

1. $T$ is a finite set of terminal symbols, the symbols that appear in the final strings.
2. $N$ is a finite set of non-terminal symbols, that are expanded into other symbols.
3. $H$ is a finite set of latent annotations symbols, $H = \{1, \cdots, K\}$. And $N[H]$ denotes the set of complete non-terminal symbols, i.e. $N[H] = \{A[x] \mid A \in N, x \in H\}$.
4. $S$ is a special non-terminal called the start symbol, for example, S.
5. and $R$ is a finite set of production rules. It contains rules with the following forms:
   (a) $S \rightarrow S[x]$ where $x \in H$
   (b) $A[x] \rightarrow BC$ or $A[x] \rightarrow w$, where $A, B, C \in N$, $x \in H$ and $w \in T$
   (c) $A[x]B, C \rightarrow y, z$, where $B, C \in N$ are child nodes of $A[x]$, and $y, z \in H$ are latent annotations assigned to $B, C$.

A CFG-LA can be extended into a PCFG-LA by associating production rule with a probability. Given three forms of rules listed above, we use $\pi_{S\rightarrow S[x]}$ to denote the probability of rule $S \rightarrow S[x]$ for $x \in H$; $\theta_{A[x]\rightarrow U}$ to denote the probability of rule $A[x] \rightarrow U$, where $U \in (N \times N) \cup T$; $\beta_{A[x],B,C\rightarrow y,z}$ to denote the probability of assigning latent annotation $y, z$ to child nodes $B, C$ of $A[x]$. The probabilities for all the rules that expand the same left-part must sum to one.

### 2.3.2 Bayesian PCFG-LA

Given an input corpus of sentences $\boldsymbol{w}=(w^{(1)}, \cdots, w^{(n)})$, we introduce two variables for representing latent tree structures. The first one $\boldsymbol{t}=(t^{(1)}, \cdots, t^{(n)})$ denotes the complete trees with latent annotations; the second one $\boldsymbol{\tau}=(\tau^{(1)}, \cdots, \tau^{(n)})$ denotes observable incomplete trees, for example, trees without latent annotations. The joint posterior distribution of $\boldsymbol{t}$ and $\boldsymbol{\pi}, \boldsymbol{\theta}, \boldsymbol{\beta}$ conditioned on $\boldsymbol{w}$ and $\boldsymbol{\tau}$ is:

$$\begin{aligned} p(\boldsymbol{t}, \boldsymbol{\pi}, \boldsymbol{\theta}, \boldsymbol{\beta} \mid \boldsymbol{w}, \boldsymbol{\tau}) &\propto p(\boldsymbol{\pi}, \boldsymbol{\theta}, \boldsymbol{\beta}) p(\boldsymbol{w}, \boldsymbol{\tau} \mid \boldsymbol{t}) p(\boldsymbol{t} \mid \boldsymbol{\pi}, \boldsymbol{\theta}, \boldsymbol{\beta}) \\ &= p(\boldsymbol{\pi}) p(\boldsymbol{\theta}) p(\boldsymbol{\beta}) (\prod\nolimits_{i=1}^{n} p(w^{(i)}, \tau^{(i)} \mid t^{(i)}) p(t^{(i)} \mid \boldsymbol{\pi}, \boldsymbol{\theta}, \boldsymbol{\beta})) \end{aligned}$$

Here $p(w^{(i)}, \tau^{(i)} \mid t^{(i)}) = 1$ if the terminals of $t^{(i)}$ are the sequence $w^{(i)}$ and partial structure of $t^{(i)}$ is $\tau^{(i)}$, and 0 otherwise. The above expression assumes independence of: the individual sentences, given their parse trees; the individual parse trees, given production rule probabilities; and of the individual rules within a parse tree. Also assume that $\boldsymbol{\pi}$, $\boldsymbol{\theta}$ and $\boldsymbol{\beta}$ are independent to get $p(\boldsymbol{\pi}, \boldsymbol{\theta}, \boldsymbol{\beta}) = p(\boldsymbol{\pi}) p(\boldsymbol{\theta}) p(\boldsymbol{\beta})$.

To learn parameters $\boldsymbol{\pi}$, $\boldsymbol{\theta}$, $\boldsymbol{\beta}$, we use products of Dirichlet distributions as priors for $\boldsymbol{\pi}$ $\boldsymbol{\theta}$ and $\boldsymbol{\beta}$. The distribution for all rules expanding $A[x]$ is:

$$p(\boldsymbol{\theta} \mid \boldsymbol{\alpha^{\theta}}) = \prod_{A \in N, x \in H} Dir(\theta_{A[x]} \mid \alpha^{\theta}_{A[x]})$$

The distribution for latent annotations associated with child nodes of $A[x] \rightarrow BC$ is:

$$p(\boldsymbol{\beta} \mid \boldsymbol{\alpha^{\beta}}) = \prod_{A[x],B,C \in N[x] \times N \times N} Dir(\beta_{A[x],B,C} \mid \alpha^{\beta}_{A[x],B,C}).$$

And the distribution for latent annotations associated to $S$ is:

$$p(\boldsymbol{\pi} \mid \boldsymbol{\alpha^{\pi}}) = Dir(\pi \mid \alpha^{\pi})$$

This Dirichlet parameterization allows for easy updates to the conditional posterior. For all unary and binary rules expanding $A[x]$:

$$\theta_{A[x]} \mid \boldsymbol{t}, \alpha^{\theta} \sim Dir(f_{A[x]}(\boldsymbol{t}) + \alpha^{\theta}_{A[x]})$$

Here $f_{A[x]}(\boldsymbol{t})$ is a vector, and each component of $f_{A[x]}(\boldsymbol{t})$, $f_r(\boldsymbol{t})$, is the number of occurrences of rule $r$ expanding nonterminal $A[x]$ in all trees. Also, for combination of latent annotations assigned to $B, C$ in rule $A[x] \rightarrow B, C$:

$$\beta_{A[x],B,C} \mid \boldsymbol{t}, \alpha^{\beta} \sim Dir(f_{A[x],B,C}(\boldsymbol{t}) + \alpha^{\beta}_{A[x],B,C})$$

Here, each component of $f_{A[x],B,C}((t))$, $f_d(\boldsymbol{t})$, is the number of occurrences of combination $d \in \{y, z : y \in H, z \in H\}$ in $\boldsymbol{t}$. And similarly $\pi$ can be updated by:

$$\pi \mid \boldsymbol{t}, \alpha^{\pi} \sim Dir(f_S(\boldsymbol{t}) + \alpha^{\pi})$$

Here, each component of $f_S(\boldsymbol{t})$, $f_{S[x]}(\boldsymbol{t})$, is the number of occurrences of $S[x]$ in $\boldsymbol{t}$.

```
Require: $w_1, \cdots, w_n$ are raw sentences; $\theta_0, \beta_0$ are initial values; $\alpha^\theta, \alpha^\beta$ are priors; $M$ is the number of iterations
  function PARSE($w_1, .., w_n, \theta_0, \beta_0, \alpha^\theta, \alpha^\beta, M$)
      for iteration $i = 1$ to $M$ do
          for sentence $s = 1$ to $n$ do
              Calculate Inside Table
              Sample tree nodes and associated latent annotations, get tree structure $t_s^{(i)}$
          end for
          Sample $\theta^{(i)}$, $\beta^{(i)}$
      end for
      for sentence $s = 1$ to $n$ do
          Remove the latent annotations to get unannotated trees $T_s^{(1)}, \cdots, T_s^{(M)}$
          Find the mode of $T_s^{(1)}, \cdots, T_s^{(M)}$: $T_s$
      end for
      return $T_1, \cdots, T_n$
  end function
```

Algorithm 2: Parsing new sentences with PCFG-LA

### 2.3.3 Gibbs sampling for PCFG-LA with no observed incomplete trees

When no observed incomplete trees are available, i.e. to parse raw text, the sampler in Section 2.2 is extended to PCFG-LA. Given priors $\boldsymbol{\alpha}^{\boldsymbol{\theta}}$, $\boldsymbol{\alpha}^{\boldsymbol{\beta}}$, $\boldsymbol{\alpha}^{\boldsymbol{\pi}}$ and raw text, the algorithm alternates between two steps. The first samples trees for the entire corpus; the second samples $\boldsymbol{\theta}$, $\boldsymbol{\beta}$ and $\boldsymbol{\pi}$ from Dirichlet distributions with updated parameters, combining priors and counts from sampled trees. The algorithm then alternates between these steps until convergence. The outputs are samples of $\boldsymbol{\theta}$, $\boldsymbol{\beta}$, $\boldsymbol{\pi}$ and annotated trees.

The parsing process is specified in Algorithm 2. The first step assigns a tree to a sentence, say $w_{0,l}$, $l$ is the length of this sentence. An inside table is constructed first (see Section 2.2). Each entry in the table stores the probability that a word span is produced by a given annotated nonterminal. For root node $S$ one annotation is sampled based on all $p_{S[x],0,l}$, $x \in H$ and $\boldsymbol{\pi}$. Assume that $x$ is sampled for $S$, a rule to expand $S[x]$ and possible splits of the span $w_{0,l}$ is further sampled jointly. Assume that nonterminals $B, C$ are sampled to expand $S[x]$, where $B$ is responsible for $w_{0,j}$ and $C$ is responsible for $w_{j,l}$. Annotations for $B, C$ are further sampled together, say $y, z$. Then rules and split positions are sampled to expand $B[y]$ and $C[z]$, and continue until reaching the terminals.

Once trees (with latent annotations) are available, the step of sampling $\boldsymbol{\theta}$, $\boldsymbol{\beta}$ and $\boldsymbol{\pi}$ from a Dirichlet distribution is direct. The algorithm needs to count the number of occurrences $f_r(t)$ for each rule $r$ like $A[x] \rightarrow U$, $U \in (N \times N) \cup T$ in updated annotated trees $\boldsymbol{t}$, and draw $\theta_{A[x]}$ from the updated Dirichlet distribution $Dir(f_{A[x]}(\boldsymbol{t}) + \alpha^{\theta}_{A[x]})$. The algorithm also needs to count the number of occurrences of $f_d(t)$ for each combination of $(y, z) \in H \times H$ assigned to $B, C$ given $A[x] \rightarrow B, C$ in $\boldsymbol{t}$, and draw $\beta_{A[x],B,C}$ from the updated Dirichlet distribution $Dir(f_{A[x],B,C}(\boldsymbol{t}) + \alpha^{\beta}_{A[x],B,C})$ similarly. And finally the algorithm needs to draw $\pi$ from Dirichlet distribution $Dir(f_S(\boldsymbol{t}) + \alpha^{\pi})$.

To parse a sentence first calculate the inside table. Then sample a tree using the inside table. The algorithm iterates between these two steps. The details of these two steps are shown below.

*Calculate the inside table.* Given $\boldsymbol{\theta}, \boldsymbol{\beta}, \boldsymbol{\pi}$ and a string $\boldsymbol{w}{=}w_{0,l}$, a table is constructed with entries $p_{A[x],i,k}$ for each $A{\in}N$, $x \in H$ and $0 \leq i < k \leq l$, where $p_{A[x],i,k}$ is the probability that words $i$ through $k$ were produced by the annotated nonterminal $A[x]$. The table can be computed recursively, for all $A \in N$, $x \in H$, by

$$p_{A[x],k-1,k} = \theta_{A[x] \rightarrow w_k}$$
$$p_{A[x],i,k} = \sum_{A[x] \rightarrow BC: BC \in N \times N} \sum_{j: i<j<k} \sum_{yz \in H \times H} \theta_{A[x] \rightarrow BC} \beta_{A[x]BC \rightarrow yz} p_{B[y],i,j} p_{C[z],j,k}$$

*Sample the tree, top to bottom.* First, from start symbol $S$, sample latent annotation from multinomial with probability $\pi_{S[x]} p_{S[x],0,l}$ for each $x \in H$. Next, given annotated non-terminal $A[x]$ and $i, k$, sample possible child nodes and split positions from multinomial with probability:

$$p(B, C, j) = \frac{1}{p_{A[x],i,k}} \sum_{y,z \in H} \theta_{A[x] \rightarrow BC} \beta_{A[x]BC \rightarrow yz} p_{B[y],i,j} p_{C[z],j,k}$$

Here the probability is calculated by marginalizing all possible latent annotations for $B, C$, and $\theta_{A[x] \rightarrow BC} \beta_{A[x]BC \rightarrow yz}$ is the probability of choosing $B[y], C[z]$ to expand $A[x]$, and $p_{B[y],i,j}p_{C[z],j,k}$ are the probabilities for $B[y]$ and $C[z]$ to be responsible for word span $w_{i,j}$ and $w_{j,k}$ respectively. And $p_{A[x],i,k}$ is the normalizing term.

Third, given $A[x], B, C, i, j, k$, sample annotations for $B, C$ from multinomial with probability:

$$p(y,z) = \frac{\beta_{A[x]BC\to yz}p_{B[y],i,j}p_{C[z],j,k}}{\sum_{y,z}\beta_{A[x]BC\to yz}p_{B[y],i,j}p_{C[z],j,k}}$$

A crucial aspect of this procedure is that all trees can be sampled independently. This parallel process produces a substantial speed gain that is important particularly when using more latent annotations. After all trees have been sampled (independently), the counts from each individual tree are combined prior to the next sampling iteration.

### 2.3.4 Gibbs sampling for PCFG-LA with observed incomplete trees

When observed incomplete trees are available, we develop an algorithm based on Gibbs sampling to learn the probabilities of production rules. In this work, the observed incomplete trees are given by training data: parse trees without annotations. Gibbs sampling is performed on the training data by first iteratively sampling probabilities and then assigning annotations to tree nodes. The average counts of annotated production rules from sampled trees is used to produce the prior $\boldsymbol{\alpha}^{\boldsymbol{\theta}}$ $\boldsymbol{\alpha}^{\boldsymbol{\beta}}$ and $\boldsymbol{\alpha}^{\boldsymbol{\pi}}$ incorporated into parsing raw sentences.

First index the non-terminal nodes of each tree $T$ by $1, 2, \cdots$ from top to bottom, and left to right. Then the sampler iterates between two steps. The first samples $\boldsymbol{\theta}$, $\boldsymbol{\beta}$ and $\boldsymbol{\pi}$ given annotated trees (as in Section 2.3.3). The second samples

```
Require: $T_1, \cdots, T_n$ are fully parsed trees; $\theta_0, \beta_0$ are initial values; $\alpha^{\theta_0}, \alpha^{\beta_0}$ are priors; $M$ is the number of iterations
  function ANNO($T_1, \cdots, T_n, \theta_0, \beta_0, \alpha^{\theta_0}, \alpha^{\beta_0}, M$)
      for iteration $i = 1$ to $M$ do
          for sentence $s = 1$ to $n$ do
              Calculate inside probability
              Sample latent annotations for each node in the tree, get tree with latent annotations $t_s^{(i)}$
          end for
          Sample $\theta^{(i)}$, $\beta^{(i)}$
      end for
      return Mean of number of occurrences of production rules and associated latent annotations from all sampled annotated trees
  end function
```

Algorithm 3: Learning prior from training

latent annotations for nonterminal nodes in parsed trees, which also takes two steps. The first step is to calculate and store the probability that the node is annotated by $x$ for each node in the tree. The second step is to jointly sample latent annotations for child nodes of root nodes, and then continue this process from top to bottom until reaching the pre-terminal nodes. The main difference from the procedure above is that we know the unannotated tree structure, and we know the spans for each unannotated non-terminal. This reduces the problem to simply sampling latent annotations.

*Step one: inside probabilities.* Given tree $T$, compute $b_T^i[x]$ for each non-terminal $i$ recursively:

1. If node $N_i$ is a pre-terminal node above terminal symbol $w$, then for $x \in H$

$$b_T^i[x] = \theta_{N_i[x] \to w}$$

2. Otherwise, let $j, k$ be two child nodes of $i$, then for $x \in H$

$$b_T^i[x] = \sum_{y,z \in H} \theta_{N_i[x] \to N_j N_k} \beta_{N_i[x] N_j N_k \to y,z} b_T^j[y] b_T^k[z]$$

*Step two: outside sampling.* Given inside probability $b_T^i[x]$ for every non-terminal $i$ and all latent annotations $x \in H$, the algorithm samples latent annotations top to bottom:

1. If node $i$ is the root node ($i = 1$), then sample $x \in H$ from a multinomial distribution with $f_T^i[x] = \pi(N_i[x])$.

2. For a parent node with sampled latent annotation $N_i[x]$ with children $N_j, N_k$, sample latent annotations for these two nodes from a multinomial distribution with

$$f_T^i[y, z] = \frac{1}{b_T^i[x]} \cdot \theta_{N_i[x] \to N_j N_k} \beta_{N_i[x] N_j N_k \to y,z} b_T^j[y] b_T^k[z]$$

After training, the average counts of sampled annotated rules, combinations of latent annotations and latent annotations for $S$ are collected. They are used as priors $\boldsymbol{\alpha}^{\boldsymbol{\theta}}, \boldsymbol{\alpha}^{\boldsymbol{\beta}}, \boldsymbol{\alpha}^{\boldsymbol{\pi}}$ in the algorithm for parsing raw sentences.

## 2.4 Experiments

We perform experiment on five languages with varying amounts of training data in order to understand parsing efficacy using PCFG-LA for low-resource languages. The results are compared to previously established baselines: for all languages, both a standard unsmoothed PCFG and the Bikel parser; additionally, for English and Chinese, to state-of-the-art parsers. Specifically, for Chinese, the comparison is with Huang & Harper (2009), using their results that only use the Chinese Treebank (CTB) without domain knowledge. For English, the comparison is with Liang et al. (2009).

### 2.4.1 Data

English (ENG) and Chinese (CHI) are the two main languages used for this work; they are commonly used in parser evaluation and have previous examples of statistical parsers using a Bayesian framework. Because this work primarily is interested in parsing low-resource languages, we include results for Kinyarwanda (KIN) and Malagasy (MLG) – languages without substantial existing treebanks. Finally, Italian (ITL) is used as a middle-ground language.

For English, Chinese, and Italian, we use pre-existing treebanks. For English, this work uses the Wall-Street Journal section of the Penn Treebank (WSJ) (Marcus et al., 1993). The data split is sections 02-21 for training, section 22 for development, and section 23 for testing. For Chinese, the Chinese Treebank (CTB5) (Xue et al., 2005) was used. The data split is files 81-899 for training, files 41-80 for development, and files 1-40/900-931 for testing. The ITL data is from the Turin University Treebank (TUT) (Bosco et al., 2000) and consists of 2,860 Italian sentences from a variety of domains. It was split into training, development, and test sets with a 70/15/15 percentage split.

For the low-resource languages, we use significantly smaller but also pre-existing data sources. We use section 02 of the WSJ data as small data set for ENG data in order to compare to the results of Liang et al. (2009). We further create an artificially small data set for ENG data containing about 200 sentences from WSJ data. This dataset has similar size of KIN and MLG data. And this dataset is used to demonstrate the convergence behavior and performance of our algorithm on extremely low-resource languages.

KIN and MLG are naturally low-resource languages. KIN data comes from transcripts of testimonies by survivors of the Rwandan genocide provided by the Kigali Genocide Memorial Center, along with a few BBC news articles. MLG data are articles from the websites Lakroa and La Gazette and Malagasy Global Voices. Both datasets are described in Garrette and Baldridge (2013). These languages have significantly smaller datasets than English and Chinese. Specifically, the KIN dataset contains 677 sentences, while the MLG dataset has only 113. In comparison, the English data set has about 25000 sentences, and the Chinese data set has about 20000 sentences.

