Parsing low-resource languages using Gibbs sampling for PCFGs with latent annotations

Liang Sun[1] **Jason Mielens**[2] **Jason Baldridge**[2]

[1]Department of Mechanical Engineering
The University of Texas at Austin
sally722@utexas.edu

[2]Department of Linguistics
The University of Texas at Austin
{jmielens,jbaldrid}@utexas.edu

Abstract

PCFGs with latent annotations have been shown to be a very effective model for phrase structure parsing. We present a Bayesian model and algorithms based on a Gibbs sampler for parsing with a grammar with latent annotations. For PCFG-LA, we present an additional Gibbs sampler algorithm to learn annotations from training data, which are parse trees with coarse (unannotated) symbols. We show that a Gibbs sampling technique is capable of parsing sentences in a wide variety of languages and producing results that are on-par with or surpass previous approaches. Our results for Kinyarwanda and Malagasy in particular demonstrate that low-resource language parsing can benefit substantially from a Bayesian approach.

1 Introduction

Despite great progress over the past two decades on parsing, relatively little work has considered the problem of creating accurate parsers for low-resource languages. Existing work in this area focuses primarily on approaches that use some form of cross-lingual bootstrapping to improve performance. For instance, Hwa et al. (2005) use a parallel Chinese/English corpus and an English dependency grammar to induce an annotated Chinese corpus in order to train a Chinese dependency grammar. Kuhn (2004b) also considers the benefits of using multiple languages to induce a monolingual grammar, making use of a measure for data reliability in order to weight training data based on confidence of annotation. Bootstrapping approaches such as these achieve markedly improved results, but they are dependent on the existence of a parallel bilingual corpus. Very few such corpora are readily available, particularly for low-resource languages, and creating such corpora obviously presents a challenge for many practical applications. Kuhn (2004a) shows some of the difficulty in handling low-resource languages by examining various tasks using Q'anjob'al as an example. Another approach is that of Bender et al. (2002), who take a more linguistically-motivated approach by making use of linguistic universals to seed newly developed grammars. This substantially reduces the effort by making it unnecessary to learn the basic parameters of a language, but it lacks the robustness of grammars learned from data.

Recent work on Probabilistic Context-Free Grammars with latent annotations (PCFG-LA) (Matsuzaki et al., 2005; Petrov et al., 2006) have shown them to be effective models for syntactic parsing, especially when less training material is available (Liang et al., 2009; Shindo et al., 2012). The coarse nonterminal symbols found in vanilla PCFGs are refined by latent variables; these latent annotations can model subtypes of grammar symbols that result in better grammars and enable better estimates of grammar productions. In this paper, we provide a Gibbs sampler for learning PCFG-LA models and show its effectiveness for parsing low-resource languages such as Malagasy and Kinyawanda.

Previous PCFG-LA work focuses on the problem of parameter estimation, including expectation-maximization (EM) (Matsuzaki et al., 2005; Petrov et al., 2006), spectral learning (Cohen et al., 2012; Cohen et al., 2013), and variational inference (Liang et al., 2009; Wang and Blunsom, 2013). Regardless of inference method, previous work has used the same method to parse new sentences: a Viterbi parse under a new sentence-specific PCFG obtained from an approximation of the original grammar (Matsuzaki et al., 2005). Here, we provide an alternative approach to parsing new sentences: an extension of the Gibbs sampling algorithm of Johnson et al. (2007), which learns rule probabilities in an unsupervised PCFG.

We use a Gibbs sampler 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). We also build a Bayesian model for parsing with a treebank, and incorporate information from training data as a prior. Moreover, we extend the Gibbs sampler to learn and parse PCFGs with latent annotations. Learning the latent annotations is a compute-intensive process. We show how a small amount of training data can be used to bootstrap: after running a large number of sampling iterations on a small set, the resulting parameters are used to seed a smaller number of iterations on the full training data.

This allows us to employ more latent annotations while maintaining reasonable training times and still making full use of the available training data.

To determine the cross-linguistic applicability of these methods, we evaluate on a wide variety of languages with varying amounts of available training data. We use English and Chinese as examples of languages with high data availability, while Italian, Malagasy, and Kinyarwanda provide examples of languages with little available data.

We find that our technique comes near state of the art results on large datasets, such as those for Chinese and English, and it provides excellent results on limited datasets – both artificially limited in the case of English, and naturally limited in the case of Italian, Malagasy, and Kinyarwanda. This, combined with its ability to run off-the-shelf on new languages without any supporting materials such as parallel corpora, make it a valuable technique for the parsing of low-resource languages.

