Spectral Learning of Latent-Variable PCFGs

Shay B. Cohen[1], **Karl Stratos**[1], **Michael Collins**[1], **Dean P. Foster**[2], **and Lyle Ungar**[3]
[1]Dept. of Computer Science, Columbia University
[2]Dept. of Statistics/[3]Dept. of Computer and Information Science, University of Pennsylvania
{scohen,stratos,mcollins}@cs.columbia.edu, foster@wharton.upenn.edu, ungar@cis.upenn.edu

Abstract

We introduce a spectral learning algorithm for latent-variable PCFGs (Petrov et al., 2006). Under a separability (singular value) condition, we prove that the method provides consistent parameter estimates.

1 Introduction

Statistical models with hidden or latent variables are of great importance in natural language processing, speech, and many other fields. The EM algorithm is a remarkably successful method for parameter estimation within these models: it is simple, it is often relatively efficient, and it has well understood formal properties. It does, however, have a major limitation: it has no guarantee of finding the global optimum of the likelihood function. From a theoretical perspective, this means that the EM algorithm is not guaranteed to give consistent parameter estimates. From a practical perspective, problems with local optima can be difficult to deal with.

Recent work has introduced polynomial-time learning algorithms (and consistent estimation methods) for two important cases of hidden-variable models: Gaussian mixture models (Dasgupta, 1999; Vempala and Wang, 2004) and hidden Markov models (Hsu et al., 2009). These algorithms use spectral methods: that is, algorithms based on eigenvector decompositions of linear systems, in particular singular value decomposition (SVD). In the general case, learning of HMMs or GMMs is intractable (e.g., see Terwijn, 2002). Spectral methods finesse the problem of intractibility by assuming separability conditions. For example, the algorithm of Hsu et al. (2009) has a sample complexity that is polynomial in $1/\sigma$, where σ is the minimum singular value of an underlying decomposition. These methods are not susceptible to problems with local maxima, and give consistent parameter estimates.

In this paper we derive a spectral algorithm for learning of latent-variable PCFGs (L-PCFGs) (Petrov et al., 2006; Matsuzaki et al., 2005). Our method involves a significant extension of the techniques from Hsu et al. (2009). L-PCFGs have been shown to be a very effective model for natural language parsing. Under a separation (singular value) condition, our algorithm provides consistent parameter estimates; this is in contrast with previous work, which has used the EM algorithm for parameter estimation, with the usual problems of local optima.

The parameter estimation algorithm (see figure 4) is simple and efficient. The first step is to take an SVD of the training examples, followed by a projection of the training examples down to a low-dimensional space. In a second step, empirical averages are calculated on the training example, followed by standard matrix operations. On test examples, simple (tensor-based) variants of the inside-outside algorithm (figures 2 and 3) can be used to calculate probabilities and marginals of interest.

Our method depends on the following results:

• *Tensor form of the inside-outside algorithm*. Section 5 shows that the inside-outside algorithm for L-PCFGs can be written using tensors. Theorem 1 gives conditions under which the tensor form calculates inside and outside terms correctly.

• *Observable representations.* Section 6 shows that under a singular-value condition, there is an *observable form* for the tensors required by the inside-outside algorithm. By an observable form, we follow the terminology of Hsu et al. (2009) in referring to quantities that can be estimated directly from data where values for latent variables are unobserved. Theorem 2 shows that tensors derived from the observable form satisfy the conditions of theorem 1.

• *Estimating the model.* Section 7 gives an algorithm for estimating parameters of the observable representation from training data. Theorem 3 gives a sample complexity result, showing that the estimates converge to the true distribution at a rate of $1/\sqrt{M}$ where M is the number of training examples.

The algorithm is strikingly different from the EM algorithm for L-PCFGs, both in its basic form, and in its consistency guarantees. The techniques de-

Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics, pages 223–231,
Jeju, Republic of Korea, 8-14 July 2012.

veloped in this paper are quite general, and should be relevant to the development of spectral methods for estimation in other models in NLP, for example alignment models for translation, synchronous PCFGs, and so on. The tensor form of the inside-outside algorithm gives a new view of basic calculations in PCFGs, and may itself lead to new models.

2 Related Work

For work on L-PCFGs using the EM algorithm, see Petrov et al. (2006), Matsuzaki et al. (2005), Pereira and Schabes (1992). Our work builds on methods for learning of HMMs (Hsu et al., 2009; Foster et al., 2012; Jaeger, 2000), but involves several extensions: in particular in the tensor form of the inside-outside algorithm, and observable representations for the tensor form. Balle et al. (2011) consider spectral learning of finite-state transducers; Lugue et al. (2012) considers spectral learning of head automata for dependency parsing. Parikh et al. (2011) consider spectral learning algorithms of tree-structured directed bayes nets.