### 2.4.2 Practical refinements

**Data preprocessing** As a preprocessing step, binarization and normalization of data is performed. All trees are converted into Chomsky Normal-Form (CNF) such that all non-terminal productions are binary and all unary chains are removed. Additional standard normalization is performed. Functional tags (e.g. the *SBJ* part of NP-SBJ), empty nodes (traces), and indices are removed. The binarization is simple: given a parent, select the rightmost child as the head and add a stand-in node that contains the remainder of the original children; the process then

| System | K=1 | K=2 | K=4 | K=8 | K=16 |
|---|---|---|---|---|---|
| Unsmoothed PCFG | 40.2 | — | — | — | — |
| Bikel Parser | 57.9 | — | — | — | — |
| Liang et al. 07 | 60.5 | 71.1 | 77.2 | **79.2** | 78.2 |
| Berkeley Parser | 60.8 | **74.4** | **78.4** | 79.1 | **78.7** |
| Gibbs PCFG-LA | **61.0** | 71.3 | 76.6 | 78.7 | 78.0 |

Table 2.1: F1 scores for small English training data experiments. 'K' is the number of latent annotations – $K = 1$ represents a vanilla, unannotated PCFG.

recurses. This simple technique uses no explicit head-finding rules, which eases cross-linguistic applicability. From this normalized data, latent PCFGs are trained with $K = 1, 2, 4, 8, 16, 32$ (where $K = 1$ is equivalent to the standard PCFG described in Section 2.2).

**Unknown word handling** A similar unknown word handling procedure to Liang et al. (2009) is used in this work. Features associated with every word from the raw corpus are extracted, these features include surrounding context words as well as substring suffix/prefix features. Using these features fifty clusters are produced using k-means. Then, as a pre-parsing step, all words occurring less than five times are replaced with their cluster label - this simulates unknown words for training. Finally, during evaluation, any word not seen in training was also replaced with its corresponding cluster label. This final step is simple because there are no 'unknown unknowns' in the corpus, as the clustering has been performed over the entire corpus prior to training. This approach is similar to methods in Dasgupta & Ng (2007).

## 2.5 Results

In this section the performance is evaluated with F1 scores. Every parse tree can be represented by a series of brackets. Take the sentence in Figure 1 as example, the gold standard brackets for *The dog ate the food* are $(S-(0,5), NP-(0,2), VP-(2,5), NP-(3,5))$, where, for example, $NP-(3,5)$ means that $Noun-Phrase$ is responsible for words $the\ food$. The ratio of the number of correct brackets to the total number of golden brackets is labeled precision. And the ratio of the number of correct brackets to the total number of brackets sampled is labeled recall. And F1 score is the harmonic mean of precision and recall.

Tables 2.1 and 2.2 show performance when training on smaller amounts of data: section 02 of the WSJ, pretending that English is a low-resource language. The results show that the basic Gibbs PCFG-LA (where K=1), with an F1-score of 61.0, substantially outperforms not only an unsmoothed PCFG (the simplest baseline), but also the Bikel parser (Bikel, 2004b) trained on the same amount of data. Table 2.1 also shows further large gains are obtained from using latent annotations—from 60.5 for K=1 to 78.7 for K=8.

| System | WSJ Sec. 02 | KIN | MLG |
|---|---|---|---|
| Berkeley Parser | **78.3 ± 0.93** | 60.6 ± 1.1 | 52.2 ± 2.0 |
| Gibbs PCFG-LA | 76.7 ± 0.63 | **67.2 ± 0.92** | **57.5 ± 1.1** |

Table 2.2: F1 scores with standard deviation over ten runs of small training data, K=4.

The Gibbs PCFG-LA also compares quite favorably to the PCFG-LA of Liang et al. (2009)—slightly better for K=1 and K=2 and slightly worse for K=4

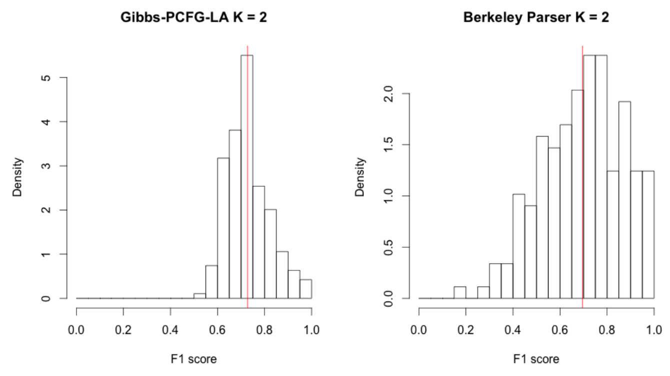
| System | F1 / StDev |
|---|---|
| Berkeley Parser | **77.5 ± 2.1** |
| Gibbs PCFG | 77.0 ± 1.4 |

Table 2.3: F1 scores with standard deviations over twenty runs, training on individual WSJ sections (02-21).

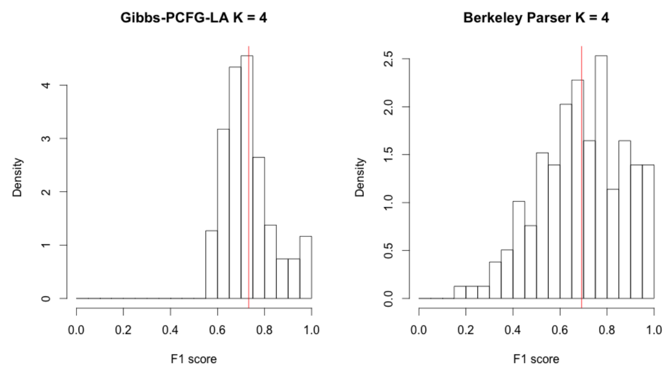
and K=8. Table 2.2 shows that the Gibbs PCFG-LA is able to produce results with a smaller amount of variance relative to the Berkeley Parser, even at low training sizes. This trend is repeated in Table 2.3, which shows that the Gibbs PCFG-LA also produces less variance when training on different single sections of the WSJ relative to the Berkeley Parser, although it again produces slightly lower F1 scores.

The Gibbs PCFG-LA is able to produce results on sentences-level with a smaller amount of variance relative to the Berkeley Parser, especially for extremely low resource languages. We use ENG data with 200 sentences for demonstration. Figure 2.2 shows F1 scores for every single sentence produced by Gibbs PCFG-LA and Berkeley Parser respectively. We can see that Gibbs PCFG-LA, compared to Berkeley Parser, has significant smaller variance on F1 scores across all test sentences. Moreover, for K=2 and K=4, we can see Gibbs PCFG-LA also produces higher average F1 scores on this extremely low resource languages, which is in line with the good performance of Gibbs PCFG-LA on naturally low resource languages- KIN and MLG.

To investigate the convergence of Gibbs samplers, we plot one parameter, the probability of a specific annotated production rule over 50 training iterations and 10 testing iterations. The experiment is trained with WSJ data with 200 sentences



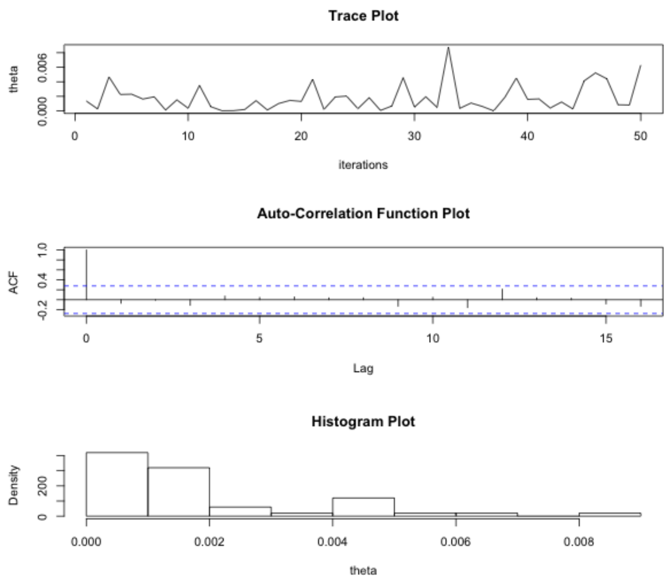
(a) K = 2



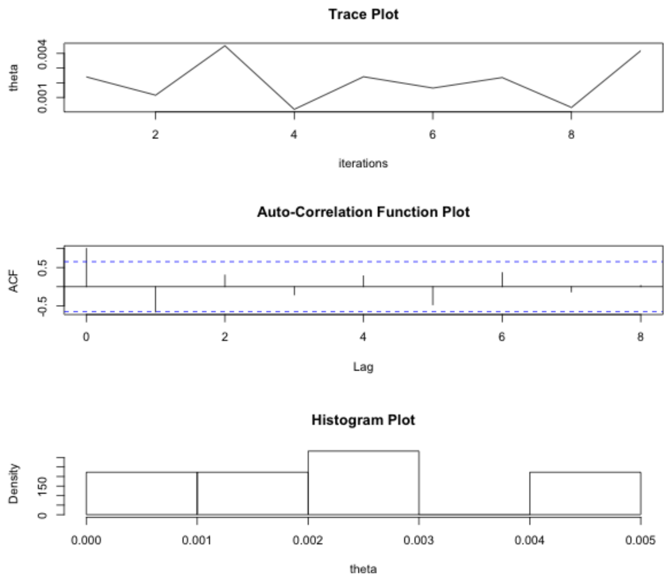
(b) K = 4

Figure 2.2: Histograms for F1 scores produced by Gibbs PCFG-LA and Berkeley Parser, K = 2, 4. The red vertical lines are average F1 scores.

and the number of latent annotations is 2. Figure 2.3 shows the trace plots, the auto-correlation function plots and the histograms of sampled values from Gibbs PCFG-LA in training (Figure 2.3a) and Gibbs PCFG-LA in testing (Figure 2.3b). The lag-k auto-correlation function (acf) estimates correlation between observations k steps apart. We can see that in training, the acf drops quickly, which indicates fast convergence, and allows us to choose small number of iterations for training. In testing, the plot also shows fast decay of acf.



(a) Rule probability sampled in training



(b) Rule probability sampled in testing

Figure 2.3: Plots of one sampled rule probability produced by Gibbs PCFG-LA, K = 2

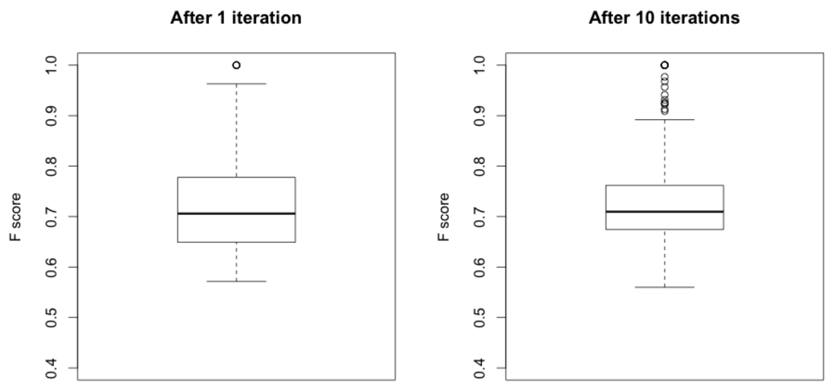


Figure 2.4: Boxplots of F1 scores produced by Gibbs PCFG-LA, K = 2 after 1 testing iteration and 10 testing iterations

Figure 2.4 shows the boxplots for F1 scores for every single sentence produced by Gibbs PCFG-LA with K=2 after 1 testing iteration and 10 testing iterations. From the plot we can see that the average F1 score only increases a little with more iterations due to strong prior we get from training stage. However, we can see with more testing iterations, we have smaller variance on F1 scores across all test sentences, and more sentences achieved good performance ($> 0.9$) in parsing. So we choose 10 iterations for testing here due to time limit in practice.

Figure 2.5 shows how F1 score varies with increasing number of training iterations. We trained the Gibbs PCFG-LA with K = 4 with full ENG data. And the figure shows the performance increases steadily with more training iterations.

We introduce a new parameter $\lambda$ to denote the ratio of the counts extracted from training data as prior to the counts extracted from current parse trees. In (2.3), we update rule probabilities by sampling from a Dirichlet distribution with

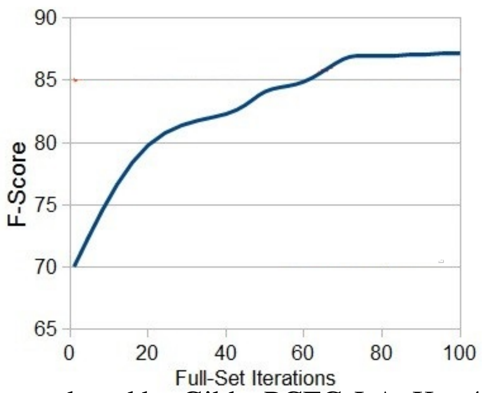


Figure 2.5: F1 scores produced by Gibbs PCFG-LA, K = 4, varying full-set training iterations

parameters $f_A(\boldsymbol{t})+\alpha_A$. Here $f_A(\boldsymbol{t})$ are counts from current updated parse trees. And $\alpha_A$ is a prior that learned from rule counts from training data, which can be viewed as $\lambda \cdot f_A(\boldsymbol{t_{train}})$. Larger $\lambda$ indicates that we are not willing to move away from prior. It is possible that the value of lambda can make a difference in performance, as evidenced by Figure 2.6 (with an optimal value was obtained with an $\lambda$ value of 5 for this small training set).

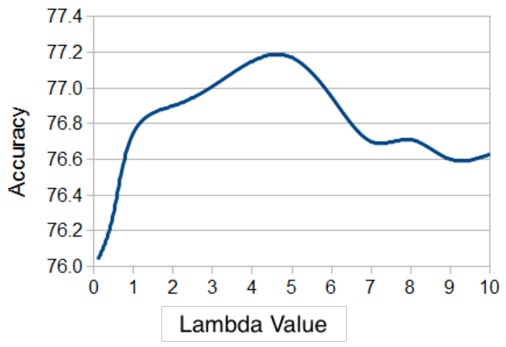


Figure 2.6: Accuracy by varying $\lambda$ levels for small English data.

Table 2.4 shows the results for the main experiments. Due to long training time per iteration for large training data sets, in practice, this work takes 25 itera-

| Condition | ENG | CHI | ITA | KIN | MLG |
|---|---|---|---|---|---|
| Unsmoothed PCFG | 69.9 | 66.8 | 62.1 | 45.9 | 49.2 |
| Liang et al. 07 | 87.1 | — | — | — | — |
| Huang & Harper09 | — | **84.1** | — | — | — |
| Bikel Parser | 86.9 | 81.1 | **74.5** | 55.7 | 49.5 |
| Berkeley Parser | **90.1** | 83.4 | 71.6 | 61.4 | 51.8 |
| Gibbs PCFG,K=1 | 79.3 | 75.4 | 66.3 | 58.5 | 55.1 |
| Gibbs PCFG,K=2 | 82.6 | 79.8 | 69.3 | 65.0 | 57.0 |
| Gibbs PCFG,K=4 | 86.0 | 82.3 | 71.9 | 67.2 | 57.8 |
| Gibbs PCFG,K=16 | 87.2 | 83.2 | 72.4 | **68.1** | **58.2** |
| Gibbs PCFG,K=32 | 87.4 | 83.4 | 71.0 | 66.8 | 55.3 |

Table 2.4: F1 scores for experiments on sampled PCFGs. Note that Wang and Blunsom (2013) obtain an ENG F1-score of 77.9% using collapsed VB for K=2. Though they do not give exact numbers, their Fig. 7 indicates an F1-score of about 87% for K=16.

tions for sampling after 15 burning iterations for full data sets. And for small data sets, this work takes 100 sampling iterations after burning 100 iterations. Sampling a vanilla PCFG (K=1) produces results that are not state-of-the-art, but still good overall and always better than an unsmoothed PCFG. The benefits of the latent annotations are further shown in the increase of F1 score in all languages, as compared to the vanilla PCFG. Experiments were run up to K=32 primarily due to time constraint. Although previous literature results report increases up to the equivalent of K=64, it may be the case that higher K values with no merge step more easily lead to overfitting in this model – reducing the effectiveness of those high values, as shown by the overall poorer performance on several languages at K=32 when compared to K=16 as well as the general leveling-off seen at the high K values.

For English and Chinese, the previous Bayesian framework parsers outperform this work, but only by around two points. Additionally, parsing of Chinese in this work improves on the Bikel parser on the training data despite the fact that the Bikel parser makes use of language specific optimizations. The parser in this work needs no changes to switch languages.

The Gibbs PCFG with K=16 is superior to the Bikel and Berkeley Parser benchmarks for both KIN and MLG. This provides a promising result for future work on parsing low-resource languages in general. The parser in this work exhibits less variance than Berkeley Parser especially for KIN and MLG, which supports the fact that the variance of Berkeley Parser is higher for models with few subcategories (Petrov et al., 2006).

## 2.6 Conclusion

In this project, a Gibbs sampler is used to collect sampled trees theoretically distributed from the true posterior distribution in order to parse. Priors in a Bayesian model can control the sparsity of grammars (which the inside-outside algorithm fails to do), while naturally incorporating smoothing into the model (Johnson et al., 2007; Liang et al., 2009). This work also builds a Bayesian model for parsing with a treebank and incorporate information from training data as a prior. Moreover, this work extends the Gibbs sampler to learn and parse PCFGs with latent annotations.

The experiments demonstrate that sampling standard PCFGs, as well as PCFGs with latent annotations, is feasible with the use of a Gibbs sampler technique and produces results that are in line with previous parsers on controlled test sets. The results also show that the methods in this work are effective on a wide variety of languages with no language-specific model modifications needed, including two low-resource languages. Additionally, although not a uniform winner, the Gibbs-PCFG shows a propensity for performing well on naturally small corpora (here, KIN/MLG).

# Chapter 3

# Parse imputation for dependency annotations

Having established the Bayesian framework for parsing and its relative tolerance for low amounts of data, this chapter extends the model to dependency parsing. In this chapter, an algorithm is proposed for imputing missing dependencies from sentences that have been partially annotated, such that a standard dependency parser can then be trained on all annotations. This can be viewed as an extension of the results in the previous chapter. A dependency grammar can be viewed as an instance of a CFG. However, in this chapter, we consider receiving partial information on the tree structure.

## 3.1 Background

The parse trees discussed in Chapter 2 fall into a category known as constituency grammars. There is another way to describe sentence structure in natural language: by drawing links connecting individual words, which is called dependency grammar. Unlike constituency parsing, which focuses on identifying phrases and their recursive structure, dependency parsing focuses on relations between words. An example dependency representation of a sentence is shown in Figure 3.1, where a dependency link is an arrow pointing from head to dependent.

A *ROOT* symbol is automatically generated for each sentence, and it generates the head of the sentence. In this example, *barks* is the head of the sentence, which is generated from *ROOT* first. Then *barks* generates a left dependent *dog*, and *dog* has two left dependents.

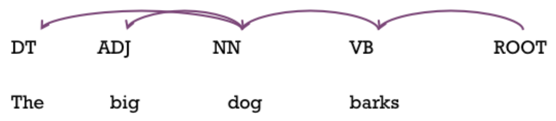


Figure 3.1: Dependency Structure of sentence *The big dog barks*

State-of-the-art unsupervised dependency parsers often require large amount of training data and/or additional prior knowledge. For instance, Naseem et al. (2010) reported an unsupervised approach which does not require dependency annotations, but makes use of the raw version of the full Penn Treebank. The approach presented by Marecek and Straka (2013) requires extra unlabeled texts to estimate parameters. The additional requirement of prior knowledge and training data becomes a problem for low-resource languages which do not have a clean, digitized corpus of sentences.

Supervised dependency parsers are expected to produce more accurate parsing results than unsupervised ones, as linguistically annotated data provides additional information. However, in most annotation projects, the availability of fully annotated data is limited. It is important to make efficient use of available annotators, particularly in the early stage of the annotation project when the total available corpus is small.