2 Gibbs sampling for PCFGs

Our starting point is a Gibbs Sampling algorithm for vanilla PCFGs introduced by Johnson et al. (2007) for estimating rule probabilities in an unsupervised PCFG. We focus instead on using this algorithm for parsing new sentences and then extending it to learn PCFGs with latent annotations. We begin by summarizing the Bayesian PCFG and Gibbs sampler defined by Johnson et al. (2007).

Bayesian PCFG For a grammar G, each rule r in the set of rules R has an associated probability θ_r. The probabilities for all the rules that expand the same nonterminal A must sum to one: $\sum_{A\to\beta\in R}\theta_{A\to\beta}=1$.

Given an input corpus $\boldsymbol{w}$=($\boldsymbol{w^{(1)}},\cdots,\boldsymbol{w^{(n)}}$), we introduce a latent variable $\boldsymbol{t}$=($t^{(1)},\cdots,t^{(n)}$) for trees generated by G for each sentence. The joint posterior distribution of $\boldsymbol{t}$ and θ conditioned on $\boldsymbol{w}$ is:

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

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 yield of $t^{(i)}$ is the sequence $w^{(i)}$, and 0 otherwise.

We use a Dirichlet distribution parametrized by α_A: $Dir(\alpha_A)$ as the prior of the probability distribution for all rules expanding non-terminal A ($p(\theta_A)$). The prior for all θ, $p(\theta)$, is the product of all Dirichlet distributions over all non-terminals $A\in N$: $p(\theta\mid\alpha)=\prod_{A\in N}p(\theta_A\mid\alpha_A)$.

Since the Dirichlet distribution is conjugate to the Multinomial distribution, which we use to model the likelihood of trees, the conditional posterior of θ_A can be updated as follows:

$$\begin{aligned} p_G(\theta\mid\boldsymbol{t},\alpha) &\propto p_G(\boldsymbol{t}\mid\theta)p(\theta\mid\alpha) \\ &\propto (\textstyle\prod_{r\in R}\theta_r^{f_r(\boldsymbol{t})})(\prod_{r\in R}\theta_r^{\alpha_r-1}) \\ &= \textstyle\prod_{r\in R}\theta_r^{f_r(\boldsymbol{t})+\alpha_r-1} \quad (2)\end{aligned}$$

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

Gibbs sampler The parameters of the PCFG model can be learned from an annotated corpus by simply counting rules. However, parsing cannot be done directly with standard CKY as with standard PCFGs, so we use the Gibbs sampling algorithm presented in Johnson et al. (2007). An additional motivation for using this algorithm is that Johnson et al. use it for learning without annotated structures, and in future work we seek to learn from fewer, and at times partial, annotations.

An advantage of using Gibbs sampling for Bayesian inference, as opposed to other approximation algorithms such as Variational Bayesian inference (VB) and Collapsed Variational Bayesian inference (CVB), is that Markov Chain Monte Carlo (MCMC) algorithms are guaranteed to converge to a sample from the true posterior under appropriate conditions (Taddy, 2011). Both VB and CVB converge to inaccurate and locally optimal solutions, like EM. In some models, CVB can achieve more accurate results due to weaker assumptions (Wang and Blunsom, 2013). Another advantage of Gibbs sampling is that the sampler allows for parallel computation by allowing each sentence to be sampled entirely independently of the others. After each parallel sampling stage, all model parameters are updated in a single step, and the process then repeats (see §2).

To sample the joint posterior $p(\boldsymbol{t},\theta\mid\boldsymbol{w})$, we sample production probabilities θ and then trees $\boldsymbol{t}$ from these conditional distributions:

$$p(\boldsymbol{t}\mid\theta,\boldsymbol{w},\alpha)=\textstyle\prod_{i=1}^{n}p(t_i\mid w_i,\theta) \quad (3)$$

$$p(\theta\mid\boldsymbol{t},\boldsymbol{w},\alpha)=\textstyle\prod_{A\in N}Dir(\theta_A\mid f_A(\boldsymbol{t})+\alpha_A) \quad (4)$$

Step 1: Sample Rule Probabilities. Given trees $\boldsymbol{t}$ and prior α, the production probabilities θ_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.

Step 2: Sample Tree Structures. To sample trees from $p(t_i\mid\boldsymbol{w_i},\theta)$, we use the efficient sampling scheme used in previous work (Goodman, 1998; Finkel et al., 2006; Johnson et al., 2007). There are two parts to this 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.