3 Notation

Given a matrix A or a vector v, we write $A^\top$ or $v^\top$ for the associated transpose. For any integer $n \geq 1$, we use $[n]$ to denote the set $\{1, 2, \dots n\}$. For any row or column vector $y \in \mathbb{R}^m$, we use $\text{diag}(y)$ to refer to the $(m \times m)$ matrix with diagonal elements equal to y_h for $h = 1 \dots m$, and off-diagonal elements equal to 0. For any statement Γ, we use $[[\Gamma]]$ to refer to the indicator function that is 1 if Γ is true, and 0 if Γ is false. For a random variable X, we use $\mathbf{E}[X]$ to denote its expected value.

We will make (quite limited) use of tensors:

Definition 1 *A tensor $C \in \mathbb{R}^{(m \times m \times m)}$ is a set of m^3 parameters $C_{i,j,k}$ for $i, j, k \in [m]$. Given a tensor C, and a vector $y \in \mathbb{R}^m$, we define $C(y)$ to be the $(m \times m)$ matrix with components $[C(y)]_{i,j} = \sum_{k \in [m]} C_{i,j,k} y_k$. Hence C can be interpreted as a function $C : \mathbb{R}^m \rightarrow \mathbb{R}^{(m \times m)}$ that maps a vector $y \in \mathbb{R}^m$ to a matrix $C(y)$ of dimension $(m \times m)$.*

In addition, we define the tensor $C_* \in \mathbb{R}^{(m \times m \times m)}$ for any tensor $C \in \mathbb{R}^{(m \times m \times m)}$ to have values

$$[C_*]_{i,j,k} = C_{k,j,i}$$

Finally, for vectors $x, y, z \in \mathbb{R}^m$, $xy^\top z^\top$ is the tensor $D \in \mathbb{R}^{m \times m \times m}$ where $D_{j,k,l} = x_j y_k z_l$ (this is analogous to the outer product: $[xy^\top]_{j,k} = x_j y_k$).

4 L-PCFGs: Basic Definitions

This section gives a definition of the L-PCFG formalism used in this paper. An L-PCFG is a 5-tuple $(\mathcal{N}, \mathcal{I}, \mathcal{P}, m, n)$ where:

- $\mathcal{N}$ is the set of non-terminal symbols in the grammar. $\mathcal{I} \subset \mathcal{N}$ is a finite set of *in-terminals*. $\mathcal{P} \subset \mathcal{N}$ is a finite set of *pre-terminals*. We assume that $\mathcal{N} = \mathcal{I} \cup \mathcal{P}$, and $\mathcal{I} \cap \mathcal{P} = \emptyset$. Hence we have partitioned the set of non-terminals into two subsets.
- $[m]$ is the set of possible hidden states.
- $[n]$ is the set of possible words.
- For all $a \in \mathcal{I}$, $b \in \mathcal{N}$, $c \in \mathcal{N}$, $h_1, h_2, h_3 \in [m]$, we have a context-free rule $a(h_1) \rightarrow b(h_2)\ \ c(h_3)$.
- For all $a \in \mathcal{P}$, $h \in [m]$, $x \in [n]$, we have a context-free rule $a(h) \rightarrow x$.

Hence each in-terminal $a \in \mathcal{I}$ is always the left-hand-side of a binary rule $a \rightarrow b\ c$; and each pre-terminal $a \in \mathcal{P}$ is always the left-hand-side of a rule $a \rightarrow x$. Assuming that the non-terminals in the grammar can be partitioned this way is relatively benign, and makes the estimation problem cleaner.

We define the set of possible "skeletal rules" as $\mathcal{R} = \{a \rightarrow b\ c : a \in \mathcal{I}, b \in \mathcal{N}, c \in \mathcal{N}\}$. The parameters of the model are as follows:

- For each $a \rightarrow b\ c \in \mathcal{R}$, and $h \in [m]$, we have a parameter $q(a \rightarrow b\ c|h, a)$. For each $a \in \mathcal{P}$, $x \in [n]$, and $h \in [m]$, we have a parameter $q(a \rightarrow x|h, a)$. For each $a \rightarrow b\ c \in \mathcal{R}$, and $h, h' \in [m]$, we have parameters $s(h'|h, a \rightarrow b\ c)$ and $t(h'|h, a \rightarrow b\ c)$.

These definitions give a PCFG, with rule probabilities

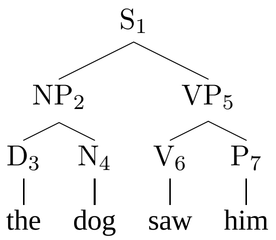
$$p(a(h_1) \rightarrow b(h_2)\ c(h_3)|a(h_1)) =$$
$$q(a \rightarrow b\ c|h_1, a) \times s(h_2|h_1, a \rightarrow b\ c) \times t(h_3|h_1, a \rightarrow b\ c)$$

and $p(a(h) \rightarrow x|a(h)) = q(a \rightarrow x|h, a)$.