Parsers with minimal amounts of supervision or expert knowledge, or "weak supervision", have been proven as an effective approach to improve the insufficient accuracy of unsupervised parsers. For example, Naseem et al. (2010) demonstrated parsers with a set of universal dependency rules and showed substantial gains over unsupervised methods in many languages. Spitkovsky et al. (2010, 2011) used web mark-up and punctuation as additional annotations to improve parser performance.

Graph Fragment Language (GFL) has been developed as a light-weight dependency annotation scheme (Schneider et al., 2013; Chris and Smith, 2014). It allows relatively inexperienced annotators to partially annotate a sentence that they are unsure of. By allowing annotators to specify the major structure of a sentence without specifying lower-level structure, it is possible to develop a computational parser targeting annotations that maximize helpful information.

This work takes advantage of this fact and develops a two-stage system for parsing. The first stage takes partial GFL annotations as input, and outputs complete parse trees – in effect filling in the missing dependency information. This algorithm is based on a Gibbs sampling approach from Johnson et al. (2007) and further developed by Sun et al. (2014). The GFL annotations constrain the tree sampling space by using both dependencies and the constituent boundaries they express. Thus, the system is essentially performing missing dependency arc imputation using Gibbs sampling – refer to this approach as the Gibbs Parse Completer (GPC). The second stage uses the full dependencies output by the GPC to train Turbo Parser (Martins et al., 2010), and evaluation is done with this trained model on unseen sentences.

The two-stage system has the benefit of needing a small amount of data and a part-of-speech tagger. The system requires only a rather small number of sentences, less than one hundred, and relies on no outside tools or corpora. The part-of-speech tagger used in this system is also constructible with just two hours of annotation time (Garrette et al., 2013). Experimental results show that completing the partial annotations is able to provide gains of up to ten points of Unlabeled Attachment Score, and that two hours of annotation effort is in fact a sufficient amount of time to construct a useful dependency parser.

The main contributions of this work are: 1) an algorithm for imputing missing dependency information from partial annotations; 2) a two-stage system for generating dependency structures for unlabeled sentences – where in the first stage, complete annotations for a partially annotated corpus are created, and in the second stage the resulting fully annotated corpus and standard training methods are used for a dependency parser. Results show this two-stage system outperforms a system that only uses completely labeled examples. Moreover, the experiments show that remarkably small amounts of data can, with this label completion method, rival much larger training data sets. This work is especially applicable to low-resource languages.

This work is published in Mielens et al. (2015). The main contributions of this thesis are the models and algorithms presented. The main contribution of the coauthor, Jason Mielens, is data acquisition and performing experiments.

## 3.2 Data

Four languages from three language families are used. The data comes from English (ENG ), Chinese (CHI ), Portuguese (POR ), and Kinyarwanda (KIN ). They are used in experiments to verify the cross-linguistic applicability of this system, accounting for variations in linguistic properties. They also helps to realistically simulate a real-world, low-resource environment.

For ENG and CHI we collect data from pre-existing treebanks. For ENG the Penn Treebank (Marcus et al., 1993), converted into dependencies by the standard process, is used. Section 23 is used as a test set, and a random sample of sentences from sections 02-21 are selected for annotation with GFL as described below and subsequently used as the minimal training set. For CHI the Chinese Treebank (CTB5) (Xue et al., 2005) is used, also converted to dependencies. The testing set consists of files 1-40/900-931, and the sentences presented for GFL annotation are randomly sampled from files 81-899.

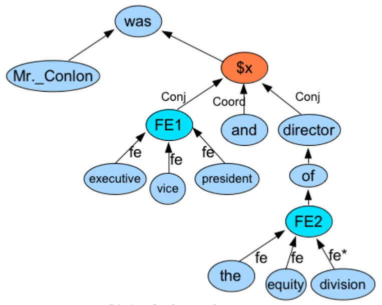
The POR data is from the CoNLL-X Shared Task on Multilingual Dependency Parsing and is derived from the Bosque portion of the Floresta sintá(c)tica corpus (Afonso et al., 2002), using the standard provided splits for training and testing. The KIN data is a corpus consisting of transcripts of testimonies by survivors of the Rwandan genocide, provided by the Kigali Genocide Memorial Center – this data is described by Garrette et al. (2013).

### 3.2.1 GFL Annotation

This work uses a small number of sentences annotated using the Graph Fragment Language (GFL), a simple ASCII markup language for dependency grammar (Schneider et al., 2013). Unlike traditional syntactic annotation strategies requiring trained annotators and great effort, rapid GFL annotations can be collected from

```
[Mr. Conlon] > was < $x
$x :: {(executive vice president) director} :: {and}
(the equity division*) > of > director
```

(a) GFL annotation



(b) Analysis graph

Figure 3.2: GFL example for *Mr. Conlon was executive vice president and director of the equity division*.

annotators who have minimal training. Kong et al. (2014) demonstrate the feasibility of training a dependency parser based on a GFL-annotated corpus of English tweets.

An example of GFL is shown in Figure 3.2: (a) is the GFL markup itself and (b) is a graphical representation of the dependencies it encodes. Figure 3.2 specifies several dependencies: *of* is a dependent of *director*; *(executive vice president)* and *director* are conjuncts and *and* is the coordinator. However, complete internal structure of the phrase *the equity division* remains unspecified, other than *division* being marked as the head (via an asterisk). The graphical representation shows both of these, *the equity division** and *(executive vice president)*, as FE nodes, for *fudge expression*, indicating they are grouped together but otherwise underspecified. Finally, *Mr. Conlon* in square brackets indicates it is a multiword expression.

This work takes advantage of underspecified sentences, while other work might fail to. For example, Kong et al. (2014) stipulate that the GFL annotations in their corpus must be fully-specified. They are thus unable to make use of such underspecified sentences, and we address that limitation here. From the GFL annotations we can extract and deduce dependency arcs and constraints (see Section 3.3.2 for full details) in order to guide the Gibbs sampling process.

### 3.2.2 Time-bounded annotation

As described in Section 3.1, a primary goal of this work was to consider the time in which a useful number of dependency tree annotations might be collected, such as might be required during the initial phase of a language documentation project or corpus build. To this end the annotators were operating under a strict two hour time limit. Two further hours for English were also collected.

| | CHI | ENG | KIN | POR |
|---|---|---|---|---|
| Sentences Annotated | 24 | 34 | 69 | 63 |
| Tokens Annotated | 820 | 798 | 988 | 1067 |
| Fully Specified Sentences | 4 | 15 | 31 | 20 |

Table 3.1: Two Hour GFL Annotation Statistics

The annotators were instructed to annotate as many sentences as possible in the two hours, and that they should liberally use underspecification, especially for particularly difficult sequences in a given sentence. This was done to facilitate the availability of partial annotations for experimentation. All of the annotators had some previous experience providing GFL annotations, so no training period was needed. Annotation was done in 30-minute blocks, to provide short breaks for the annotators and so that learning curves could be generated. Each language was annotated by a single annotator. The ENG and CHI annotators were native speakers of their annotation language, while the POR and KIN annotators were non-native though proficient speakers.

Table 3.1 shows the size of the GFL corpora that were created. Typically over 50% of the sentences were not fully specified. The partial annotations provided in these are useless to Turbo Parser unless the missing dependencies are imputed.

## 3.3 Gibbs Parse Completer (GPC)

### 3.3.1 Gibbs sampler for CFG-DMV Model

**CFG-DMV model** One of the most popular unsupervised dependency models is Dependency Model with Valence (DMV) by Klein and Manning (2004). This model generates one sentence by generating the head of the sentence first, and then each head recursively generates its left and right dependents. Previous research shows that dependency parses can be mapped to context free grammar (CFG) derivations, using a split-head construction (Eisner and Satta, 1999; Blunsom and Cohn, 2010; Johnson, 2007). This work uses this particular variety of CFGs for the DMV model (CFG-DMV), which enables the use of a Gibbs sampler algorithm for estimating PCFGs (Johnson et al., 2007; Sun et al., 2014).

Denote the input corpus as $\boldsymbol{\omega} = (\boldsymbol{\omega}^{(1)}, \cdots, \boldsymbol{\omega}^{(n)})$, where each $\boldsymbol{\omega}^{(s)}$ is a sentence consisting of words and in a sentence $\boldsymbol{\omega}$, word $\omega_i$ has an corresponding part-of-speech tag $\tau_i$. The split-head construction represents each terminal, part-of-speech $\tau_i$ in this work, by two unique terminals $\tau_{i,L}, \tau_{i,R}$ in the CFG parse. Henceforth, the yield of sentence $\boldsymbol{\omega}$ in this split-head CFG parse is $\tau_{1,l}, \tau_{1,r}, \cdots, \tau_{m,l}, \tau_{m,r}$, where $m$ is the length of original sentence $\boldsymbol{\omega}$ (e.g., the terminals for *the dog walks* are $DT_L$ $DT_R$ $N_L$ $N_R$ $V_L$ $V_R$). Denote this yield as $\boldsymbol{w} = w_{0,l}$, $l$ is the length of the terminals.

Denote the set of all words as $V_\omega$ and the set of all parts-of-speech as $V_\tau$. Define the following context-free grammar rules to generate a dependency parse tree. Note that these rules are for $\forall H \in V_\tau$, that is, for each part-of-speech there is an instance of that rule. We show a parsing of "The big dog barks" using some of these rules in Figure 3.3.

- $S \rightarrow Y_H$ used to generate the head $H$ of the sentence.
- $Y_H \rightarrow L_H R_H$ shows that $Y_H$ splits words into left and right parts.
- $L_H \rightarrow H_L$ means $H$ has no left dependents and linking to terminal $H_L$; $L_H \rightarrow L_H^1$ means $H$ has at least one left dependents.
- $L'_H \rightarrow H_L$ means $H$ stops generating left dependents and linking to terminal $H_L$; $L'_H \rightarrow L_H^1$ meaning $H$ has another left dependent.
- $L_H^1 \rightarrow Y_A L'_H$ means that $A$ is a left dependent of $H, \forall A \in V_\tau$.
- $R_H \rightarrow H_R$ means $H$ has no right dependents and linking to terminal $H_R$; $R_H \rightarrow R_H^1$ means $H$ has at least one right dependents.
- $R'_H \rightarrow H_R$ means $H$ stops generating right dependents and linking to terminal $H_R$; $R'_H \rightarrow R_H^1$ meaning $H$ has another right dependent.
- $R_H^1 \rightarrow R'_H Y_A$ means that $A$ is a right dependent of $H, \forall A \in V_\tau$.

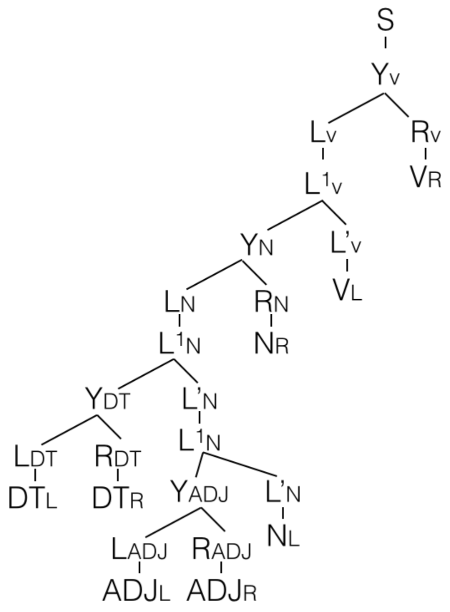


Figure 3.3: DMV-CFG parse of "The big dog barks."

**Gibbs sampler** The split-head representation encodes dependencies as a CFG. This enables the use of a Gibbs sampler algorithm for estimating PCFGs (Johnson et al., 2007; Sun et al., 2014). To incorporate constraints from partial annotations into this sampler, the tree-sampling step is modified to incorporate constraints derived from GFL annotations and thereby impute the missing dependencies.

```
Require: $A$ is parent node of binary rule; $w_{i,k}$ is a valid span of terminals and $i+1<k$
  function TreeSampler($A, i, k$)
      for $i<j<k$ and pair of child nodes of $A$:$B, C$ do
          $P(j,B,C) = \frac{\theta^{w}_{A\rightarrow BC} c(i,j) c(j,k) \cdot p_{B,i,j} \cdot p_{C,j,k}}{p_{A,i,k}}$
      end for
  Sample $j*, B*, C*$ from multinomial distribution for $(j,B,C)$ with probabilities calculated above
      return $j*, B*, C*$
  end function
```

Algorithm 4: Sampling split position and rule to expand parent node

Given a string $\boldsymbol{w} = (w_1, \cdots w_l)$, define a span of $\boldsymbol{w}$ as $w_{i,k} = (w_{i+1}, \cdots, w_k)$, so that $\boldsymbol{w} = w_{0,l}$. Pereira and Schabes (1992) firstly introduces an inside-outside algorithm to learn rule probabilities with partially bracketed corpus. A bracketing $\mathcal{B}$ of a sentence $\boldsymbol{w}$ is a finite set of spans on $\boldsymbol{w}$ satisfying the requirement that no two spans in a bracketing may overlap unless one span contains the other. For each sentence $\boldsymbol{w} = w_{0,l}$ Pereira and Schabes (1992) defines the auxiliary function for each span $w_{i,j}$, $0 \leq i < j \leq l$:

$$c(i,j) = \begin{cases} 1 & \text{if span } w_{i,j} \text{ is valid for } \mathcal{B}; \\ 0 & \text{otherwise.} \end{cases}$$

Here one span is valid for $\mathcal{B}$ if it doesn't cross any brackets. This algorithm uses the same idea that incorporating information of constraints by introducing the auxiliary function. Section 3.3.2 describes how to derive bracketing information from GFL annotations and how to determine if a span $w_{i,j}$ is valid or not. Note that for parsing a corpus without any annotations and constraints, $c(i,j) = 1$ for any span, and the algorithm is equivalent to the Gibbs sampler in Chapter 2.

There are two parts to the tree-sampling. The first constructs an inside table as in the Inside-Outside algorithm for PCFGs and the second selects the tree by recursively sampling productions from top to bottom. Consider a sentence $\boldsymbol{w}$, with sub-spans $w_{i,k} = (w_{i+1}, \cdots, w_k)$. Given $\theta^w$ (modified rule probabilities $\theta$ given constraints of sentence $\boldsymbol{w}$, see Section 3.3.2), construct the inside table with entries $p_{A,i,k}$ for each nonterminal and each span $w_{i,k}$: $0 \leq i < k \leq l$. Introduce $c(i,j)$ into the calculation of inside probabilities:

$$p_{A,i,k} = c(i,k) \cdot \sum_{A \to BC \in R} \sum_{i<j<k} \theta^w_{A \to BC} \cdot p_{B,i,j} \cdot p_{C,j,k} \tag{3.1}$$

Here, $p_{A,i,k}$ is the probability that terminals $i$ through $k$ were produced by the non-terminal $A$. Probability $\theta_{A \to BC \in R}$ is the probability of the rule to expand $A$ and $B, C$ are two child nodes of $A$ in that rule. The inside table is computed recursively using (3.1).

The resulting inside probabilities are then used to generate trees from the distribution of all valid trees of the sentence. The tree is generated from top to bottom recursively with the function $TreeSampler$ defined in Algorithm 4, which introduces $c(i, j)$ into the sampling function.

```
Require: $Arcs$ is the set of all directed arcs extracted from annotation for sentence $\boldsymbol{w}$
  function RULEPROB-SENT($\boldsymbol{w}, \theta, Arcs$)
      $\theta^w = \theta$
      for each directed arc $w_i < w_j$ do
          if $i < j$ then
              for nonterminal A $\neq L_{\tau_j}$ do
                  $\theta^w_{A \to \beta} = 0$ if $\beta$ contains $Y_{\tau_i}$
              end for
          else
              for nonterminal A $\neq R_{\tau_j}$ do
                  $\theta^w_{A \to \beta} = 0$ if $\beta$ contains $Y_{\tau_i}$
              end for
          end if
      end for
      return $\theta^w$
  end function
```

Algorithm 5: Modifying Rule Probabilities for $w$ to ensure parse tree contains all directed arcs. The algorithm essentially forces all rules where $j$ has a parent other than $i$ to have zero probability.

### 3.3.2 Constraints derived from GFL

This work exploits one dependency constraint and two constituency constraints from partial GFL annotations. Partial annotations produce constraints on the tree sampling procedure. Some of the partial annotations produce constraints in the inside probabilities construction step – specifically through defining valid spans, $c(i, j)$. Other partial annotations give arcs, which force a particular sampling path in producing a tree.

**Dependency Rule** Directed arcs are indicated with angle brackets pointing from the dependent to its head, e.g. *black* > *cat*. Once a directed arc is annotated, say word $j$ is modifying word $i$ ($\omega_i < \omega_j$), if $i$ is greater than $j$, which means word $i$ has a left child, the correct parse tree must contain rule $L^1_{\tau_i} \rightarrow Y_{\tau_j} L'_{\tau_i}$, where $\tau_i, \tau_j$ are part-of-speech tags of word $i, j$ (similarly if $i$ is smaller than $j$, the parse tree needs to contain $R^1_{\tau_i} \rightarrow R'_{\tau_i} Y_{\tau_j}$). This is enforced by modifying the rule probabilities for sample sentence $\boldsymbol{w}$ to ensure that any sampled tree contains all specified arcs, see Algorithm 5.

**Brackets** GFL allows annotators to group words with parenthesis, which provides an explicit indicator of constituent brackets. Even when there are not many annotations indicating the full internal structure(e.g. *(the equity division*) in Figure 3.2 (a)*, the head is usually marked with *, and this can be used to infer subconstituents. Given such a set of parentheses and the words inside them, brackets can be generated over the split-head representations of their parts-of-speech, based on possible positions of the head. Figure 3.4 shows how to generate brackets for three situations: the head is the leftmost word, rightmost word, or is in a medial position. For example, the first annotation indicates that *under* is the head of *under the agreement*, and the rest of words are right descendants of *under*. This leads to the bracketing shown over the split-heads.

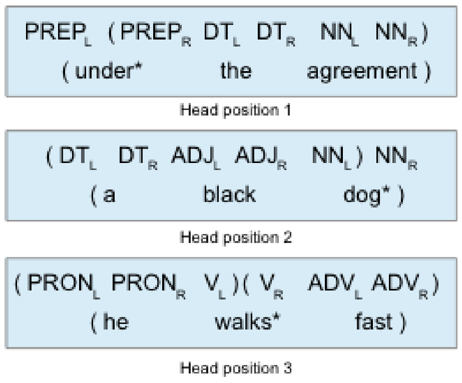

PREP_L ( PREP_R DT_L DT_R NN_L NN_R )
( under* the agreement )
Head position 1
( DT_L DT_R ADJ_L ADJ_R NN_L ) NN_R
( a black dog* )
Head position 2
( PRON_L PRON_R V_L )( V_R ADV_L ADV_R )
( he walks* fast )
Head position 3

Figure 3.4: Generating brackets for known head

**Half Brackets** One-sided half brackets can also be derived from dependency arcs by assuming that dependencies are projective. For example, in Figure 3.5, the annotation $a > dog$ specifies that *dog* has a left child *a*, which indicates that there is a right bracket before the right-head of *dog*. Thus, invalid spans can be detected using the half brackets; if a span starts after *a* and ends after *dog*, this span is invalid because it would result in crossing brackets. This half bracketing is a unique advantage provided by the split-head representation. The details of this algorithm are shown in Algorithm 6.