Require: A is parent node of 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**
$P(j,B,C) = \frac{\theta_{A\to BC}\cdot p_{B,i,j}\cdot p_{C,j,k}}{\cdot} p_{A,i,k}$
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

Consider a sentence $\boldsymbol{w}$, with sub-spans $w_{i,k} = (w_{i+1}, \cdots, w_k)$. Given $\boldsymbol{\theta}$, we construct the inside table with entries $p_{A,i,k}$ for each nonterminal and each word span $w_{i,k} : 0 \le i < k \le l$, where $p_{A,i,k} = P_{G_A}(w_{i,k}|\theta)$ is the probability that words i through k were produced by the non-terminal A. The table is computed recursively by

$$p_{A,k-1,k} = \theta_{A\to w_k} \quad (5)$$

$$p_{A,i,k} = \sum_{A\to BC\in R}\sum_{i<j<k} \theta_{A\to BC}\cdot p_{B,i,j}\cdot p_{C,j,k} \quad (6)$$

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

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.

In unsupervised PCFG learning, the rule probabilities can be resampled using the sampled trees, then used to reparse the corpus, and so on. We use this property to refine latent annotations for the PCFG-LA model described in the next section.

3 PCFG with latent annotations

When labeled trees are available, rule frequencies can be directly extracted and used as priors for a PCFG. However, when learning PCFG-LAs, we must learn the fine-grained rules from the coarse trees, so we extend the Gibbs sampler to assign latent annotations to unannotated trees. The resulting learned PCFG-LA parser outputs samples of annotated trees so that we can obtain unannotated trees after marginalizing.

3.1 Model

With the PCFG-LA model (Matsuzaki et al., 2005; Petrov et al., 2006) fine-grained CFG rules are automatically induced from training, effectively providing a form of feature engineering without human intervention. Given $H = \{1, \cdots, K\}$, a set of latent annotation symbols, and $x \in H$:

- $\theta_{A[x]\to U}$ is the probability of rule $A[x] \to U$, where $U \in N\times N \cup T$. The probabilities for all rules that expand the same annotated non-terminal must sum to one.
- $\beta_{A[x],B,C\to y,z}$ is the probability of assigning latent annotation y, z to child nodes B, C of $A[x]$. $\sum_{y,z\in H\times H}\beta_{A[x],B,C\to y,z} = 1$.

The inputs to the PCFG-LA are a CFG G with finite number of latent annotations for each non-terminal, an initial guess of probabilities of grammar rule θ_0, and a prior α_θ is learned from training.

The joint posterior distribution of $\boldsymbol{t}$ and θ, β conditioned on $\boldsymbol{w}$ is:

$$\begin{aligned} p(\boldsymbol{t},\theta,\beta \mid \boldsymbol{w}) &\propto p(\theta,\beta)p(\boldsymbol{w}\mid\boldsymbol{t})p(\boldsymbol{t}\mid\theta,\beta) \\ &= p(\theta)p(\beta)(\prod_{i=1}^{n} p(w_i\mid t_i)p(t_i\mid\theta,\beta)) \end{aligned} \quad (7)$$

We assume that θ and β are independent to get $P(\theta,\beta) = P(\theta)P(\beta)$.

To learn parameters θ, β, we use a Dirichlet distribution as a prior for both θ and β. The distribution for all rules expanding $A[x]$ is:

$$P(\theta\mid\alpha^\theta) = \prod_{A\in N, x\in H} P(\theta_{A[x]}\mid\alpha^\theta_{A[x]}) \quad (8)$$

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

$$P(\beta\mid\alpha^\beta) = \prod_{y,z\in H\times H} P(\beta_{A[x],B,C}\mid\alpha^\beta_{A[x],B,C}). \quad (9)$$

With this setting, the conditional posterior of $\theta_{A[x]}$ and $\beta_{A[x],B,C}$ can be updated, as in §2. For all unary and binary rules r expanding $A[x]$:

$$\theta_{A[x]}\mid\boldsymbol{t},\alpha^\theta \sim Dir(f_r(\boldsymbol{t})+\alpha^\theta_r) \quad (10)$$

Here, $f_r(\boldsymbol{t})$ is the number of occurrence of annotated rule r in $\boldsymbol{t}$. Also, for combination of latent annotations $y, z \in H\times H$ assigned to B, C in rule $A[x]\to B, C$:

$$\beta_{A[x],B,C}\mid\boldsymbol{t},\alpha^\beta \sim Dir(f_d(\boldsymbol{t})+\alpha^\beta_d) \quad (11)$$

Here, $f_d(\boldsymbol{t})$ is the number of occurrences of combination d in $\boldsymbol{t}$.

3.2 Learning PCFG-LAs from raw text

To learn from raw text, we extend the sampler in §2 to PCFG-LA. Given priors $\alpha^\theta, \alpha^\beta$ and raw text, the algorithm alternates between two steps. The first samples trees for the entire corpus; the second samples θ and β 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 θ, β and annotated trees.