We use both half bracket and full bracket information, $\mathcal{B}$, to determine whether a span is valid. We set $c(i, j) = 0$ for all spans over $\boldsymbol{w}$ detected by Al-

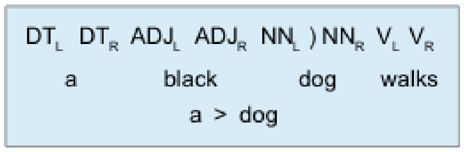


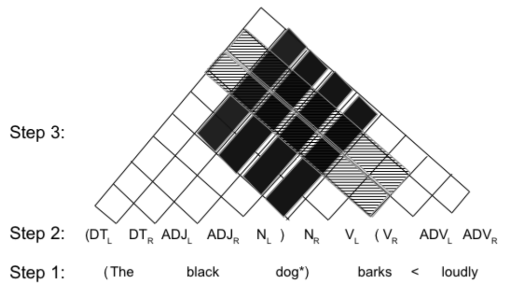
Figure 3.5: Generating half brackets

```
Require: $Arcs$ is the set of all directed arcs extracted for sentence, $w_{a,b}$ is a span to detect
  function DetectInvalidSpan($a, b, Arcs$)
    for each directed arc $\omega_i < \omega_j$ do
      if $i < j$ then
        if $a < 2i - 1 < b < 2j$ then
          $c(a, b) = 0$
        end if
      else
        if $2j - 2 < a < 2i - 1 < b$ then
          $c(i, j) = 0$
        end if
      end if
    end for
    return $c(a, b)$
  end function
```

Algorithm 6: Detect whether one span is invalid given all directed arcs.

gorithm 6 and violating $\mathcal{B}$. Then, in the sampling scheme, we'll only sample parse trees that satisfy these underlying constraints.

Figure 3.6 shows the resulting blocked out spans in the chart based on both types of brackets for the given partial annotation, which is Step 1 of the process. *The black dog* is a constituent with *dog* marked as its head, so we generate a full bracket over the terminal string in Step 2. Also, *barks* has a right child *loudly*; this generates a half bracket before $V_R$. In Step 3, the chart in Figure 3.6 represents all spans over terminal symbols. The cells in black are invalid spans based on the full



Step 3:
Step 2: (DT$_L$ DT$_R$ ADJ$_L$ ADJ$_R$ N$_L$ ) N$_R$ V$_L$ ( V$_R$ ADV$_L$ ADV$_R$
Step 1: ( The black dog*) barks < loudly

Figure 3.6: Process of generating brackets and detecting invalid spans

bracket, and the hatched cells are invalid spans based on the half bracket.

## 3.4 Results

There are two points of variation to consider in empirical evaluations of this approach. The first is the effectiveness of the GPC in imputing missing dependencies and the second is the effectiveness of the GFL annotations themselves. Of particular note with respect to the latter is the reasonable likelihood of divergence between the annotator and the corpus used for evaluation—for example, how coordination is handled and whether subordinate verbs are dependents or heads of auxiliary verbs. To this end, this work performs simulation experiments that remove increasing portions of gold dependencies from a training corpus to understand imputation performance and annotation experiments to evaluate the entire pipeline in a realistically constrained annotation effort.

In that regard, one thing to consider are the part-of-speech tags used by the unlexicalized GPC. These do not come for free, so rather than ask annotators to provide part-of-speech tags, the raw sentences to be annotated were tagged. In the case of English and Kinyarwanda, taggers trained with resources built in under two hours (Garrette et al., 2013) are used, so these results are actually constrained to the GFL annotation time plus two hours. Such taggers were not available for Chinese or Portuguese, so the Stanford tagger (Toutanova et al., 2003) was used for these.

After imputing missing dependencies, the GFL-GPC outputs a fully resolved set of dependencies that are in turn used to train TurboParser (Martins et al., 2010). In all cases, the experiments compare out approach to a right-branching baseline (RB), which always takes the first word as head of sentence, then generates the next right word as dependent until generating the last word. For the GFL annotation experiments, two additional baselines are used. The first is simply to use the sentences with full annotations and drop any incomplete ones (GFL-DROP). The second is to make any partial annotations usable by assuming a right-branching completion (GFL-RBC).

### 3.4.1 Simulated Partial Annotations

Figure 3.7 shows the learning curve with respect to number of annotated tokens when retaining 100%, 75%, and 25% of gold-standard training dependencies and using the GFL-GPC to impute the removed ones. A supervised dependency parser, Turbo Parser, then is used to train on these full dependencies, and evaluation is done on unseen sentences. The performance is evaluated by unlabeled attachment score (UAS), which is the percentage of words that have the correct head. Figure 3.7 demonstrates the degradation of performance by the GFL-GPC: the curve for a given removal proportion tracks with the curve for the full data with a more or less constant penalty paid for being provided less human guidance.

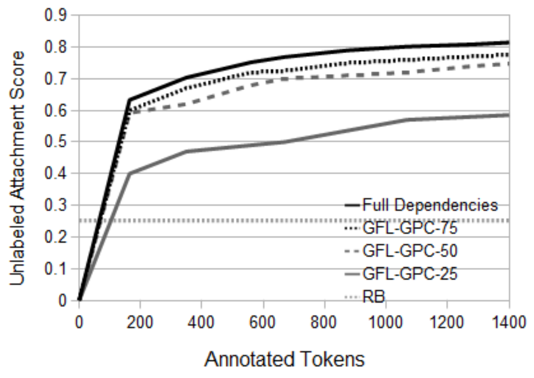


Figure 3.7: English Oracle and Degradation Results

Table 3.2 shows the unlabeled attachment scores obtained for English, Chinese, and Portuguese with varying proportions of dependencies removed for the GFL-GPC to impute. Note that these are based on the same sentences used in the GFL annotation experiments for each language discussed in the next experiment. Similar patterns are seen across languages: degradation as dependencies are removed.

| Language | ENG | CHI | POR |
|---|---|---|---|
| RB | 25.0 | 11.6 | 27.0 |
| GFL-GPC-25 | 58.7 | 33.5 | 60.2 |
| GFL-GPC-50 | 75.0 | 46.1 | 71.4 |
| GFL-GPC-75 | 77.8 | 50.1 | 73.7 |
| Full | 81.6 | 56.2 | 78.1 |

Table 3.2: Results with simulated partial annotations, GFL-GPC-X indicates X percent of dependencies were retained.

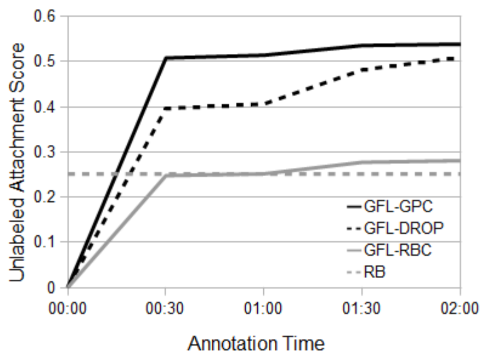
Additionally, the simulations of degraded data indicate that, given an equivalent number of total annotated arcs, running the GFL-GPC is more beneficial than requiring annotators to fully specify annotations. In other words, imputing fifty percent of the dependency arcs from sentences containing 1000 tokens is typically more effective by a few points than using the full gold-standard arcs from sentences containing 500 tokens. This simulation leaves out consideration of the time and effort required to actually obtain those full gold-standard arcs, which would be considerable relative to the partial annotations.

### 3.4.2 GFL Annotations

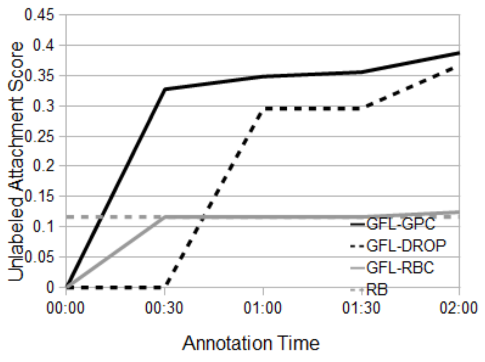
This work conducted three sets of experiments on the GFL annotations. The experiments are evaluated on sentences of all lengths, less than 10 words, and less than 20 words for all languages. This was done to determine the types of sentences that this method works best on and to compare to previous work that evaluates on sentences of different lengths.

The data in Table 3.3 shows how our results on ENG compare to results from the literature. Blunsom and Cohn (2010) was selected for their state of the art unsupervised result on all lengths, while Naseem et al. (2010) was chosen as a previous weakly-supervised approach. The GFL-GPC achieves similar results on the 'all lengths' criterion as the state of the art unsupervised result and substantially outperforms the previous weakly-supervised approach on sentences less than 20 words.

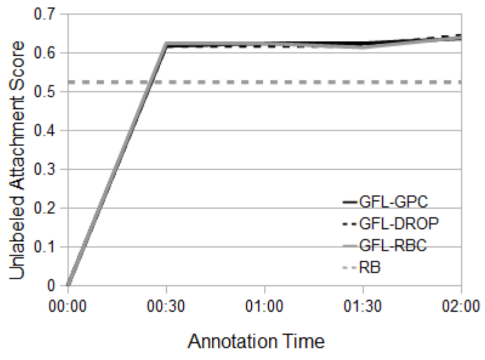
Poor performance on short sentences of this work is slightly surprising, and may result from an uneven length distribution in the sentences selected for annotation, as discussed by Spitkovsky et al. (2010). To correct this problem, both long



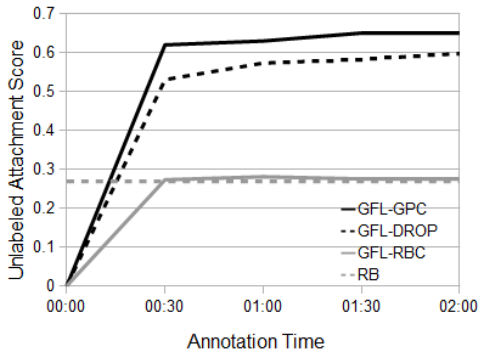
(a) English



(b) Chinese



(c) Kinyarwanda



(d) Portuguese

Figure 3.8: GPC results by annotation time for evaluation sentences of all lengths.

and short sentences should be included to construct a more representative sample for annotation.

In practice, GFL-RBC performs very similarly to RB. The relatively large number of under-specified sentences may have led to the right-branching quality of GFL-RBC dominating, rather the more informed GFL-based annotations.

The results of the ENG annotation session can be seen in Figure 3.8a. The

| Eval Length | < 10 | < 20 | all |
|---|---|---|---|
| GFL-DROP (4hr) | 54.5 | 55.0 | 52.6 |
| GFL-GPC (4hr) | 60.1 | **61.8** | 55.1 |
| Blunsom and Cohn, 2010 | 67.7 | – | **55.7** |
| Naseem et al., 2010 | **71.9** | 50.4 | – |

Table 3.3: English results compared to previous unsupervised and weakly-supervised methods

GFL-GPC is quite strong even at thirty minutes, with only seven sentences annotated. The GFL-DROP approaches the GFL-GPC towards two hours. This is likely explained by the fact that the end block contained many short sentences, meaning there was suddenly more fully-specified GFL annotations available. The learning curves for the other languages can be seen in Figures 3.8b-3.8d, with a summary available in Table 3.4.

Comparing the CHI curves to ENG shows that both languages demonstrate similar results. Of particular note is that the CHI annotations contained many fewer fully-completed sentences than the ENG annotations. Thus, the GFL-GPC was called upon to do more work in the case of CHI but it still managed to improve on the baseline of simply taking the fully-specified sentences (GFL-DROP in the figures).

The KIN results in Figure 3.8c exhibit a pattern unlike the other languages; specifically, the KIN data has a very high right-branching baseline (RB in figures) and responds nearly identically for all of the more informed methods. Upon investigation, this appears to be an artifact of the data used in KIN evaluation and perhaps some domain transfer issues. The gold data consists of transcribed natural speech, whereas the training data consists of sentences extracted from the Kinyarwanda

Wikipedia located at http://rw.wikipedia.org.

The POR data in Figure 3.8d shows a similar pattern to the ENG and CHI results, with the GFL-GPC once again improving on the raw GFL annotations despite an even greater number of partial annotations to handle (see Table 3.1).

| Language | KIN | CHI | POR |
|---|---|---|---|
| RB | 52.6 | 11.6 | 27.0 |
| GFL-DROP (2hr) | 64.4 | 36.7 | 59.8 |
| GFL-GPC (2hr) | 64.5 | 38.8 | 65.0 |

Table 3.4: Non-English results summary

All of the results sets display a large initial jump after the first round of annotations. This is encouraging for approaches that use annotated sentences: just a small number of examples provide tremendous benefit, regardless of the strategy employed.

## 3.5 Conclusion

This work has described a modeling strategy that takes advantage of a Gibbs sampling algorithm for CFG parsing plus constraints obtained from partial annotations to build dependency parsers. This strategy's performance improves on that of a parser built only on the available complete annotations. In doing so, the approach in this work supports annotation efforts that use GFL to obtain guidance from non-expert human annotators and allow any annotator to put in less effort than they would to do complete annotations.

Our algorithms enable a remarkably small amount of supervised data to rival existing unsupervised methods. While unsupervised methods have been considered an attractive option for low-resource parsing, they typically rely on large quantities of clean, raw sentences. The method in this work uses less than one hundred sentences, so in a truly low-resource scenario, in has the potential to require much less total effort. For instance, a single native speaker could easily both generate and annotate the sentences required for our method in a few hours, while the many thousands of raw sentences needed for state-of-the-art unsupervised methods could take much longer to assemble if there is no existing corpus. This also means the method in this work would be useful for getting in-domain training data for domain adaptation for parsers.

Finally, the method in this work has the ability to encode both universal grammar and test-language grammar as a prior. This would be done by replacing the uniform prior used in the work with a prior favoring those grammar rules during the updating-rule-probabilities phase of the GPC. This essentially has the effect of weighting those grammar rules.

# Chapter 4

# Bayesian Factor Model for mixed-type outcomes

In this chapter, we describe a factor model for uncovering latent factors among mixed-type outcomes. The proposed factor model is capable of discovering correlations among many types of variables, including binary, count, and continuous. We create an algorithm to inference the model parameters, and demonstrate that the algorithm successfully recovers correlation structure on simulated data. The algorithm also provides valuable insights from political science data. Finally, we implement the algorithm as an R package, which enables R users to perform factor analysis of data with mixed-types in a fully automatic way.

## 4.1 Introduction

Quantitative variables can be classed into several different types. In this chapter, we consider three types in specific: continuous data, binary data, and count data. Data of many variables with mixed types is common. For example, a heath survey may contain: a) binary outcomes: patient gender, currently suffering from diabetes, etc. b) continuous outcomes: patient age, body weight, etc. c) count outcomes: number of cigarettes consumed per day. We think of a data set as consisting of rows, for example each row containing the answers for one patient. We call a single column of the data a *feature*.

Factor analysis has been widely used to uncover underlying latent factors and capture patterns of association among features. However, most research focuses on models and methods applied to features with single type. For example, Shi and Lee (1998) applied Bayesian estimation in order to perform factor analysis with continuous observations. Richard Hahn et al. (2012) built a Gaussian/probit model, a Gaussian factor model embedded inside a multivariate probit model for analyzing binary observations. Zhou et al. (2012a) introduced Poisson Factor Analysis (PFA) for exploring correlation among features with count data through factor analysis.

Several works have dealt with factor models with mixed-type features. Quinn (2004) developed a Gaussian/probit model for continuous and ordinal data. The model relates the data with mixed-type to underlying continuous variables, and impose a Gaussian factor model on the latent variables. For ordinal data, the model relates it to underlying continuous variables through a probit model. However, this model cannot directly applied for count data. For example, word count features in a corpus, ranging from zero to several hundred, often have many empty count categories. It is not proper for the model to treat such count data as ordinal categorical data. Murray et al. (2013) extends the Gaussian/probit model. Instead of modeling feature with specific distribution, Murray et al. (2013) used the extended rank likelihood (Hoff, 2007) for modeling features with different types. Basically, they are using the inverse empirical cumulative density function to relate all features to an underlying continuous scale, and build factor model on it. However, since this model only use partial order information embedded in the data, which makes the model unable to estimate parameters characterizing the properties of individual outcomes.

The work in this chapter provides a framework for exploring association among features with mixed-types. In this work, a logistic model applied to link discrete observations to latent continuous variables. A Gaussian factor model is further built on the these latent continuous variables. Furthermore, we describe a Gibbs sampler for inferencing all parameters in a closed form as in Gaussian/probit model. The framework and the algorithm also works for probit link with minimum tuning.

The main contributions of this chapter are: 1) a flexible framework for uncovering underlying latent factors and exploring interdependency among features with mixed-types; 2) a Gibbs sampler in closed form for inferencing parameters in this framework; and 3) an R package developed for analysis. This work provides a factor model for mixed-type data by allowing for logit link functions linking binary and count observations to latent variables, and allowing for count data as opposed to solely categorical data. Furthermore, we provide the first Gibbs sampler for factor models of mixed types, which is based on a key insight of using Pòlya-Gamma distributions.

## 4.2 A Factor Model for Mixed Data

The main purpose of this section is to present a framework that is able to perform factor analysis on mixed types: continuous, binary and count. For ease of presentation, we first present the model as it would appear constrained to each of the three types. This allows us to generalize to mixed types of data.

### 4.2.1 Factor model for continuous outcomes

Bayesian factor analysis for continuous data has a significant amount of previous literature (Geweke and Zhou, 1996; Aguilar and West, 1998). In this section, we present these existing models and their applications to capturing patterns of association among features. And the ability of summarizing multivariate observations using a lower-dimensional variable makes factor models useful for data reduction.

Suppose we have $n$ continuous observations, denoted by $\{\boldsymbol{y_i^c}, i = 1, \cdots, n\}$, where $\boldsymbol{y_i^c}$ is a $p^c$-vector, and $p^c$ is the number of continuous features in each observation. In other words, the data set has $n$ rows and $p^c$ columns. We assume that these $n$ observations on $p^c$ related features are randomly sampled from a multivariate normal distribution denoted by $N(\boldsymbol{\alpha}^c, \boldsymbol{\Omega})$, where $\boldsymbol{\Omega}$ denotes an $p^c \times p^c$ non-singular covariance matrix. Our goal is to explore the covariance matrix $\boldsymbol{\Omega}$, and one popular approach is by imposing a factor structure on $\boldsymbol{\Omega}$ that $\boldsymbol{\Omega} = \boldsymbol{B}^c\boldsymbol{B}^{cT} + \Sigma$, where $\boldsymbol{\Sigma}$ is a $p^c$-by-$p^c$ diagonal matrix with non-negative elements and $rank(\boldsymbol{B}) = k < p^c$. This factor structure can be rewritten in a standard Gaussian $k$-factor model by introducing latent factors.

The standard Gaussian $k$-factor model relates each observation $\boldsymbol{y_i^c}$ to the $k$ common factors $\boldsymbol{f_i}$, an underlying $k$-vector of random variables. The model is given by

$$\boldsymbol{y_i^c} = \boldsymbol{\alpha^c} + \boldsymbol{B^c f_i} + \boldsymbol{\epsilon_i}$$

where $p^c$-vector $\boldsymbol{\alpha^c}$ is intercept; $\boldsymbol{B^c}$ is a $p^c \times k$ matrix of factor loadings, and $k < p^c$ is a specified positive integer; the factors $\boldsymbol{f_i}$ are independent with $\boldsymbol{f_i} \sim N(\boldsymbol{0}, \boldsymbol{I_k})$; and $\boldsymbol{\epsilon_i} \sim N(\boldsymbol{0}, \boldsymbol{\Sigma})$ are idiosyncratic noise with $\boldsymbol{\Sigma} = diag(\sigma_1^2, \cdots, \sigma_{p^c}^2)$. Marginalizing out the latent variables, we have $\boldsymbol{y_i^c} \sim N(\boldsymbol{\alpha^c}, \boldsymbol{B^c}\boldsymbol{B^{c^T}} + \Sigma)$. The model relates the common structure in $\boldsymbol{y_i^c}$ to underlying factors, and isolates variation that is purely idiosyncratic in the $\boldsymbol{\epsilon_i}$ terms. In this way, the common factors explain all the dependence structure among the $p^c$ continuous features.

It is useful to re-write this standard form of a continuous factor model in an alternate form. This alternate form allows us to relate the continuous factor models to factor models for discrete outcomes. To do this, we introduce a latent continuous quantities $\boldsymbol{z_i^c}$ such that:

$$\boldsymbol{z_i^c} = \boldsymbol{\alpha^c} + \boldsymbol{B^c}\boldsymbol{f_i}$$

for $i = 1, \cdots, n$. We then write $\boldsymbol{y_i^c} \sim N(\boldsymbol{z_i^c}, \boldsymbol{\Sigma})$. In this way, we can think of the factors in the factor model as determining the parameters of the distribution of $\boldsymbol{y_i^c}$. We will use the same idea for other types of data – specifically the factors will determine the parameters of the data's distribution.

Estimating the covariance structure of $\boldsymbol{y_i^c}$ directly requires estimating approximately $p^c(p^c - 1)/2$ parameters. On the other hand, for a factor model with $k < p^c$ factors, we only need to estimate approximately $p^c \times k$ terms, primarily the loading matrix $\boldsymbol{B^c}$. The difference in the number of parameters can be great, especially for practical problems where $k \ll p^c$.

### 4.2.2 Factor model for binary outcomes

In this section, we present a factor model for strictly binary features. The model assumes that for each feature the observations are following a Bernoulli distribution, and a factor structure is further imposed on the log-odds of Bernoulli distributions for all features to capture the association among them. We also show how our model relates to Gaussian/probit model.

Suppose we have $n$ binary observations, denoted by $\{\boldsymbol{y_i^b}, i = 1, \cdots, n\}$, where $\boldsymbol{y_i^b}$ is a $p^b$-vector, and $p^b$ is the number of binary features in each observation. In this model, we assume that $y_{ij}^b$ is distributed Bernoulli with probability of success $p_{ij}$. The $p_{ij}$ are related to a set of unobserved continuous quantities $\boldsymbol{z_i^b} = (z_{i,1}^b, \cdots, z_{i,p^b}^b)$ via a logit link function:

$$p_{ij} = \frac{exp(z_{ij}^b)}{1 + exp(z_{ij}^b)}$$

We further relate the unobserved continuous quantities $\boldsymbol{z_i^b}$ to underlying factors $\boldsymbol{f_i}$. We suppose that $\boldsymbol{z_i^b} = \boldsymbol{\alpha^b} + \boldsymbol{B^b f_i}$ with $\boldsymbol{\alpha^b}$ is a $p^b$-vector intercept term, $\boldsymbol{B^b}$ is a $p^b \times k$ loading matrix and the factors $\boldsymbol{f_i}$ are independently distributed with $\boldsymbol{f_i} \sim N(\boldsymbol{0}, \boldsymbol{I_k})$.

Our model can be turned into a Gaussian/probit factor model by linking $p_{ij}$ to $z_{ij}^b$ through a probit link function: $p_{ij} = \Phi(z_{ij}^b)$, where $\Phi(\cdot)$ denotes the standard Normal cumulative distribution function. Gaussian/probit factor model is widely used in analyzing the underlying correlation structure of binary features (Richard Hahn et al., 2012). We show below that our model can be thought of as a variant of these pre-existing factor models, except with a logit link.

**Relation to Gaussian/probit model** To further show the relationship between our model and previous Gaussian/probit model, we can do several steps of re-writing. First, we present a latent-variable model. Then we show this formulation is equivalent to using a logit link function, highlighting the connection to the existing Gaussian/probit models.

We introduce a continuous latent variable $y^*_{ij}$ for all $i, j$ for this alternative formulation. We then link $y^*_{ij}$ to $z^b_{ij}$ by:

$$y^*_{ij} = z^b_{ij} + \epsilon_{ij} \tag{4.1}$$

where $\epsilon_{ij}$ are independently distributed with a standard logistic distribution $\epsilon_{ij} \sim Logistic(0, 1)$. We relate $y^*_{ij}$ to observation $y^b_{ij}$ via:

$$y^b_{ij} \mid y^*_{ij} = \begin{cases} 1, & \text{if } y^*_{ij} > 0 \\ 0, & \text{otherwise.} \end{cases} \tag{4.2}$$

Equations (4.1) and (4.2) present another formulation of factor model for binary features. When $\epsilon$ is following a standard normal distribution: $\epsilon \sim N(0, 1)$, this formulation is equivalent to Gaussian/probit model for binary features in Richard Hahn et al. (2012). Furthermore, this model can be considered as a variant of the model of Quinn (2004): fixing cut-point to be $0$ for ordinal data with two categories.

We show that this alternative formulation is equivalent to our model above. Two properties of standard logistic distribution are used: a) it is symmetric about $0$; and b) for $\epsilon$ following standard logistic distribution

$$Prob(\epsilon < x) = \frac{exp(x)}{1 + exp(x)}$$

With these facts, we have:

$$\begin{aligned}
Prob(y_{ij}^b = 1 \mid z_{ij}^b) &= Prob(y_{ij}^* > 0 \mid z_{ij}^b) \\
&= Prob(z_{ij}^b + \epsilon_{ij} > 0) \\
&= Prob(\epsilon_{ij} > -z_{ij}^b) \\
&= Prob(\epsilon_{ij} < z_{ij}^b) \\
&= \frac{exp(z_{ij}^b)}{1 + exp(z_{ij}^b)}
\end{aligned}$$

which is equivalent to the original model between $y_{ij}^b$ and $z_{ij}^b$.

Furthermore, we show that the correlation of $\boldsymbol{y_i^b}$ can be captured by estimating $cov(\boldsymbol{y_i^*})$, which is equivalent to estimating $cov(\boldsymbol{z_i^b})$. Scaling the latent quantities $\boldsymbol{y_i^*}$ preserves the distribution of $\boldsymbol{y_i^b}$, by (4.2). This implies that we can capture the correlation of $\boldsymbol{y_i^b}$ by estimating $cov(\boldsymbol{y_i^*})$. From (4.1), we have $cov(y_{is}^*, y_{it}^*) = cov(z_{is}^b + \epsilon_{is}, z_{it}^b + \epsilon_{it})$, where $0 < s, t \leq p^b$. So $cov(y_{is}^*, y_{it}^*) = cov(z_{is}^b, z_{it}^b)$ if $s \neq t$, and $cov(y_{is}^*, y_{it}^*) = cov(z_{is}^b, z_{it}^b) + cov(\epsilon_{is}, \epsilon_{it}) = cov(z_{is}^b, z_{it}^b) + \pi^2/3$ if $s = t$. So estimating $cov(\boldsymbol{y_i^*})$ can be achieved by estimating $cov(\boldsymbol{z_i^b})$.

### 4.2.3 Factor model for count data

The Poisson distribution $X \sim Pois(\lambda)$ is widely used for modeling count data. However, one property of Poisson distribution that its variance is equal to its mean makes it not well-suited in many data sets (Ventura et al., 2005). To relax this assumption, a negative binomial distribution is considered in our model. Suppose we have $n$ count observations, denoted by $\{\boldsymbol{y_i^d}, i = 1, \cdots, n\}$, where $\boldsymbol{y_i^d}$ is a $p^d$-vector, and $p^d$ is the number of count features in each observation. We relate observations $y_{ij}^d$, where $i = 1, \cdots, n$, $j = 1, \cdots, p^d$ to a continuous latent quantity $z_{ij}^d$ via:

$$y_{ij}^d \sim NB(h_j, p_{ij}^d), \tag{4.3}$$

where the negative binomial distribution for us is parameterized through $h_j$ failures, and probability of success $p_{ij}$. We further relate $p_{ij}$ to $z_{ij}^d$ via the logistic function:

$$p_{ij} = \frac{exp(z_{ij}^d)}{1 + exp(z_{ij}^d)} \tag{4.4}$$

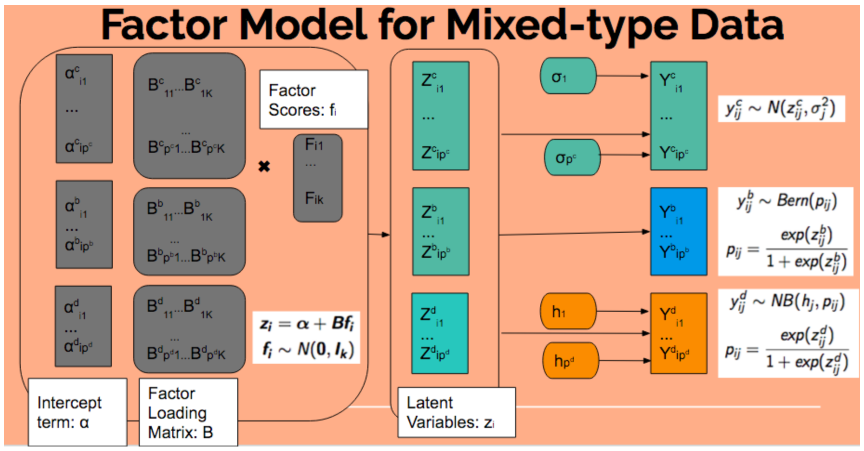
The parameter $h_j$ allows for over-dispersion compared to the Poisson, with the count $y_{ij}^d$ having a variance $h_j p_{ij}/(1 - p_{ij})^2$ larger than the mean $h_j p_{ij}/(1 - p_{ij})$.

Similar to standard factor model, we further relate the unobserved continuous quantities $\boldsymbol{z_i^d}$ to underlying factors $\boldsymbol{f_i}$. We suppose that $\boldsymbol{z_i^d} = \boldsymbol{\alpha^d} + \boldsymbol{B^d f_i}$ with intercept term $\boldsymbol{\alpha^d}$, a $p^d$-vector, a $p^d \times k$ loading matrix $\boldsymbol{B^d}$ and the factors $\boldsymbol{f_i}$ are independently distributed with $\boldsymbol{f_i} \sim N(\boldsymbol{0}, \boldsymbol{I_k})$.

The factor models for count features and binary features both relate the probabilities of success of the distributions to latent continuous quantities through a logit link. For both types of features, the same factor structures are built on the latent continuous quantities. The continuous quantities, the $z_{ij}^b$ and $z_{ij}^d$, express the log-odds of the corresponding Bernoulli distributions and Negative Binomial distributions. Understanding the covariance of the log-odds captures the association between the features.

The factor model for count features only imposes factor structures on the log-odds of Negative binomial distributions. Parameter $h_j$ only contributes the variation among $j$th feature, which is similar to $\sigma_j^2$ in factor model for continuous features. In other words, $h_j$ captures idiosyncratic variance associated with each feature. These parameters are also known as single factors in factor analysis.

### 4.2.4 Factor model for continuous, binary and count data



Factor Model for Mixed-type Data
Factor Scores: $f_i$
Intercept term: α
Factor Loading Matrix: B
Latent Variables: $z_i$
$z_i = \alpha + Bf_i$
$f_i \sim N(0, I_k)$
$y_{ij}^c \sim N(z_{ij}^c, \sigma_j^2)$
$y_{ij}^b \sim Bern(p_{ij})$
$p_{ij} = \frac{exp(z_{ij}^b)}{1+exp(z_{ij}^b)}$
$y_{ij}^d \sim NB(h_j, p_{ij})$
$p_{ij} = \frac{exp(z_{ij}^d)}{1+exp(z_{ij}^d)}$

Figure 4.1: Factor model for continuous, binary and count data

Now a factor model for mixed-type data is presented, see Figure 4.1. Here the data is collected with continuous features, binary features and count features. Suppose we have $n$ observations, denoted by $\{\boldsymbol{y_i}, i = 1, \cdots, n\}$, where $\boldsymbol{y_i}$ is a $(p^c + p^b + p^d)$-vector, which combines continuous, binary and count observations: $\boldsymbol{y_i^c}$, $\boldsymbol{y_i^b}$ and $\boldsymbol{y_i^d}$. We relate observations $\boldsymbol{y_i}$ to a set of continuous latent quantity $\boldsymbol{z_i}$. The latent quantities $\boldsymbol{z_i}$ is also a $(p^c + p^b + p^d)$-vector, and it combines three sets of latent quantities: $\boldsymbol{z_i^c}$, $\boldsymbol{z_i^b}$ and $\boldsymbol{z_i^d}$ which relates to $\boldsymbol{y_i^c}$, $\boldsymbol{y_i^b}$ and $\boldsymbol{y_i^d}$ through three classes of distributions: Gaussian distributions, Bernoulli distributions and Negative Binomial distributions respectively.

We further relate the unobserved continuous quantities $\boldsymbol{z_i}$ to underlying factors $\boldsymbol{f_i}$. We suppose that $\boldsymbol{z_i} = \boldsymbol{\alpha} + \boldsymbol{B}\boldsymbol{f_i}$ with intercept term $\boldsymbol{\alpha}$, a $p^c + p^b + p^d$-vector, a $(p^c + p^b + p^d) \times k$ loading matrix $\boldsymbol{B}$ and the factors $\boldsymbol{f_i}$ are independently distributed with $\boldsymbol{f_i} \sim N(\boldsymbol{0}, \boldsymbol{I_k})$. The loading matrix $\boldsymbol{B}$ can be partitioned into three blocks via:

$$\boldsymbol{B} = \begin{bmatrix} \boldsymbol{B^c} \\ \boldsymbol{B^b} \\ \boldsymbol{B^d} \end{bmatrix}$$

The loading matrix $\boldsymbol{B}$ must be further constrained to ensure that $cov(\boldsymbol{z_i}) = \boldsymbol{B}\boldsymbol{B^T}$ has a unique solution in $\boldsymbol{B}$. Considering $\boldsymbol{B}^* = \boldsymbol{B}\boldsymbol{P^T}$ and $\boldsymbol{f_i^*} = \boldsymbol{P}\boldsymbol{f_i}$, where $\boldsymbol{P}$ is any orthogonal $k \times k$ matrix, then $\boldsymbol{B}^*$ is also a solution to $cov(\boldsymbol{z_i})$. We adopt the approach of Geweke and Zhou (1996) here, to constrain $\boldsymbol{B}$ to be zero for upper-triangular entries $\{b_{js} = 0 : j < s, 1 \leq s \leq k\}$ and positive along the diagonal $\{b_{ss} > 0 : 1 \leq s \leq k\}$.

### 4.2.5 Related Work

In this section, a semiparametric latent variable model for mixed outcomes is described (Murray et al., 2013). The model is using the inverse empirical cumulative density function to relate all features to an underlying continuous scale, and build factor model on it. We would like to briefly introduce the model, then compare our model to their approach.

Murray et al. (2013) developed a Bayesian factor model for mixed data. They also relate all observations to corresponding underlying continuous variables. In their model, let $y_{ij}$ denote the observed $j$th feature of observation $i$ with marginal distribution $F_j$ for all $i = 1, \cdots, n$, $j = 1, \cdots p$, then $y_{ij}$ can be represented with respective to latent variable $z_{ij}$ as $y_{ij} = F_j^{-1}[\Psi(z_{ij})]$, where $\Psi(\cdot)$ denotes the normal

CDF and $z_{ij}$ is distributed standard normal. Let $\boldsymbol{z_i}$ denote the $i$th row of matrix $\boldsymbol{Z}$. The Gaussian copula model assumes that:

$$\boldsymbol{z_1}, \cdots, \boldsymbol{z_n} \mid \boldsymbol{C} \sim i.i.d. N(\boldsymbol{0}, \boldsymbol{C})$$
$$y_{ij} = F_j^{-1}[\Psi(z_{ij})]$$

where $\boldsymbol{C}$ is a $p$-by-$p$ correlation matrix.

Murray et al. (2013) uses the extended rank likelihood (Hoff, 2007) for modeling marginal distributions $F_1, \cdots, F_p$. The extended rank likelihood depends only on the ranks of the observations, which means that $y_{ij} < y_{i'j}$ implies $z_{ij} < z_{i'j}$. Therefore the model has $\boldsymbol{Z} \in D(\boldsymbol{Y})$, where

$$D(\boldsymbol{Y}) = \{\boldsymbol{Z} \in \Re^{n\times p} : max_k\{z_{kj} : y_{kj} < y_{ij}\} < z_{ij} < min_k\{z_{kj} : y_{ij} < y_{kj}\}\forall i, j\}$$

And a factor structure similar to ours is imposed on the latent matrix $\boldsymbol{Z}$ that $\boldsymbol{z_i} = \boldsymbol{\alpha} + \boldsymbol{B}\boldsymbol{f_i}$.

Compared to their approach, instead of using extended rank likelihood for modeling marginal distributions, our model specifies marginal distributions for different data type. We use Gaussian distribution to model continuous feature, Bernoulli distribution to model binary feature, and Negative binomial distribution to model count feature. All these distributions are widely used in modeling corresponding data type in regression models.

There are some disadvantages of using extended rank likelihood (ERL) for modeling discrete features. Firstly, using only partial order information cannot model the exact nature of binary and count features. ERL approach may not be useful for a practitioner who is primarily interested in parameters characterizing the properties of individual outcomes. However, in our model, the latent variables $z$'s are the log odds of Bernoulli distribution and Negative Binomial distribution for modeling binary observations and count observations. This makes our model more interpretable for analyzing data from real world. Secondly, ERL approach does not provide a model for the probability that two outcomes will be tied. ERL approach permits ties by considering the data to be only partially ordered, while it cannot properly model the conditional probability $Pr(y_{ij} = y_{i'j} \mid z_{ij}, z_{i'j})$ for two observations $i$ and $i'$ on $j$th feature having a tied response. Thirdly, as ERL is distribution-free, a practitioner cannot calculate the model evidence or other evaluation metrics based on it, which makes it hard to evaluate the model even with simulation data.

## 4.3 Bayesian model and inference

We write a complete model that allows for the specification of a Gibbs sampler. The model includes data, parameters, and prior distributions. Let $p = p^c + p^b + p^d$. The model can be thought of as having:

1. data $\boldsymbol{y} = \{\boldsymbol{y_i}, i = 1, \cdots, n\} \in \mathcal{R}^{n \times p}$, $\boldsymbol{y_i}$ is $i$th row of observation matrix $\boldsymbol{y}$

2. parameters $\sigma_j^2$ for $j \in \{1, \cdots, p^c\}$, associated to continuous features

3. parameters $h_j$ for $j \in \{1, \cdots, p^d\}$, associated to count features

4. parameter $\boldsymbol{F} = \{\boldsymbol{f_i}, i = 1, \cdots, n\} \in \mathcal{R}^{n \times k}$, representing the latent factors. Each $\boldsymbol{f_i}$ is $i$th row of matrix $\boldsymbol{F}$

5. parameter $\boldsymbol{\alpha} \in \mathcal{R}^p$, representing the intercept term

6. parameter $\boldsymbol{B} \in \mathcal{R}^{p \times k}$, representing the loading matrix

7. and the prior over the model parameters :

$$\sigma_j^2 \sim IG(\lambda_j, \lambda_j \tau_j/2) \qquad \text{where } j = 1, \cdots, p^c \tag{4.5a}$$

$$h_j \sim Gamma(a_0, 1/r_j) \qquad r_j \sim Gamma(u_0, 1/g_0) \qquad \text{where } j = 1, \cdots, p^d \tag{4.5b}$$

$$\boldsymbol{f_i} \sim N(\mathbf{0}, \boldsymbol{I_k}) \qquad \text{where } i = 1, \cdots, n \tag{4.5c}$$

$$\alpha_j \sim N(0, \nu_\alpha) \qquad \text{where } j = 1, \cdots, p \tag{4.5d}$$

$$b_{js} \sim N(0, \nu_s) \qquad \nu_s \sim IG(c_s, c_s d_s/2) \qquad \text{where } j = 1, \cdots, p,\ s = 1, \cdots, k \tag{4.5e}$$

and IG represents Inverse-Gamma distributions. These prior distributions for model parameters are designed for conjugacy to their likelihood, which allows for efficient inference. Our experiments was performed with hyperparameters $\lambda_j = 2$, $\tau_j = 1$ for all $j$; $a_0 = 0.01$, $u_0 = 0.01$, $g_0 = 0.01$ ; $\nu_\alpha = 10$; $c_s = 2$ and $d_s = 1$ for all $s$; and hyperparameters $\nu_s$ and $r_j$ inferred by the Gibbs sampler for all $s$ and $j$.