The parsing process is specified in Algorithm 2. The first step assigns a tree to a sentence, say $w_{0,l}$. We first

Require: $w_1, \cdots, w_n$ are raw sentences; θ_0, β_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**
Marginalize 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

construct an inside table (see §2). Each entry in the table stores the probability that a word span is produced by a given annotated nonterminal. For root node S, with θ, β and inside table $p_{A[x],i,k}$, we sample one annotation based on all $p_{S[x],0,l}$, $x \in H$. Assume that we sampled x for S, we further sample a rule to expand $S[x]$ and possible splits of the span $w_{0,l}$ jointly. Assume that we sampled nonterminals B, C to expand $S[x]$, where B is responsible for $w_{0,j}$ and C is responsible for $w_{j,l}$. We further sample annotations for B, C together, say y, z. Then we sample rules and split positions to expand $B[y]$ and $C[z]$, and continue until reaching the terminals.

This algorithm alone could be used for unsupervised learning of PCFG-LA if we input a non-informed or weakly-informed prior α^θ and α^β. With access to unannotated trees for training, we only need to assign latent annotations to them and then use the frequencies of these annotated rules as the prior when parsing. The details of training when trees are available are illustrated in §3.3.

Once we have trees (with latent annotations), the step of sampling θ and β from a Dirichlet distribution is direct. We need to count the number of occurrences $f_r(t)$ for each rule r like $A[x] \to U, U \in N \times N \cup T$ in updated annotated trees $\boldsymbol{t}$, and draw $\theta_{A[x]}$ from the updated Dirichlet distribution $Dir(f_{A[x]}(\boldsymbol{t}) + \alpha^\theta_{A[x]})$. We also need to count the number of occurrences of $f_d(t)$ for each combination of $yz \in H \times H$ assigned to B, C given $A[x] \to B, C$ 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.

To parse a sentence, we first calculate the inside table and then sample the tree.

Calculate the inside table. Given $\boldsymbol{\theta}, \boldsymbol{\beta}$ and a string $\boldsymbol{w}$=$w_{0,l}$, we construct a table with entries $p_{A[x],i,k}$ for each $A \in N$, $x \in H$ and $0 \le i < k \le l$, where $p_{A[x],i,k} = P_{G_{A[x]}}(w_{i,k}|\theta, \beta)$ 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

$$p_{A[x],k-1,k} = \theta_{A[x] \to w_k} \tag{12}$$

$$p_{A[x],i,k} = \sum_{A[x] \to BC : BC \in N \times N} \sum_{j : i<j<k} \sum_{yz \in H \times H} \theta_{A[x] \to BC} \beta_{A[x]BC \to yz} p_{B[y],i,j} p_{C[z],j,k} \tag{13}$$

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:

$$p(B, C, j) = \frac{1}{p_{A[x],i,k}} \cdot \sum_{y,z \in H} \theta_{A[x] \to BC} \beta_{A[x]BC \to yz} p_{B[y],i,j} p_{C[z],j,k} \tag{14}$$

Here the probability is calculated by marginalizing all possible latent annotations for B, C, and $\theta_{A[x] \to BC} \beta_{A[x]BC \to yz}$ 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:

$$p(y, z) = \frac{\beta_{A[x]BC \to yz} p_{B[y],i,j} p_{C[z],j,k}}{\sum_{y,z} \beta_{A[x]BC \to yz} p_{B[y],i,j} p_{C[z],j,k}} \tag{15}$$

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.

3.3 Learning from coarse training trees

In training, we need to learn the probabilities of fine-grained rules given coarsely-labeled trees. We perform Gibbs sampling on the training data by first iteratively sampling probabilities and then assigning annotations to tree nodes. We use the average counts of annotated production rules from sampled trees to produce the prior α^θ and α^β incorporated into parsing raw sentences.

We 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 θ, β given annotated trees (as in §3.2). The second samples latent annotations for nonterminal nodes

Require: $T_1, \cdots, T_n$ are fully parsed trees; θ_0, β_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

in parsed trees, which also takes two steps. The first step is to, for each node in the tree, calculate and store the probability that the node is annotated by x. 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.

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 $x \in H$

$$b_T^i[x] = \theta_{N_i[x] \to w} \tag{16}$$

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

$$b_T^i[x] = \sum_{y,z \in H} \theta_{N_i[x] \to N_j N_k} \beta_{N_i[x] N_j N_k \to y,z} b_T^j[y] b_T^k[z] \tag{17}$$

Step two: outside sampling. Given inside probability $b_T^i[x]$ for every non-terminal i and all latent annotations $x \in H$, we sample the latent annotations from 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

$$f_T^i[y,z] = \frac{1}{b_T^i[x]} \cdot \theta_{N_i[x] \to N_j N_k} \beta_{N_i[x] N_j N_k \to y,z} b_T^j[y] b_T^k[z] \tag{18}$$

After training, we take the average counts of sampled annotated rules and combinations of latent annotations as priors to parse raw sentences.