### 4.3.1 Inference

In this section, we introduce a Gibbs sampler used to estimate the model parameters. The sampler iteratively updates estimates for each parameter in Section 4.3 based on current values of all other parameters. Specifically, each parameter is drawn from its posterior distribution, conditioned on all other parameters. For parameters associated with continuous features, their conditional posterior distribution are well-defined distributions due to conjugacy. In other words, the Gibbs sampler can draw directly from those posterior distributions. For parameters associated with binary and count features, their conditional posterior distribution are not easy to draw from. We introduce Pòlya-Gamma latent variables to allow for conjugacy and easy Gibbs sampling of these parameters.

For ease of describing the Gibbs sampler, we first introduce some background knowledge. Section 4.3.1.1 summarizes key facts about Pòlya-Gamma distributions. This section also summarizes how to sample coefficients from conditional posterior for a linear regression model, which will be used in our Gibbs sampler. Section 4.3.1.2 shows how the Gibbs sampler works. And finally, Section 4.3.1.3 shows how to impose sparse prior for inferencing large datasets.

#### 4.3.1.1 Preliminaries

**Pòlya-Gamma distribution** To efficiently inference for binary and count data, we introduce the latent variables distributed with Pòlya-Gamma distribution. Polson et al. (2013) first introduced the Pòlya-Gamma distribution. The distribution has two parameters, we begin by specifying the distribution of $\omega \sim PG(b, 0)$, which is equal to an infinite sum of gammas:

$$\omega \overset{D}{=} \frac{1}{2\pi^2} \sum_{k=1}^{\infty} \frac{g_k}{(k-1/2)^2},$$

where each $g_k$ is an independent $Gamma(b, 1)$ random variable. The general $PG(b, c)$ class is constructed via exponential tilting of the $PG(b, 0)$ density:

$$p(\omega \mid b, c) \propto exp(-\frac{c^2}{2}\omega)p(\omega \mid b, 0) \tag{4.6}$$

Polson et al. (2013) further proved the following property:

$$L(\psi) = \frac{\{\exp(\psi)\}^a}{\{1+\exp(\psi)\}^b} \propto e^{\kappa\psi} \int_0^{\infty} e^{-\omega\psi^2/2}\, p(\omega)\, d\omega \tag{4.7}$$

$$= e^{\kappa\psi} E_\omega[exp(-\omega\psi^2/2)] \tag{4.8}$$

where $\kappa = a - b/2$ and $p(\omega) = PG(\omega \mid b, 0)$.

There are some interesting facts arising from the formulas above:

1. From (4.8), we have, conditional upon $\omega$:

$$L(\psi) \propto e^{\kappa\psi} exp(-\omega\psi^2/2) \tag{4.9}$$

$$\propto exp\{-\frac{\omega}{2}(\frac{\kappa}{\omega} - \psi)^2\} \tag{4.10}$$

   Notice (4.10) is exactly the likelihood of a Gaussian distribution with $\kappa/\omega$ as the random variable, $\psi$ as the mean and $\omega^{-1}$ as the variance. In other words, we have:

$$(\kappa/\omega \mid \psi, \omega) \sim N(\psi, \omega^{-1}) \tag{4.11}$$

2. Conditioning $\omega$ on $\psi$ again gives a Pòlya-Gamma distribution as follows:

$$p(\omega \mid \psi) \propto p(\psi \mid \omega)p(\omega)$$

$$\propto exp(-\omega\psi^2/2)p(\omega \mid b, 0) \tag{4.12}$$

$$= p(\omega \mid b, \psi) \tag{4.13}$$

   Here for (4.12) we use the fact in (4.9) that $p(\psi \mid \omega) \propto exp(-\omega\psi^2/2)$; and for (4.13) we use the definition of general case of Pòlya-Gamma distribution, (4.6).

3. Both Bernoulli and negative binomial likelihoods of logistic parameters can be written in the form of the left-hand-side of (4.7). If we denote $p = \frac{exp(\psi)}{1+exp(\psi)}$ then the Bernoulli likelihood is given by $p(x \mid p) = \frac{exp(\psi)^x}{1+exp(\psi)}$, which matches (4.7) with $a = x$, and $b = 1$. This allows us to rewrite (4.11) and (4.13) with respect to Bernoulli likelihood as:

$$((2x-1)/2\omega \mid \psi, \omega) \sim N(\psi, \omega^{-1}) \tag{4.14}$$

$$p(\omega \mid \psi) \sim PG(1, \psi) \tag{4.15}$$

   Similarly, the likelihood of negative binomial distribution can be written as

$$p(x \mid h, p) \propto \frac{exp(\psi)^x}{(1+exp(\psi))^{x+h}},$$

   which matches (4.7) with $a = x$, and $b = x + h$. This allows us to rewrite (4.11) and (4.13) with respect to Negative Binomial likelihood as:

$$((x-h)/2\omega \mid \psi, \omega) \sim N(\psi, \omega^{-1}) \tag{4.16}$$

$$p(\omega \mid \psi) \sim PG(x+h, \psi) \tag{4.17}$$

There are two reasons for introducing Pòlya-Gamma latent variable $\omega$ into inference. The construction of Pòlya-Gamma distribution allows us to easily update $\omega$ with Pòlya-Gamma prior in a Gibbs sampler, as described in Section 4.3.1.2. Also (4.11) indicates that conditional on $\psi$ and $\omega$, the parameters of both the Bernoulli and negative binomial can be written in a Gaussian form. This property allows us to efficiently inference models, and the details are shown in Section 4.3.1.2.

**Inference coefficients in a linear regression model** Suppose we have working responses $z$, design matrix $\boldsymbol{X}$, and coefficients $\boldsymbol{\beta}$ and diagonal covariance matrix $\Sigma$. They form a standard Gaussian linear model:

$$z \sim N(\boldsymbol{X\beta}, \Sigma) \tag{4.18}$$

with Gaussian prior on $\boldsymbol{\beta}$ that

$$\boldsymbol{\beta} \sim N(\boldsymbol{m_0}, \boldsymbol{V_0})$$

Then to update coefficients $\boldsymbol{\beta}$ in Gibbs sampler, the conditional posterior for $\boldsymbol{\beta}$ (Koop et al., 2007, pp 108,192) is :

$$(\boldsymbol{\beta} \mid \boldsymbol{z}, \boldsymbol{X}, \Sigma) \sim N(\boldsymbol{m}, \boldsymbol{V})$$

where

$$\boldsymbol{V} = (\boldsymbol{X^T \Sigma^{-1} X} + \boldsymbol{V_0^{-1}})^{-1} \tag{4.19a}$$

$$\boldsymbol{m} = \boldsymbol{V}(\boldsymbol{X^T}\Sigma^{-1}\boldsymbol{z} + \boldsymbol{V_0^{-1} m}) \tag{4.19b}$$

#### 4.3.1.2 Gibbs sampler for inferencing parameters

In this section, a Gibbs sampler (Geman and Geman, 1984) is presented to draw correlated samples from the joint posterior distribution of all parameters in (4.5). The notation $(\boldsymbol{Y} \mid -)$ refers to the full conditional distribution of a random variable $\boldsymbol{Y}$ conditional on everything else. Besides the parameters in (4.5), we introduce new latent variables $\boldsymbol{\omega^b_{ij}}, i = 1, \cdots, n, j = 1, \cdots, p^b$ and $\boldsymbol{\omega^d_{ij}}, i = 1, \cdots, n, j = 1, \cdots, p^d$. And the priors for $\omega^b_{ij}$ and $\omega^d_{ij}$ are:

$$\omega^b_{ij} \sim PG(1, 0) \tag{4.20}$$

$$\omega^d_{ij} \sim PG(y^d_{ij} + h_j, 0) \tag{4.21}$$

A single iteration of the Gibbs sampler performs the following steps:

1. Calculate the latent continuous quantities $\boldsymbol{z_i} = \boldsymbol{\alpha} + \boldsymbol{B}\boldsymbol{f_i}$ for $i = 1, \cdots, n$
2. Update $\omega^b_{ij}$ and $\omega^d_{it}$ for $i = 1, \cdots, n$, $j = 1, \cdots, p^b$ and $t = 1, \cdots, p^d$ with:

   $$(\omega^b_{ij} \mid -) \propto PG(\omega^b_{ij} \mid 1, z^b_{ij})$$
   $$(\omega^d_{it} \mid -) \propto PG(\omega^d_{it} \mid y^d_{it} + h_t, z^d_{it})$$

   As shown in preliminaries, every term of the likelihood for binary features can be written in the form of the left-hand-side of (4.7), with $a = y^b_{ij}$, $b = 1$, $\psi = z^b_{ij}$; and similarly with $a = y^d_{it}$, $b = y^d_{it} + h_t$ and $\psi = z^d_{it}$ for count features. Replacing corresponding parameters in (4.15) and (4.17) leads us to updating $\omega$.

3. Update $\sigma_j^2$ for $j = 1, \cdots, p^c$ with

$$\sigma_j^2 \sim IG((\lambda_j + n)/2, (\lambda_j \tau_j + (\boldsymbol{y_j^c} - \boldsymbol{z_j^c})^t (\boldsymbol{y_j^c} - \boldsymbol{z_j^c}))/2) \tag{4.22}$$

   The conditional posterior of $\sigma_j^2$ is:

$$\begin{aligned}
(\sigma_j^2 \mid -) &\propto p(\sigma_j^2 \mid \lambda_j, \tau_j, \boldsymbol{y_j^c}, \boldsymbol{z_j^c}) \propto p(\sigma_j^2, \lambda_j, \tau_j, \boldsymbol{y_j^c}, \boldsymbol{z_j^c}) \\
&\propto p(\boldsymbol{y_j^c} \mid -) p(\sigma_j^2 \mid \boldsymbol{z_j^c}, \lambda_j, \tau_j) p(\boldsymbol{z_j^c}, \lambda_, \tau_j) \\
&\propto p(\boldsymbol{y_j^c} \mid -) p(\sigma_j^2 \mid \lambda_j, \tau_j) \\
&\propto IG((\lambda_j + n)/2, (\lambda_j \tau_j + (\boldsymbol{y_j^c} - \boldsymbol{z_j^c})^t (\boldsymbol{y_j^c} - \boldsymbol{z_j^c}))/2),
\end{aligned}$$

   where $\boldsymbol{y_j^c}$ and $\boldsymbol{z_j^c}$ are $j$th column of $\boldsymbol{y^c}$ and $\boldsymbol{z^c}$. The third step uses the fact that $\boldsymbol{z_j^c}$ is independent of $\sigma_j$ and $\boldsymbol{z_j^c}$, $\lambda_j$, $\tau_j$ are considered as known. The last step uses the fact that Gaussian distribution is conjugate to Inverse-Gamma distribution, which allows to draw $\sigma_j^2$ from an updated Inverse Gamma distribution (Koop et al., 2007, pp 17).

4. Update $(h_j, r_j)$ for $j = 1, \cdots, p^d$. Zhou et al. (2012b) provides a solution to inference $h_j$ as well as $r_j$ based on compound-Poisson augmentation of the negative binomial distribution. Their main result shows that with we could update $h_j$ by drawing from an updated Gamma distribution with prior shown in (4.5b).

5. Update $\boldsymbol{f_i}$ for $i = 1, \cdots, n$ with:

$$(\boldsymbol{f_i} \mid -) \sim N(\boldsymbol{m}, \boldsymbol{V}) \tag{4.23}$$

where

$$\boldsymbol{V^{-1}} = \boldsymbol{B^T \Omega_i B} + \boldsymbol{I_k} \tag{4.24}$$

$$\boldsymbol{m} = \boldsymbol{V}(\boldsymbol{B^T x_i} - \boldsymbol{\alpha}) \tag{4.25}$$

Here we define a $n \times p$ matrix $\Omega^{-1}$ with the $i$th row given by the quantities $\boldsymbol{\omega_i^{-1}} = (\sigma_1^2, \cdots, \sigma_{p^c}^2, \omega_{i1}^{b^{-1}}, \cdots, \omega_{ip^b}^{b^{-1}}, \omega_{i1}^{d^{-1}}, \cdots, \omega_{ip^d}^{d^{-1}})$. Furthermore, we define a diagonal matrix in (4.24) as $\Omega_i^{-1} = diag(\boldsymbol{\omega_i^{-1}})$. We also define a $p \times k$ matrix $\boldsymbol{X}$ that could be partitioned to $3 \times 1$ blocks that:

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{X^c} \\ \boldsymbol{X^b} \\ \boldsymbol{X^d} \end{bmatrix}$$

where $\boldsymbol{X^c}$ is a matrix with entries $x_{ij}^c = y_{ij}^c$ for $i = 1, \cdots, n$ and $j = 1, \cdots, p^c$; $\boldsymbol{X^b}$ is a matrix with entries $x_{ij}^b = \frac{2y_{ij}^b - 1}{2\omega_{ij}^b}$ for $i = 1, \cdots, n$ and $j = 1, \cdots, p^b$; and $\boldsymbol{X^d}$ is a matrix with entries $x_{ij}^d = \frac{y_{ij}^d - h_j}{2\omega_{ij}^d}$ for $i = 1, \cdots, n$ and $j = 1, \cdots, p^d$.

Recall that conditional on $\omega$, we have Gaussian relationship for binary features as in (4.14), and for count features as in (4.16). Replacing corresponding parameters, we have:

$$x_{ij}^b \sim N(z_{ij}^b, \omega_{ij}^{b^{-1}}) \tag{4.26}$$

$$x_{ij}^d \sim N(z_{ij}^d, \omega_{ij}^{d^{-1}}) \tag{4.27}$$

Combining (4.26), (4.27) and $x_{ij}^c = y_{ij}^c \sim N(z_{ij}^c, \sigma_j^2)$, we write them in a multivariate form that for $i = 1, \cdots, n$:

$$\boldsymbol{x_i} \sim N(\boldsymbol{\alpha} + \boldsymbol{B f_i}, \Omega_i^{-1}) \tag{4.28}$$

where $\boldsymbol{x_i}$ is $i$th row of matrix $\boldsymbol{X}$. This can be considered as a standard linear model with observations $\boldsymbol{x_i} - \boldsymbol{\alpha}$, design matrix $\boldsymbol{B}$ and covariance matrix $\Omega_i^{-1}$. And the regressor $\boldsymbol{f_i}$ has a Gaussian prior: $\boldsymbol{f_i} \sim N(\boldsymbol{0}, \boldsymbol{I_k})$. As shown in (4.19), the conditional posterior distribution for $\boldsymbol{f_i}$ is also Gaussian, and can be updated as shown above.

6. Update $\alpha_j$ for $j = 1, \cdots, p^c$ with:

$$(\alpha_j \mid -) \sim N(m_\alpha, V_\alpha) \tag{4.29}$$

where

$$V_\alpha = (\nu_\alpha^{-1} + \sum_i \Omega_{ij})^{-1} \tag{4.30}$$

$$m_\alpha = V_\alpha \sum_i \Omega_{ij}(x_{ij} - \boldsymbol{B_j f_i^T}) \tag{4.31}$$

We rewrite (4.28) column-wisely for $j = 1, \cdots, p$:

$$\boldsymbol{x_j} \sim N(\boldsymbol{1} \cdot \alpha_j + \boldsymbol{F B_j^T}, \Omega_j^{-1}) \tag{4.32}$$

where $\boldsymbol{x_j}$ is the $j$th column of matrix $\boldsymbol{X}$; $\boldsymbol{1}$ is a $n$-vector with all entries $1$; $\boldsymbol{B_j}$ is the $j$the row of matrix $\boldsymbol{B}$; and $\Omega_j^{-1} = diag(\boldsymbol{\omega_j^{-1}})$, $\boldsymbol{\omega_j^{-1}}$ is $j$th column of matrix $\Omega^{-1}$. This is a special case of Gaussian linear model (4.18) with $z = \boldsymbol{x_j} - \boldsymbol{F B_j^T}$, $\boldsymbol{X} = \boldsymbol{1}$ and $\Sigma = \Omega_j^{-1}$. And the prior for $\alpha_j$ is $\alpha_j \sim N(0, \nu_\alpha)$. This allows us to draw $\alpha_j$ from a Gaussian distribution as in (4.19). Notice here we write out the matrix multiplication into element-wise summation form in updates.

7. Update elements $b_{js}$ for $s = 1, \cdots, k$, $j = s, \cdots, p$ with:

$$(b_{js} \mid -) \sim N(m_b, V_b) \qquad b_{js} > 0 \qquad \text{if } j = s$$

$$(b_{js} \mid -) \sim N(m_b, V_b) \qquad \qquad \text{if } j > s$$

where

$$V_b = (\nu_s^{-1} + \sum_i \Omega_{ij} f_{is}^2)^{-1}$$
$$m_b = V_b \sum_i f_{is} \Omega_{ij} \tilde{x_{ij}}.$$

Here, we draw $b_{ss}$ from a truncated normal distribution with constraint that $b_{ss} > 0$ as mentioned in Section 4.2.4. And $\tilde{x_{ij}} = x_{ij} - \alpha_j - \sum_{t \neq s} b_{jt} f_{it}$.

Rewrite (4.18) for all $i = 1, \cdots, n$, $j = 1, \cdots, k$:

$$x_{ij} \sim N(\alpha_j + \boldsymbol{B}_j \boldsymbol{f_i}, \Omega_{ij}^{-1})$$
$$\equiv N(\alpha_j + \sum_t b_{jt} f_{it}, \Omega_{ij}^{-1})$$

Then for given $s$, $1 \leq s \leq k$, we have:

$$\tilde{x_{ij}} = (x_{ij} - \alpha_j - \sum_{t \neq s} b_{jt} f_{it}) \sim N(f_{is} b_{js}, \Omega_{ij}^{-1}) \tag{4.33}$$

Rewrite (4.33) in multivariate form, for given $j$, $s$, we have:

$$\tilde{\boldsymbol{x_i}} \sim N(\boldsymbol{f_s} b_{js}, \Omega_j^{-1}) \tag{4.34}$$

where $\tilde{\boldsymbol{x_i}} = (\tilde{x_{i1}}, \cdots, \tilde{x_{ip}})$; $\boldsymbol{f_s}$ is the $s$th column of matrix $\boldsymbol{F}$ and $\Omega_j^{-1} = diag(\boldsymbol{\omega_j^{-1}})$ with $\boldsymbol{\omega_j^{-1}}$ being the $j$th column of matrix $\Omega^{-1}$. This is a special case of Gaussian linear model (4.18) with $z = \tilde{\boldsymbol{x_i}}$, $\boldsymbol{X} = \boldsymbol{f_s}$ and $\Sigma = \Omega_j^{-1}$.

And the prior for $b_{js}$ is $b_{js} \sim N(0, \nu_s)$. This allows us to draw $b_{js}$ from a Gaussian distribution as in (4.19). Notice here we write out the matrix multiplication into element-wise summation form in updates.

8. Update hyperparameter $\nu_s$ for $s = 1, \cdots, k$ with

$$(\nu_s \mid -) \sim IG((c_s + n_s)/2, (c_s d_s + \boldsymbol{B_s}^T \boldsymbol{B_s})/2) \tag{4.35}$$

where $n_s$ be the number of unconstrained elements in $s$th column of $\boldsymbol{B}$, $\boldsymbol{B_s}$ for $s = 1, \cdots, k$. The prior for $\nu_s$ is Inverse-Gamma distribution, and the likelihood of $\nu_s$ is Gaussian, which makes conditional posterior distribution of $\nu_s$ is also Inverse-Gamma.