4 Experiments[1]

Our goal is to understand parsing efficacy using sampling and latent annotations for low-resource languages, so we perform experiments on five languages with varying amount of training data. We compare our results to a number of previously established baselines. First, for all languages, we use both a standard unsmoothed PCFG and the Bikel parser, trained on the training corpus. Additionally, we compare to state-of-the-art results for both English and Chinese, which have an existing body of work in PCFGs using a Bayesian framework. For Chinese, we compare to Huang & Harper (2009), using their results that only use the Chinese Treebank (CTB). For English, we compare to Liang et al. (2009). Prior results for parsing the constituency version of the Italian data are available from Alicante et al. (2012), but as they make use of a different version of the treebank including extra sentences, and additionally use the extensive functional tags present in the corpus, we do not directly compare our results to theirs.[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. And since we primarily are interested in parsing low-resource languages, we include results for Kinyarwanda (KIN) and Malagasy (MLG) as examples of languages without substantial existing treebanks. Finally, as a middle-ground language, we use Italian (ITL).

For English, we use 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.

The KIN texts are transcripts of testimonies by survivors of the Rwandan genocide provided by the Kigali Genocide Memorial Center, along with a few BBC news articles. The MLG texts are articles from the websites Lakroa and La Gazette and Malagasy Global Voices. Both datasets are described in Garrette and Baldridge (2013). The KIN and MLG data is very small compared to ENG and CHI: the KIN dataset con-

[1] Code available at github.com/jmielens/gibbs-pcfg-2014, along with instructions for replicating experiments when possible

[2] As part of a standardized pre-processing step, we strip functional tags, which makes a direct comparison to their results inappropriate.

tains 677 sentences, while the MLG dataset has only 113. Also, we simulated a small training set for ENG data by using only section 02 of the WSJ for training.

4.2 Experimental Setup

As a preprocessing step, all trees are converted into Chomsky Normal-Form 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. Our 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 recurses. This simple technique uses no explicit head-finding rules, which eases cross-linguistic applicability.

From this normalized data, we train latent PCFGs with K=1,2,4,8,16,32 (where K=1 is equivalent to the plain PCFG described in section 2).

4.3 Practical refinements

Unknown word handling. We use a similar unknown word handling procedure to Liang et al. (2009). From our raw corpus we extract features associated with every word, these features include surrounding context words as well as substring suffix/prefix features. Using these features we produce fifty clusters using k-means. Then, as a pre-parsing step, we replace all words occurring less than five times 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 our corpus, as the clustering has been performed over the entire corpus prior to training. This approach is similar to methods for unsupervised POS-tag induction that also utilize clusters in this manner (Dasgupta & Ng, 2007).

We compare this unknown word handling method to one in which the clustering and a classifier is trained not on the corpus under consideration, but rather on a separate corpus of unrelated data. This comparison was made to understand the effects of including the evaluation set in the training data (without labels) versus training on out-of-domain texts. This is a more realistic measurement of out-of-the-box performance of a trained model.

Jump-starting sampling. In the basic setup, training high K-value models takes a prohibitively long time, so we also consider a jump-start technique that allows larger annotation values (such as K=16) to be run in less time. We train these high-K value models first on a highly reduced training set (5% of the full training set) for a large number of iterations, and then use the found θ values as the starting point for training on the full training set for a small number of iterations. Although many of the estimated parameters on the reduced set will be zero, the prior allows us to eventually recover this information in the full set. This allows us to train on the full training set, which is desirable relative to training on a reduced set, while still allowing the model sufficient iterations to burn in. The fact that we are likely starting in a fairly good position within the search space (due to estimating θ from the corpus) also likely helps enable these lower iteration counts.

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	**61.0**	71.3	76.6	78.7	78.0

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

5 Results

We start with Tables 1 and 2, which show performance when training on section 02 of the WSJ (pretending English is a "low-resource" language). The results show that the basic Gibbs PCFG (where K=1), with an F-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 1 also shows further large gains are obtained from using latent annotations—from 60.5 for K=1 to 78.7 for K=8.

The Gibbs PCFG 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 and K=8. Table 2 shows that the Gibbs PCFG 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 3, which shows that the Gibbs PCFG 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.