#### 4.3.1.3 Inferencing in large data sets

In this section, we introduce a sparse prior imposed on loading matrix, which permits some of the loadings to be exactly zero. The sparse prior has been used in previous factor models (Bernardo et al., 2003; Carvalho et al., 2008; Richard Hahn et al., 2012), and we adopt the approach of Richard Hahn et al. (2012) here. These models assume that each latent factor will be associated with only a small number of features. Introducing sparse prior can be considered as performing feature selection automatically when we estimate the loading matrix. This is useful when we have a large number of features and want a more parsimonious covariance structure.

The sparse prior we impose on the loading matrix $\boldsymbol{B}$ is:

$$\begin{aligned}(b_{js} \mid \nu_s, q_s) &\sim q_s \cdot N(0, \nu_s) + (1 - q_s)\delta_0(b_{js}) \\ \nu_s &\sim IG(c_s/2, c_s d_s/2) \\ q_s &\sim Beta(1, 1)\end{aligned}$$

where $\delta_0(b_{js})$ denotes a point-mass measure at $b_{js} = 0$, and $q_s$ is a hyperparameter so that large $q_s$ indicates a high likelihood for $b_{js}$ to be $0$.

We also need some modification in our Gibbs sampler for updating $b_{js}, \nu_s$, and an extra step for updating $q_s$. The steps are described in detail in (Richard Hahn et al., 2012), and we summarize here:

1. Update $b_{js}$ with

$$(b_{js} \mid -) \sim (1 - \hat{q}_{js})\delta_0 + \hat{q_{js}} N(m_b, V_b) \tag{4.36}$$

   where

$$\begin{aligned}V_b &= (\nu_s^{-1} + \sum_i \Omega_{ij} f_{is}^2)^{-1} \\ m_b &= V_b \sum_i \Omega_{ij} f_{s,i} \tilde{x_{ij}} \\ \hat{\rho}_{js} &= \frac{N(0 \mid 0, \nu_j)}{N(0 \mid m_b, V_b)} \\ \hat{q}_{js} &= \frac{\hat{\rho}_{js}}{\frac{1-q_s}{q_s} + \hat{\rho}_{js}}\end{aligned}$$

   Here $N(0 \mid y, z)$ is the value of pdf of normal distribution $N(y, z)$ at $0$.

2. Update $\nu_s$ as in (4.35) except that $n_s$ be the number of unconstrained elements in $\boldsymbol{B_s}$ currently set to non-zero.

3. Update $q_s$ with:

$$(q_s \mid -) \sim Beta(1 + n_s, 1 + \tilde{n_s} - n_s) \tag{4.37}$$

where $\tilde{n_s}$ is the number of unconstrained elements in $\boldsymbol{B_s}$.

## 4.4 Experiments and Results

To demonstrate the proposed factor model for mixed-type data can approximately re-construct the covariance structure underlying the observations, two experiments are conducted in this section. The first experiment is performed on simulation data, showing the behavior of the factor model under different parameters. The second experiment is performed on real-world data from political science and provides valuable insights for the data set.

### 4.4.1 Simulated Data

We simulate data with various combinations of $N/P/K$, where $N$ is the number of observations, i.e. number of rows of observation matrix; $P$ is the total number of mixed-type features, with $P/3$ count features, $P/3$ binary features and $P/3$ continuous features; and $K$ is the number of underlying factors. For each case, we simulate the intercept $\boldsymbol{\alpha}$, loading matrix $\boldsymbol{B}$ by drawing iid normal values. The factor scores $\boldsymbol{f_i}$ are sampled independently from $N(\boldsymbol{0}, \boldsymbol{I_K})$ for each observation. We than calculate the latent states $\boldsymbol{z_i}$ for each observation by $\boldsymbol{z_i} = \boldsymbol{\alpha} + \boldsymbol{B}\boldsymbol{f_i}$. As described in Section 4.2, we use the latent states as a mean to simulate continuous observations; use the latent states as log-odds to simulate binary observations; and simulate the count observations with the latent states as log-odds along with pre-set over-dispersion parameter $h = 1$.

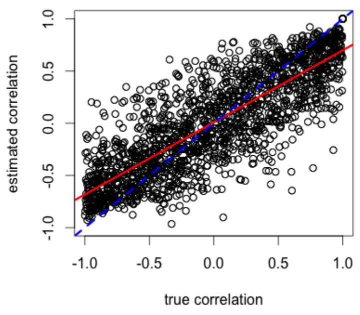
For each case, we use a 3-factor model, and Gibbs sampler for estimating the model parameters. In each case, the model was estimated using 2000 Gibbs iterations after a 500 iteration burn-in, keeping every 5th sample for a final sample size of 300. We use the mean of those $300$ samples to estimate the model parameters. We assess the performance of our model by computing root mean squared error between true latent states $\boldsymbol{z}$ and estimated latent states $\hat{\boldsymbol{z}}$: $[\frac{1}{N \cdot P} \sum_{1 \leq i \leq N, 1 \leq j \leq P} (z_{ij} - \hat{z_{ij}})^2]^{1/2}$.

Results are in Table 4.1. Given fixed number of latent factors in the true models, RMSE decreases when there are more features or more observations for each feature. This is expected, as more information is provided given the same complexity of underlying covariance structure. When $K$ is larger in the true model vs. the estimated model, comparing left and right columns of the table, we also get increasing RMSE.

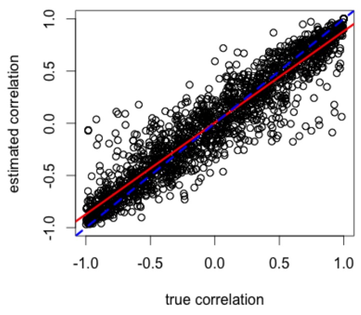
| N/P/K | RMSE for Fitted Model | N/P/K | RMSE for Fitted Model |
|---|---|---|---|
| 50/15/3 | 0.92 | 50/15/5 | 1.04 |
| 50/60/3 | 0.57 | 50/60/5 | 0.74 |
| 50/150/3 | 0.45 | 50/150/5 | 0.67 |
| 200/15/3 | 0.82 | 200/15/5 | 0.67 |
| 200/60/3 | 0.35 | 200/60/5 | 0.64 |
| 200/150/3 | 0.27 | 200/150/5 | 0.59 |

Table 4.1: Root Mean Squared Error (RMSE) on latent $\boldsymbol{z}$ values for various $N/P/K$ combinations. We always estimate a model with $K = 3$, and the $N/P/K$ above describe the model for generating the true data.

In some of these examples, $N/P/K = 50/60/3$ and $N/P/K = 50/150/3$, we have more features than observations. Estimating correlation structure in data sets like these is challenging (Berger and Sun, 2008). Our model, however, is able to estimate the correlation structure even with a small amount of observations. This is because of the factor-structure in the model we've created. Figure 4.2 shows that as we have more observation data ($N$), the estimation of correlation structure of features is improved. The figure shows true values of the correlation of $BB^T$ plotted against estimated values at different amounts of data.



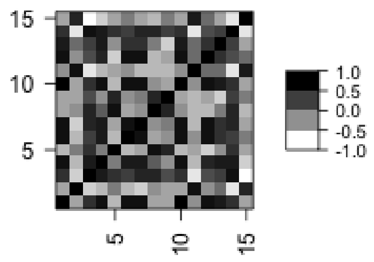
(a) N/P/K = 50/60/3



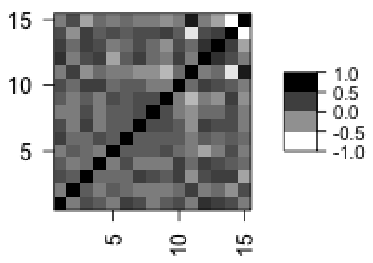
(b) N/P/K = 200/60/3

Figure 4.2: Plots of estimated correlations $cor(\hat{BB^T})$ versus true correlations $cor(BB^T)$. Red solid lines are regression lines.

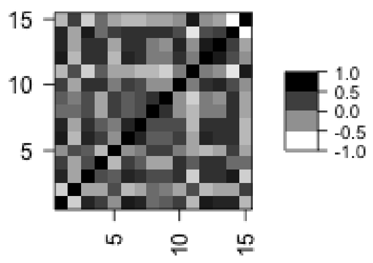
Figure 4.3 further shows that our model is able to re-construct the latent correlation among all features. Figure 4.3a shows the actual correlation among latent states that we want to reconstruct. Figure 4.3b shows reconstructing it by naive correlation of raw data. The correlation of the raw data in Figure 4.3b shows an attenuation effect compared to actual correlation as in Figure 4.3a. This effect is dramatic in count features and binary features because of the non-linear relationship between the observations and the latent states. However the correlation structure of the latent features is uncovered by our model, see Figure 4.3c. This improved performance is attributed to the parametrized model.



(a) Actual Correlation among latent states



(b) Raw Correlation from Data



(c) Estimated Correlation among latent states

Figure 4.3: Results for case $N/P/K = 50/15/3$. (a) shows the actual correlation among latent states (b)shows the raw correlation among features (c) shows the estimated correlation among latent states. Feature $1-5$ are count features, $6-10$ are binary features, and $11-15$ are count features.

This section shows via simulation that our factor model for mixed-type data is capable of re-constructing the covariance structure of the latent states. We perform various experiments on data simulated with different number of observations, features and latent factors. And we compare our estimated results to the true model parameters, and show that our model is able to re-construct the latent correlation among features with mixed-types.

### 4.4.2 Real data

In this section we perform factor analysis on real-world data with mixed-types. The data we use are 534 speeches in the House of Commons in 1866 about the Second Reform Act. Moser and Reeves (2014) describe the dataset in details. The data contains three types of features: a) 8467 count features - each one represents the number of occurrences of that word in one speech; 2) 4 binary features - "party2c" feature equal to 1 indicating the speaker is conservative; "type.county" feature and "type.university" feature are indicators for two constituency types; "miniTRUE" feature indicates the speaker is a minister or not; 3) 1 continuous feature - "malaportionment" feature is a score representing overrepresented if greater than 0 and underrepresented if smaller than 0.

We use a 10-factor model on the data set to see how our model discovers the underlying factors. We also use a sparse prior on the parameters of the loading matrix to make each observation depend on only a few factors. As a result, all the underlying correlation can be captured by estimated loading matrix and 10 factors.

The features associated with each latent factor have similar patterns of responses. As a result, features under the same latent factor are closely correlated and convey similar concepts. It is confirmed by the experimental results that features in the same factor have closely related to the factor. The first row of Table 4.2 shows the features having highest loadings in two sample factors out of the ten factors. Words with highest loadings in the first sample factor include names of foreign countries and cities, like "melbourn", "spain", "canada"; and words related to foreign policy, like "militari", "nonintervent". Thus we define this sample factor as *diplomatic*. The second factor talks about *political typology*, like “democraci”, “economi”, “societi”, “darwen”, “conserv”, etc.

Another analysis is to investigate features’ proportions of variance (PoV) explained in one factor. For one feature $i$, PoV explained by a certain factor $j$ is computed by its squared loading divided by the total squared loadings of the feature, i.e. $PoV_{ij} = b_{ij}^2/(\sum_{1\le s\le k} b_{is}^2)$. For one factor, features with high PoV explained by this factor specifically represent the concept of this factor. Here we are not interested in words with low total squared loading, i.e. words contributing little to common variance.

Features with high PoV can be different from features with high loadings in a factor. The second row of Table 4.2 shows the words having highest PoVs in two sample factors. For factor *diplomatic*, word “cowper” has the highest PoV, which means most of its variance is captured by factor *diplomatic*. The word “cowper” is most likely from Charles Cowper, who was the premier of New South Wales back to 1866. That makes it specifically related to factor *diplomatic*. For factor *political typology*, we also see words like “commonwealth”, “freedom” that specifically relate to the factor.

Our factor model allows for analysis of the complexity of semantics of words within the speeches. We define the dimension of a word as the number of non-zero loadings of the word. The dimension of a word represents the number of latent factors describing the word feature. Figure 4.4 shows the histogram of dimensions of 8467 words. Words with low dimension include: “like”, “good” , “most”, “more” , “everi” , “great” etc. High dimension words include “return”,

| Features | Sample Factor 1 (diplomatic) | Sample Factor 2 (political typology) |
|---|---|---|
| High loading | “hain” “melbourn” “girond” “nonintervent” “furious” “spain” “cricket” “victoria” “militari” “besot” “canada” “narrat” “juror” “franc” “poland” “type.university” “wallingford” “woodstock” | “democraci”,“economi”, “democrat”, “societi”, “prosper”, “upper”, “nation”, “mechan”, “fellow”, “darwen”, “conserv” “amongst” “cent” “workingclass” “trade” “genuin” “whig” “american” “class” “artizan” |
| High PoV | “cowper” “melbourn” “bandon” “besot” “enniskillen” “spain” “furious” “kinsal” “narrat” “cork” “girond” “odonoghu” “hain” “athlon” “michi” “sydney” “unhandsom” “stawel” “inver” “waterford” | “serv” “deci” “british” “nation” “popular” “suggest” “event” “import” “commenc” “institut” “greec” “generous” “commonwealth” “document” “freedom” “preced” “obstacl” “disposit” “anterior” “upper” “imperil” |

Table 4.2: Features with highest loading and highest PoV in two sample factors

“conserv”, “wednesburi”, “redeem”, etc.

A word with low dimension is essentially independent from any latent factors and other words. Knowing the occurrence rate of this word in a speech, would not tell you very much about the content of the speech. On the other hand, a word with a high dimension depends on many of the latent factors and is correlated with many other words. The number of occurrences of a high dimension word, would tell you more about the content of the speech. This is potentially useful for feature selection in text classification. One could throw out the zero-dimension words, because they have very low correlations with other words.

As a summary, we analyze the speeches in the House of Commons in 1866 about the Second Reform Act with proposed factor model for mixed-type data. The

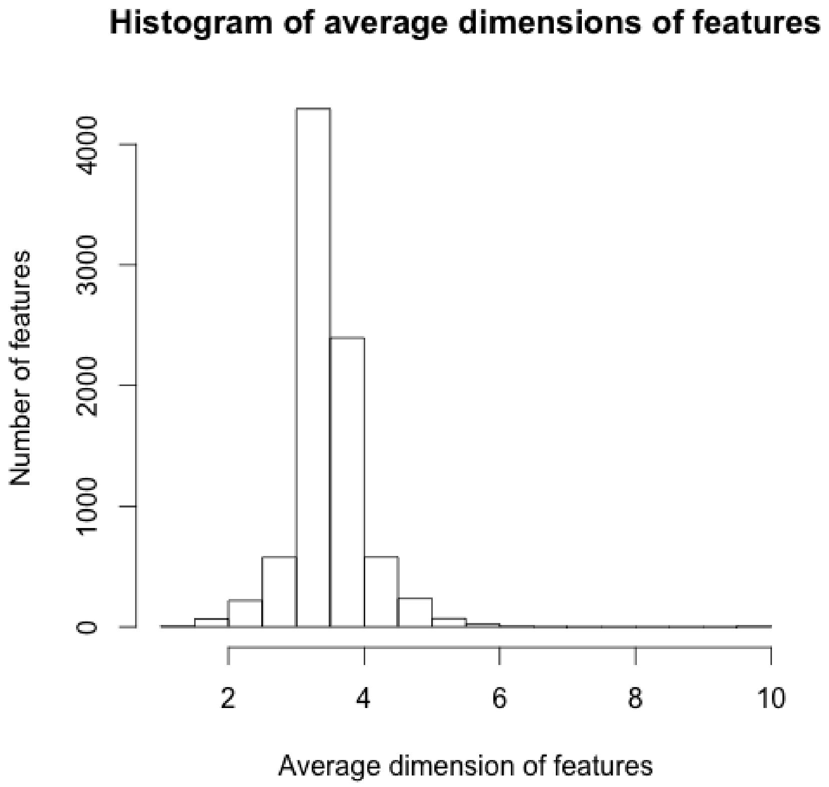


Figure 4.4: Histogram of average dimensions of all features

results provide valuable insights. Perform unsupervised clustering of similar words by considering words associated with the same factor. A factor may indicate a particular concept. The concept can be identified by looking at words with high loadings or high proportion of variance. The semantic complexity of a word can be represented by dimension of the word, which may be useful in text classification.

## 4.5 Conclusion

We present a factor model as a framework to explore the correlation structure of mixed-type outcomes, including continuous, binary and count data. The model links discrete observations of various types to latent continuous variables. The correlation structure of the mixed-type observations is inferred by a Gaussian factor model which is built on these latent continuous variables. We use a Gibbs sampler to estimate the parameters of the Gaussian factor model. We validate the algorithm by successfully reconstructing the correlation matrix of simulated data. The algorithm also unsupervisedly discovers patterns in political science data. Finally, we implement the proposed algorithm as an open-source R package which allows factor analysis for any mixed-type data sets.

## 4.6 Future work

Our factor model can further incorporate ordinal data and unordered categorical data. For ordered categorical data, we use Gaussian/probit model. For unordered categorical data, we build a Gaussian factor model embedded inside a multinomial probit model. Both models can be easily combined with our current model.

### 4.6.1 Factor model for ordinal data

In this section, we present a factor model for strictly ordinal features. The model assumes that for each feature the observations are taken to reflect an latent continuous variable with some cut points. Then we impose a factor structure on the latent continuous variable.

Suppose we have $n$ ordinal observations, denoted by $\{\boldsymbol{y_i^o}, i = 1, \cdots, n\}$, where $\boldsymbol{y_i^o}$ is a $p^o$-vector, and $p^o$ is the number of ordinal features in each observation. For ease of describing the model, we only take one ordinal feature as example, say $y$. Suppose $y$ has $J$ categories, and the observations $\boldsymbol{y}$ are defined according to the value of an underlying continuous variable $\boldsymbol{y*}$:

$$\begin{aligned} y_i &= 1 \qquad \text{if } -\infty < y_i^* \leq \kappa_1 \\ y_i &= j \qquad \text{if } \kappa_{j-1} < y_i^* \leq \kappa_j \qquad j = 2, \cdots, J-1 \\ y_i &= J \qquad \text{if } \kappa_{J-1} < y_i^* \leq \infty \end{aligned}$$

Here $\kappa_j, j = 1, \cdots, J-1$ are $J-1$ cut points. We relate $\boldsymbol{y*}$ to our latent quantity $\boldsymbol{z}$ by for $i = 1, \cdots, n$:

$$y_i^* = z_i + \epsilon_i$$

where $\epsilon_i \sim N(0, 1)$. Now considering all ordinal features, we have a $n \times p^o$ matrix $\boldsymbol{z^o}$, and we further impose a factor structure on $\boldsymbol{z^o}$ that for $i = 1, \cdots, n$:

$$\boldsymbol{z_i^o} = \boldsymbol{\alpha^o} + \boldsymbol{B^o f_i} \tag{4.38}$$

with $\boldsymbol{\alpha^o}$ is a $p^o$-vector intercept term, $\boldsymbol{B^o}$ is a $p^o \times k$ loading matrix and the factors $\boldsymbol{f_i}$ are independently distributed with $\boldsymbol{f_i} \sim N(\boldsymbol{0}, \boldsymbol{I_k})$.

This model is essentially the Gaussian/probit model for ordinal data (Quinn, 2004). And this model can be combined to our current model in the same way that we combine our model for binary and count features.

### 4.6.2 Factor model for unordered categorical data

In this section, we present a factor model for strictly unordered categorical (multinomial) features. The model builds a Gaussian factor model embedded inside a multinomial probit model for analyzing categorical features. And can be incorporated into our current model.