We also use the small English corpus to determine the effects of weighting the prior when sampling annotations, varying α between 0.1 and 10.0. Though performance is not sensitive to varying α for larger corpora, Figure 1 shows it can make a substantial difference for smaller corpora (with an optimal value was obtained with an α value of 5 for this small training set). This seems to indicate that the lower counts associated with the smaller training sets should be compensated for by weighting those counts more heavily when processing the evaluation set, as we had anticipated.

System	WSJ Sec. 02	KIN	MLG
Berkeley Parser	**78.3** $\pm$ **0.93**	60.6 $\pm$ 1.1	52.2 $\pm$ 2.0
Gibbs PCFG	76.7 $\pm$ 0.63	**67.2** $\pm$ **0.92**	**57.5** $\pm$ **1.1**

Table 2: F1 scores with standard deviation over ten runs of small training data, K=4.

System	F1 / StDev
Berkeley Parser	**77.5** $\pm$ **2.1**
Gibbs PCFG	77.0 $\pm$ 1.4

Table 3: F1 scores with standard deviations over twenty runs, training on individual WSJ sections (02-21).

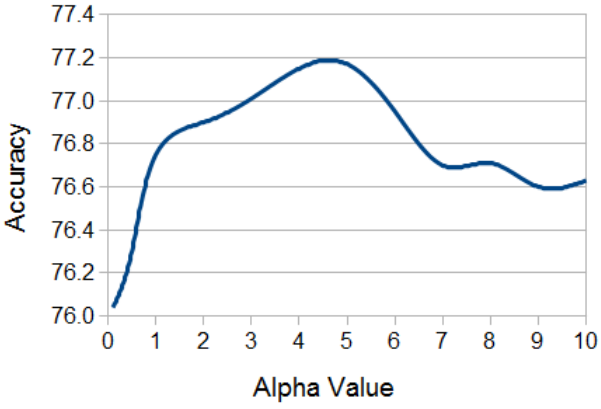

Figure 1: Accuracy by varying α levels for small English data.

To evaluate the effectiveness of the jump-start technique, we ran the full ENG data set with K=4 to compare the results from the full training setup to jump-starting. For this, we performed 100 training iterations on the reduced training set (WSJ section 02) and then used the resulting θ values to seed training on the full training set. Those training runs varied between three and nine iterations, and the results are shown in Figure 2. The full ENG K=4 F-score is 86.2, so these results represent a slight step back. Nonetheless, the technique is still valuable in that it allows for inferring latent annotations for higher K-values than would typically be available to us in a reasonable timeframe.

Table 4 shows the results for the main experiments. 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

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

Figure 2: F-Score for K=4, varying full-set training iterations (with and without 100x jump start).

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 our 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 levelling-off seen at the high K values.

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

The Gibbs PCFG with K=16 is superior to the strong Bikel and Berkeley Parser benchmarks for both KIN and MLG, a promising result for future work on parsing low-resource languages in general. Note also that our parser 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).

Examples of the improvement across latent annotations for a given tree are shown in Figure 3. The details of the noun phrase ‘no major bond offerings’ were the same for each tree, and are thus abstracted here. The low K-value tree (K=2) is shown in 3a, and primarily suffers from issues related to the prepositional phrase, ‘in Europe friday’. In particular, the low K-value tree incorrectly groups ‘Europe friday’ as a noun phrase object of ‘in’.

The higher K-value tree (K=8) is shown in 3b. This tree manages to correctly analyze the prepositional phrase, accurately separately the temporal locative ‘Friday’ from the actual prepositional phrase of ‘in Europe’. However, the high K-value tree makes a

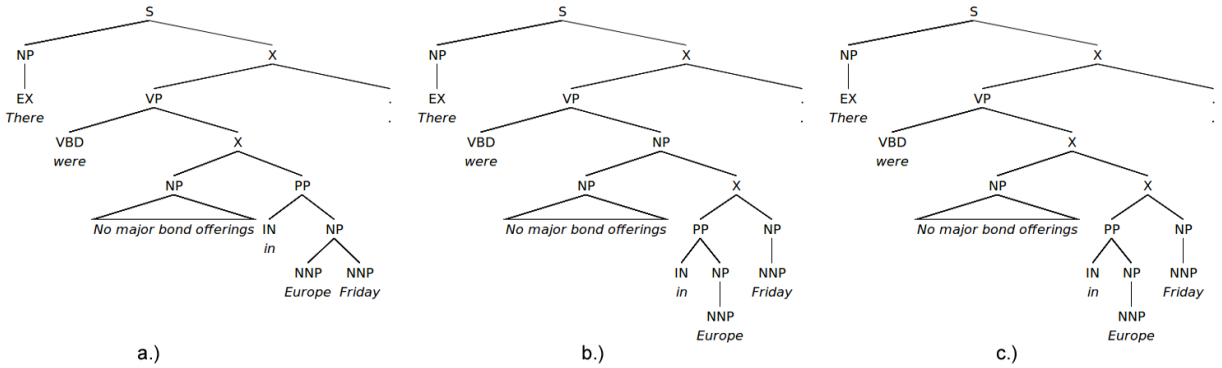

Figure 3: Examples of tree progression in the Gibbs PCFG with a) K=2, b) K=8, and c) gold tree.

different mistake that the low K-value tree did not; it groups 'no major bond offerings in Europe Friday' as a noun phrase, when it should be three separate phrases (two noun phrases and a prepositional phrase). This error may be related to the additional latent annotations. With more available noun phrase subtypes, it may be the case that a more unusual noun phrase could be permitted that would have been too low probability with only a few subtypes.

To determine whether the substantial range in F1 scores across languages are primarily the result of the much larger training corpora available for certain languages, two extreme training set reduction experiments were conducted. The training sets for all languages were reduced to a total of either 100 or 500 sentences. This process was repeated 10 times in a cross-validation setup, where 10 separate sets of sentences were selected for each language. The results of these experiments are shown in Table 5.

We conclude that while data availability is a major factor in the higher performance of English and Chinese in our original experiments, it is not the only issue. Clearly, either the linguistic facts of particular languages or perhaps choices of formalism and annotation conventions in the corpora make some of the languages more difficult to parse than others. The primary questions is why Gibbs-PCFG is able to achieve higher relative performance on the KIN/MLG datasets when compared to the other parsers, and why this advantage does not necessarily transfer to the extreme small-scale versions of the ENG/CHI/ITL data. Preliminary investigation into the properties of the corpora have revealed a number of potential answers. For instance, the POS tagsets for KIN/MLG are substantially reduced compared to the other corpora, and there are differences in the branching factor of the native versions of the corpora as well: a typical maximum branching factor for a tree in ENG/CHI/ITL is around 4-5, while for KIN/MLG it is almost always 2 (natively binary). Branching factors above 5 essentially never occur in KIN/MLG, while they are not rare in ENG/CHI/ITL. The question of exactly why the Gibbs-PCFG seems to perform well on these corpora remains an open question, but these differences could provide a starting point for future analysis.

Condition	In-Domain	Out-of-Domain
Full English (K=4)	**86.0**	83.3
Small English (K=4)	**76.6**	75.7
Kinyarwanda (K=4)	**67.2**	65.1
Malagasy (K=4)	**57.8**	55.4

Table 6: Effect of differing regimes for handling unknown words.

In addition to the actual F1 scores, the relative uniformity of the standard deviation results indicates that the individual parsers are not that much different in terms of their ability to provide consistent results at these small data extremes, as opposed to the slightly higher training levels where the Gibbs-PCFG generated smaller variances.

Considering the effects of unknown word handling, Table 6 shows that using the evaluation set when creating the unknown word classifier does improve overall parsing accuracy when compared to an unknown word handler that is trained on out-of-domain texts. This shows that results reported in previous work somewhat overstate the accuracy of these parsers when used in the wild—which matters greatly in the low-resource setting.

6 Conclusion

Our experiments demonstrate that sampling vanilla 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. Our results also show that our methods are effective on a wide variety of languages—including two low-resource languages—with no language-specific model modifications needed.

Additionally, although not a uniform winner, the Gibbs-PCFG shows a propensity for performing well on naturally small corpora (here, KIN/MLG). The exact reason for this remains slightly unclear, but the fact that a similar advantage is not found for extremely small versions of large corpora indicates that our approach may be particularly well-suited for application in real low-resource environments as opposed to a sim-

Parser	Size	ENG	CHI	ITL	KIN	MLG
Bikel	100	54.7 ± 2.2	51.4 ± 3.0	51 ± 2.4	47.1 ± 2.3	44.4 ± 2.0
Berkeley	100	55.2 ± 2.6	53.9 ± 2.9	50 ± 2.8	47.8 ± 2.1	44.5 ± 2.3
Gibbs-PCFG	100	54.5 ± 2.0	51.7 ± 2.4	49.5 ± 3.6	50.3 ± 2.3	45.8 ± 1.8
Bikel	500	56.2 ± 2.0	54.1 ± 2.7	54.2 ± 2.4	—	—
Berkeley	500	58.9 ± 2.2	56.4 ± 2.7	52.5 ± 2.7	—	—
Gibbs-PCFG	500	58.1 ± 2.0	55.7 ± 2.3	51.1 ± 3.2	—	—

Table 5: 100/500 sentence training set results, including st.dev over 10 runs. KIN/MLG did not have enough data to run the 500 sentence version.

ulated environment.