Suppose we have $n$ multinomial observations, denoted by $\{\boldsymbol{y_i^m}, i = 1, \cdots, n\}$, where $\boldsymbol{y_i^m}$ is a $p^m$-vector, and $p^m$ is the number of categorical features in each observation. For ease of describing the model, we only take one multinomial feature as an example, say $y$. Suppose $y$ has $J$ categories, and the observations $\boldsymbol{y}$ are defined according to the value of $J-1$ underlying continuous variables $\boldsymbol{y}*$ via:

$$y_i \mid y_{ij}^* = \begin{cases} 0, & \text{if } max(\boldsymbol{y_i^*}) < 0 \\ j, & \text{if } max(\boldsymbol{y_i^*}) = y_{ij}^* > 0 \end{cases} \tag{4.39}$$

for $i = 1, \cdots, n$, where $\boldsymbol{y_i^*} = (y_{i1}, \cdots, y_{i,J-1})$, and $max(\boldsymbol{y_i^*})$ is the largest element of the vector $\boldsymbol{y_i^*}$. The latent variables $\boldsymbol{y^*}$ is related to the latent quantities $\boldsymbol{z}$ by:

$$y_{ij}^* = z_{ij} + \epsilon_{ij} \tag{4.40}$$

where $\epsilon_{ij}$ is following a standard normal distribution for $i = 1, \cdots, n$ and $j = 1, \cdots, J-1$.

Now take all categorical features into consideration. Suppose for each multinomial feature $j$, we have $J_j$ categories, for $j = 1, \cdots, p^m$. Then we have $J_j - 1$ latent variables and latent quantities associated to $y_{ij}$ for $i = 1, \cdots, n$. Thus we have a $n \times p^M$ matrix $\boldsymbol{z^m}$, where $p^M = \sum_{j=1,\cdots,p^m}(J_j - 1)$. We further impose a factor structure on $\boldsymbol{z^m}$ that for $i = 1, \cdots, n$:

$$\boldsymbol{z_i^m} = \boldsymbol{\alpha^m} + \boldsymbol{B^m f_i} \tag{4.41}$$

with $\boldsymbol{\alpha^m}$ is a $p^M$-vector intercept term, $\boldsymbol{B^m}$ is a $p^M \times k$ loading matrix and the factors $\boldsymbol{f_i}$ are independently distributed with $\boldsymbol{f_i} \sim N(\boldsymbol{0}, \boldsymbol{I_k})$.

This model can also be incorporated to our current model. We can combine the loading matrix, intercept terms and the latent quantities for all features with all types to form a combined Gaussian factor model. And this is exactly the same way that we combine all other features as in Figure 4.1.

### 4.6.3 Software Development

These two models can be considered as Gaussian factor models embedded in a probit model for ordered/unordered categorical data. Albert and Chib (1993) and Johnson and Albert (2006) described Gibbs samplers in details about inferencing parameters in a probit model for ordered/unordered categorical data. We follow their steps for inferencing all parameters mentioned above except parameters related to factor structure: $\boldsymbol{B}$, $\boldsymbol{f}$ and $\boldsymbol{\alpha}$, which can be inferenced the same way as in Section 4.3.1.2.

Currently, we developed an R package for analysis on mixed data, with at most five types: continuous, binary, count, ordinal and multinomial. We would like to improve the package in several ways:

1. to use a Gaussian approximation to the Pòlya-Gamma random variable for faster inference (Glynn et al., 2015). This will make our package to perform

more efficiently for high dimensional data set.

2. to further develop the algorithm for different link functions. This will allow users to choose different link functions (logit or probit) for modeling binary/ordinal/multinomial data based on specific data set.

3. to benchmark our package to other packages that works on similar, but more constrained data types.

# Chapter 5

# Future Directions

A key draw-back of the Bayesian model for latent annotations described in Chapter 2 is the processing time required to do syntax reconstruction. This processing time increases significantly with the number of latent annotations, and the number of sentences in the training set. This can be potentially improved with parallelization. Nevertheless, the algorithm provides for potential significant advances in parsing low resource languages.

One way the auto-completer in Chapter 3 can be improved is through the incorporation of universal grammar rules. The auto-completer has the ability to encode both universal grammar and test-language grammar as a prior. This would be done by replacing the uniform prior used in the work with a prior favoring those grammar rules during the updating-rule-probabilities phase of the GPC. This essentially has the effect of making those grammar rules more likely during reconstruction. This can provide significant improvement in annotation projects where very little annotation data is available.

One draw-back of the algorithm described in Chapter 4 is that Gibbs sampling is slow. This is primarily due to sampling the Pòlya-Gamma random variables. We would like to improve this by using a Gaussian approximation to the

Pòlya-Gamma random variable for faster inference. Furthermore, we would like to improve the methodology by extending the algorithm to different link functions. This would be especially helpful with binary, ordinal, and multinomial data. The factor model for exploring covariation among multiple outcomes of mixed types described in Chapter 4 has already been implemented as an R package, which currently supports five types: continuous, binary, count, ordinal and multinomial.

# Bibliography

Afonso, S., Bick, E., Haber, R., and Santos, D. (2002). "floresta sintá(c)tica": a treebank for portuguese. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation (LREC)*, pages 1698–1703. LREC.

Aguilar, O. and West, M. (1998). *Bayesian dynamic factor models and variance matrix discounting for portfolio allocation*. Institute of Statistics and Decision Sciences, Duke University.

Aho, A. V. and Ullman, J. D. (1972). *The theory of parsing, translation, and compiling*. Prentice-Hall, Inc.

Albert, J. H. and Chib, S. (1993). Bayesian analysis of binary and polychotomous response data. *Journal of the American statistical Association*, 88(422):669–679.

Alicante, A., Bosco, C., Corazza, A., and Lavelli, A. (2012). A treebank-based study on the influence of Italian word order on parsing performance. In Chair), N. C. C., Choukri, K., Declerck, T., Doan, M. U., Maegaard, B., Mariani, J., Odijk, J., and Piperidis, S., editors, *Proceedings of LREC'12*, Istanbul, Turkey. European Language Resources Association (ELRA).

Bender, E. M., Flickinger, D., and Oepen, S. (2002). The Grammar Matrix: An Open-Source Starter-Kit for the Rapid Development of Cross-Linguistically Consistent Broad-Coverage Precision Grammars. In Carroll, J., Oostdijk, N., and

Sutcliffe, R., editors, *Proceedings of the Workshop on Grammar Engineering and Evaluation at the 19th International Conference on Computational Linguistics*, pages 8–14, Taipei, Taiwan.

Berger, J. O. and Sun, D. (2008). Objective priors for the bivariate normal model. *The Annals of Statistics*, pages 963–982.

Bernardo, J., Bayarri, M., Berger, J., Dawid, A., Heckerman, D., Smith, A., and West, M. (2003). Bayesian factor regression models in the large p, small n paradigm. *Bayesian statistics*, 7:733–742.

Bikel, D. (2004a). *On The Parameter Space of Generative Lexicalized Statistical Parsing Models*. PhD thesis, University of Pennsylvania.

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

Black, E., Jelinek, F., Lafferty, J., Magerman, D. M., Mercer, R., and Roukos, S. (1992). Towards history-based grammars: Using richer models for probabilistic parsing. In *Proceedings of the workshop on Speech and Natural Language*, pages 134–139. Association for Computational Linguistics.

Blunsom, P. and Cohn, T. (2010). Unsupervised induction of tree substitution grammars for dependency parsing. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1204–1213. Association for Computational Linguistics.

Booth, T. L. and Thompson, R. A. (1973). Applying probability measures to abstract languages. *Computers, IEEE Transactions on*, 100(5):442–450.

Bordes, A., Bottou, L., and Gallinari, P. (2009). Sgd-qn: Careful quasi-newton stochastic gradient descent. *The Journal of Machine Learning Research*, 10:1737–1754.

Bosco, C., Lombardo, V., Vassallo, D., and Lesmo, L. (2000). Building a Treebank for Italian: a Data-driven Annotation Schema. In *In Proceedings of the Second International Conference on Language Resources and Evaluation LREC-2000 (pp. 99*, pages 99–105.

Carroll, G. and Charniak, E. (1992). *Two experiments on learning probabilistic dependency grammars from corpora*. Department of Computer Science, Univ.

Carvalho, C. M., Chang, J., Lucas, J. E., Nevins, J. R., Wang, Q., West, M., et al. (2008). High-dimensional sparse factor modeling: Applications in gene expression genomics. *Journal of the American Statistical Association*, 103(484):1438–1456.

Charniak, E. (1996). Tree-bank grammars. In *Proceedings of the National Conference on Artificial Intelligence*, pages 1031–1036.

Charniak, E. (2000). A maximum-entropy-inspired parser. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, pages 132–139. Association for Computational Linguistics.

Chomsky, N. (1956). Three models for the description of language. *Information Theory, IRE Transactions on*, 2(3):113–124.

Chris, M. T. M. N. S. and Smith, D. N. A. (2014). Simplified dependency annotations with gfl-web. *ACL 2014*, page 121.

Cohen, S. B., Stratos, K., Collins, M., Foster, D. P., and Ungar, L. (2012). Spectral learning of latent-variable PCFGs. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 223–231. Association for Computational Linguistics.

Cohen, S. B., Stratos, K., Collins, M., Foster, D. P., and Ungar, L. (2013). Experiments with spectral learning of latent-variable PCFGs. In *Proceedings of NAACL-HLT*, pages 148–157.

Collins, M. (1997). Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics*, pages 16–23. Association for Computational Linguistics.

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

Collins, M. J. (1996). A new statistical parser based on bigram lexical dependencies. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, pages 184–191. Association for Computational Linguistics.

Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159.

Dunson, D. B. (2000). Bayesian latent variable models for clustered mixed outcomes. *Journal of the Royal Statistical Society. Series B, Statistical Methodology*, pages 355–366.

Dunson, D. B. and Herring, A. H. (2005). Bayesian latent variable models for mixed discrete outcomes. *Biostatistics*, 6(1):11–25.

Eisner, J. and Satta, G. (1999). Efficient parsing for bilexical context-free grammars and head automaton grammars. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, pages 457–464. Association for Computational Linguistics.

Finkel, J. R., Manning, C. D., and Ng, A. Y. (2006). Solving the problem of cascading errors: Approximate Bayesian inference for linguistic annotation pipelines. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 618–626. Association for Computational Linguistics.

Garrette, D. and Baldridge, J. (2013). Learning a Part-of-Speech Tagger from Two Hours of Annotation. In *Proceedings of NAACL*, Atlanta, Georgia.

Garrette, D., Mielens, J., and Baldridge, J. (2013). Real-World Semi-Supervised Learning of POS-Taggers for Low-Resource Languages. In *Proceedings of the*

*51th annual meeting on Association for Computational Linguistics*. Association for Computational Linguistics.

Geman, S. and Geman, D. (1984). Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (6):721–741.

Geweke, J. and Zhou, G. (1996). Measuring the pricing error of the arbitrage pricing theory. *Review of Financial Studies*, 9(2):557–587.

Glynn, C., Tokdar, S. T., Banks, D. L., and Howard, B. (2015). Bayesian analysis of dynamic linear topic models. *arXiv preprint arXiv:1511.03947*.

Goodman, J. T. (1998). *Parsing Inside-Out*. PhD thesis, Harvard University Cambridge, Massachusetts.

Gueorguieva, R. V. and Agresti, A. (2001). A correlated probit model for joint modeling of clustered binary and continuous responses. *Journal of the American Statistical Association*, 96(455):1102–1112.

Hoff, P. D. (2007). Extending the rank likelihood for semiparametric copula estimation. *The Annals of Applied Statistics*, pages 265–283.

Huang, Z. and Harper, M. (2009). Self-Training PCFG grammars with latent annotations across languages. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2-Volume 2*, pages 832–841. Association for Computational Linguistics.

Hwa, R. (1999). Supervised grammar induction using training data with limited constituent information. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, pages 73–79. Association for Computational Linguistics.

Hwa, R., Resnik, P., and Weinberg, A. (2005). Breaking the Resource Bottleneck for Multilingual Parsing. In *The Proceedings of the Workshop on Linguistic Knowledge Acquisition and Representation: Bootstrapping Annotated Language Data*. Conference on Language Resources and Evaluation.

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

Johnson, M. (2007). Transforming projective bilexical dependency grammars into efficiently-parsable cfgs with unfold-fold. In *ANNUAL MEETING-ASSOCIATION FOR COMPUTATIONAL LINGUISTICS*, volume 45, page 168.

Johnson, M., Griffiths, T., and Goldwater, S. (2007). Bayesian inference for PCFGs via Markov Chain Monte Carlo. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 139–146.

Johnson, V. E. and Albert, J. H. (2006). *Ordinal data modeling*. Springer Science & Business Media.

Klein, D. and Manning, C. D. (2003). Accurate unlexicalized parsing. In *Proceed-*

*ings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 423–430. Association for Computational Linguistics.

Klein, D. and Manning, C. D. (2004). Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 478. Association for Computational Linguistics.

Kong, L., Schneider, N., Swayamdipta, S., Bhatia, A., Dyer, C., and Smith, N. A. (2014). A dependency parser for tweets. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing, Doha, Qatar, to appear*.

Koop, G., Poirier, D. J., and Tobias, J. L. (2007). *Bayesian econometric methods*. Cambridge University Press.

Kuhn, J. (2004a). Applying computational linguistic techniques in a documentary project for Qanjobal (Mayan, Guatemala). In *In Proceedings of LREC 2004*. Citeseer.

Kuhn, J. (2004b). Experiments in parallel-text based grammar induction. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 470. Association for Computational Linguistics.

Lary, K. and Young, S. J. (1990). The estimation of stochastic context-free grammars using the inside-outside algrithm. *Computer, Speech and Language*, 4:35–56.

Liang, P., Jordan, M. I., and Klein, D. (2009). Probabilistic grammars and hierarchical Dirichlet processes. *The handbook of applied Bayesian analysis*.

Littlestone, N., Warmuth, M. K., and Long, P. M. (1995). On-line learning of linear functions. *Computational Complexity*, 5(1):1–23.

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

Marcus, M. P., Santorini, B., and Marcinkiewicz, M. A. (1993). Building a large annotated corpus of english: The penn treebank. *COMPUTATIONAL LINGUISTICS*, 19(2):313–330.

Marecek, D. and Straka, M. (2013). Stop-probability estimates computed on a large corpus improve unsupervised dependency parsing. In *ACL (1)*, pages 281–290.

Martins, A. F., Smith, N. A., Xing, E. P., Aguiar, P. M., and Figueiredo, M. A. (2010). Turbo parsers: Dependency parsing by approximate variational inference. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 34–44. Association for Computational Linguistics.

Matsuzaki, T., Miyao, Y., and Tsujii, J. (2005). Probabilistic CFG with latent annotations. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 75–82. Association for Computational Linguistics.

Mielens, J., Sun, L., and Baldridge, J. (2015). Parse imputation for dependency annotations. In *Proc. of ACL*.

Moser, S. and Reeves, A. (2014). Taking the leap: Voting, rhetoric, and the determinants of electoral reform. *Legislative Studies Quarterly*, 39(4):467–502.

Murray, J. S., Dunson, D. B., Carin, L., and Lucas, J. E. (2013). Bayesian gaussian copula factor models for mixed data. *Journal of the American Statistical Association*, 108(502):656–665.

Naseem, T., Chen, H., Barzilay, R., and Johnson, M. (2010). Using universal linguistic knowledge to guide grammar induction. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1234–1244. Association for Computational Linguistics.

Pereira, F. and Schabes, Y. (1992). Inside-outside reestimation from partially bracketed corpora. In *Proceedings of the 30th annual meeting on Association for Computational Linguistics*, pages 128–135. Association for Computational Linguistics.

Petrov, S., Barrett, L., Thibaux, R., and Klein, D. (2006). Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 433–440. Association for Computational Linguistics.

Petrov, S. and Klein, D. (2007). Improved Inference for Unlexicalized Parsing. In *HLT-NAACL*, pages 404–411.

Petrov, S. and Klein, D. (2008). Sparse multi-scale grammars for discriminative latent variable parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 867–876. Association for Computational Linguistics.

Polson, N. G., Scott, J. G., and Windle, J. (2013). Bayesian inference for logistic models using pólya–gamma latent variables. *Journal of the American Statistical Association*, 108(504):1339–1349.

Ponvert, E., Baldridge, J., and Erk, K. (2011). Simple Unsupervised Grammar Induction from Raw Text with Cascaded Finite State Models. In *ACL*, pages 1077–1086.

Quinn, K. M. (2004). Bayesian factor analysis for mixed ordinal and continuous responses. *Political Analysis*, 12(4):338–353.

Richard Hahn, P., Carvalho, C. M., and Scott, J. G. (2012). A sparse factor analytic probit model for congressional voting patterns. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 61(4):619–635.

Schneider, N., O'Connor, B., Saphra, N., Bamman, D., Faruqui, M., Smith, N. A., Dyer, C., and Baldridge, J. (2013). A framework for (under) specifying dependency syntax without overloading annotators. *arXiv preprint arXiv:1306.2091*.

Scott, J. and Pillow, J. W. (2012). Fully bayesian inference for neural models with negative-binomial spiking. In *Advances in neural information processing systems*, pages 1898–1906.

Shi, J.-Q. and Lee, S.-Y. (1998). Bayesian sampling-based approach for factor analysis models with continuous and polytomous data. *British Journal of Mathematical and Statistical Psychology*, 51(2):233–252.

Shindo, H., Miyao, Y., Fujino, A., and Nagata, M. (2012). Bayesian symbol-refined tree substitution grammars for syntactic parsing. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 440–448. Association for Computational Linguistics.

Spitkovsky, V. I., Alshawi, H., and Jurafsky, D. (2011). Punctuation: Making a point in unsupervised dependency parsing. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*, pages 19–28. Association for Computational Linguistics.

Spitkovsky, V. I., Alshawi, H., and Jurafsky, D. (2012). Three dependency-and-boundary models for grammar induction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 688–698. Association for Computational Linguistics.

Spitkovsky, V. I., Jurafsky, D., and Alshawi, H. (2010). Profiting from mark-up: Hyper-text annotations for guided parsing. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1278–1287. Association for Computational Linguistics.

Sun, L., Mielens, J., and Baldridge, J. (2014). Parsing low-resource languages using gibbs sampling for pcfgs with latent annotations. In *Proceedings of the 2014*

*Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Taddy, M. A. (2011). On estimation and selection for topic models. *arXiv preprint arXiv:1109.4518*.

Toutanova, K., Klein, D., Manning, C. D., and Singer, Y. (2003). Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 173–180. Association for Computational Linguistics.

Tsitsiklis, J. N., Bertsekas, D. P., Athans, M., et al. (1986). Distributed asynchronous deterministic and stochastic gradient optimization algorithms. *IEEE transactions on automatic control*, 31(9):803–812.

Ventura, V., Cai, C., and Kass, R. E. (2005). Trial-to-trial variability and its effect on time-varying dependency between two neurons. *Journal of neurophysiology*, 94(4):2928–2939.

Wang, P. and Blunsom, P. (2013). Collapsed Variational Bayesian Inference for PCFGs. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 173–182, Sofia, Bulgaria. Association for Computational Linguistics.

Xue, N., Xia, F., Chiou, F.-d., and Palmer, M. (2005). The Penn Chinese TreeBank: Phrase structure annotation of a large corpus. *Nat. Lang. Eng.*, 11(2):207–238.

Zhou, M., Hannah, L. A., Dunson, D. B., and Carin, L. (2012a). Beta-negative binomial process and poisson factor analysis. *stat*, 1050:4.

Zhou, M., Li, L., Dunson, D., and Carin, L. (2012b). Lognormal and gamma mixed negative binomial regression. In *Machine learning: proceedings of the International Conference. International Conference on Machine Learning*, volume 2012, page 1343. NIH Public Access.

# Vita

Liang Sun was born in Maanshan, Anhui, China in 1988. She obtained the B.S. degree in Mathematics in 2010 from the University of Hong Kong. In August 2010, Liang Sun entered the Ph.D. program of the Operations Research and Industrial Engineering group at The University of Texas at Austin.

Permanent email: liangs.sun.ls@gmail.com

This dissertation was typed by the author.

Permanent address: 11417 Powder Mill Trl
Austin, Texas 78750

This dissertation was typeset with LaTeX[†] by the author.

[†]LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.