Having established this procedure and its relative tolerance for low amounts of data, we would like to extend the model to make use of partial bracketing information instead of complete trees, perhaps in the form of Fragmentary Unlabeled Dependency Grammar annotations (Schneider et al., 2013). This would allow the sampling procedure to potentially operate using corpora with lighter annotations than full trees, making initial annotation effort not quite as heavy and potentially increasing the amount of available data for low-resource languages. Additionally, using the expert partial annotations to help restrict the sample space could provide good gains in terms of training time.

Acknowledgments

Supported by the U.S. Army Research Office under grant number W911NF-10-1-0533. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the view of the U.S. Army Research Office.

References

Anita Alicante, Cristina Bosco, Anna Corazza, and Alberto Lavelli. 2012. A treebank-based study on the influence of Italian word order on parsing performance. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Mehmet Uur Doan, Bente Maegaard, Joseph Mariani, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of LREC'12*, Istanbul, Turkey. European Language Resources Association (ELRA).

Emily M. Bender, Dan Flickinger, and Stephan Oepen. 2002. The Grammar Matrix: An Open-Source Starter-Kit for the Rapid Development of Cross-Linguistically Consistent Broad-Coverage Precision Grammars. In John Carroll, Nelleke Oostdijk, and Richard Sutcliffe, editors, *Proceedings of the Workshop on Grammar Engineering and Evaluation at the 19th International Conference on Computational Linguistics*, pages 8–14, Taipei, Taiwan.

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

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

Ezra Black, Fred Jelinek, John Lafferty, David M Magerman, Robert Mercer, and Salim Roukos. 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.

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

Cristina Bosco, Vincenzo Lombardo, Daniela Vassallo, and Leonardo Lesmo. 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.

Glenn Carroll and Eugene Charniak. 1992. *Two experiments on learning probabilistic dependency grammars from corpora*. Department of Computer Science, Univ.

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

Eugene Charniak. 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.

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

Shay B Cohen, Karl Stratos, Michael Collins, Dean P Foster, and Lyle Ungar. 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.

Shay B Cohen, Karl Stratos, Michael Collins, Dean P Foster, and Lyle Ungar. 2013. Experiments with spectral learning of latent-variable PCFGs. In *Proceedings of NAACL-HLT*, pages 148–157.

Michael John Collins. 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.

Michael Collins. 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.

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

Jenny Rose Finkel, Christopher D Manning, and Andrew Y Ng. 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.

Dan Garrette and Jason Baldridge. 2013. Learning a Part-of-Speech Tagger from Two Hours of Annotation. In *Proceedings of NAACL*, Atlanta, Georgia.

Dan Garrette, Jason Mielens, and Jason Baldridge. 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.

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

Joshua T Goodman. 1998. *Parsing Inside-Out*. Ph.D. thesis, Harvard University Cambridge, Massachusetts.

Zhongqiang Huang and Mary Harper. 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.

Rebecca Hwa, Philip Resnik, and Amy Weinberg. 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.

Mark Johnson, Thomas Griffiths, and Sharon Goldwater. 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.

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

Dan Klein and Christopher D Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 423–430. Association for Computational Linguistics.

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

Jonas Kuhn. 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.

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

Percy Liang, Michael I Jordan, and Dan Klein. 2009. Probabilistic grammars and hierarchical Dirichlet processes. *The handbook of applied Bayesian analysis*.

David M Magerman. 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.

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

Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 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.

Fernando Pereira and Yves Schabes. 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.

Slav Petrov and Dan Klein. 2007. Improved Inference for Unlexicalized Parsing. In *HLT-NAACL*, pages 404–411.

Slav Petrov and Dan Klein. 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.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 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.

Elias Ponvert, Jason Baldridge, and Katrin Erk. 2011. Simple Unsupervised Grammar Induction from Raw Text with Cascaded Finite State Models. In *ACL*, pages 1077–1086.

Nathan Schneider, Brendan O'Connor, Naomi Saphra, David Bamman, Manaal Faruqui, Noah A Smith, Chris Dyer, and Jason Baldridge. 2013. A framework for (under) specifying dependency syntax without overloading annotators. *arXiv preprint arXiv:1306.2091*.

Hiroyuki Shindo, Yusuke Miyao, Akinori Fujino, and Masaaki Nagata. 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.

Matthew A Taddy. 2011. On estimation and selection for topic models. *arXiv preprint arXiv:1109.4518*.

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

Naiwen Xue, Fei Xia, Fu-dong Chiou, and Marta Palmer. 2005. The Penn Chinese TreeBank: Phrase structure annotation of a large corpus. *Nat. Lang. Eng.*, 11(2):207–238, June.