In addition, for each $a \in \mathcal{I}$, for each $h \in [m]$, we have a parameter $\pi(a, h)$ which is the probability of non-terminal a paired with hidden variable h being at the root of the tree.

An L-PCFG defines a distribution over parse trees as follows. A *skeletal tree* (s-tree) is a sequence of rules $r_1 \dots r_N$ where each r_i is either of the form $a \rightarrow b\ c$ or $a \rightarrow x$. The rule sequence forms a top-down, left-most derivation under a CFG with skeletal rules. See figure 1 for an example.

A *full tree* consists of an s-tree $r_1 \dots r_N$, together with values $h_1 \dots h_N$. Each h_i is the value for



$r_1 = \text{S} \rightarrow \text{NP VP}$
$r_2 = \text{NP} \rightarrow \text{D N}$
$r_3 = \text{D} \rightarrow \text{the}$
$r_4 = \text{N} \rightarrow \text{dog}$
$r_5 = \text{VP} \rightarrow \text{V P}$
$r_6 = \text{V} \rightarrow \text{saw}$
$r_7 = \text{P} \rightarrow \text{him}$

Figure 1: An s-tree, and its sequence of rules. (For convenience we have numbered the nodes in the tree.)

the hidden variable for the left-hand-side of rule r_i. Each h_i can take any value in $[m]$.

Define a_i to be the non-terminal on the left-hand-side of rule r_i. For any $i \in \{2 \ldots N\}$ define $\text{pa}(i)$ to be the index of the rule above node i in the tree. Define $L \subset [N]$ to be the set of nodes in the tree which are the left-child of some parent, and $R \subset [N]$ to be the set of nodes which are the right-child of some parent. The probability mass function (PMF) over full trees is then

$$p(r_1 \ldots r_N, h_1 \ldots h_N) = \pi(a_1, h_1) \\ \times \prod_{i=1}^{N} q(r_i | h_i, a_i) \times \prod_{i \in L} s(h_i | h_{\text{pa}(i)}, r_{\text{pa}(i)}) \\ \times \prod_{i \in R} t(h_i | h_{\text{pa}(i)}, r_{\text{pa}(i)}) \qquad (1)$$

The PMF over s-trees is $p(r_1 \ldots r_N) = \sum_{h_1 \ldots h_N} p(r_1 \ldots r_N, h_1 \ldots h_N)$.

In the remainder of this paper, we make use of matrix form of parameters of an L-PCFG, as follows:

• For each $a \rightarrow b\ c \in \mathcal{R}$, we define $Q^{a \rightarrow b\ c} \in \mathbb{R}^{m \times m}$ to be the matrix with values $q(a \rightarrow b\ c | h, a)$ for $h = 1, 2, \ldots m$ on its diagonal, and 0 values for its off-diagonal elements. Similarly, for each $a \in \mathcal{P}$, $x \in [n]$, we define $Q^{a \rightarrow x} \in \mathbb{R}^{m \times m}$ to be the matrix with values $q(a \rightarrow x | h, a)$ for $h = 1, 2, \ldots m$ on its diagonal, and 0 values for its off-diagonal elements.

• For each $a \rightarrow b\ c \in \mathcal{R}$, we define $S^{a \rightarrow b\ c} \in \mathbb{R}^{m \times m}$ where $[S^{a \rightarrow b\ c}]_{h',h} = s(h' | h, a \rightarrow b\ c)$.

• For each $a \rightarrow b\ c \in \mathcal{R}$, we define $T^{a \rightarrow b\ c} \in \mathbb{R}^{m \times m}$ where $[T^{a \rightarrow b\ c}]_{h',h} = t(h' | h, a \rightarrow b\ c)$.

• For each $a \in \mathcal{I}$, we define the vector $\pi^a \in \mathbb{R}^m$ where $[\pi^a]_h = \pi(a, h)$.

5 Tensor Form of the Inside-Outside Algorithm

Given an L-PCFG, two calculations are central:

Inputs: s-tree $r_1 \ldots r_N$, L-PCFG $(\mathcal{N}, \mathcal{I}, \mathcal{P}, m, n)$, parameters

- $C^{a \rightarrow b\ c} \in \mathbb{R}^{(m \times m \times m)}$ for all $a \rightarrow b\ c \in \mathcal{R}$
- $c^{\infty}_{a \rightarrow x} \in \mathbb{R}^{(1 \times m)}$ for all $a \in \mathcal{P}, x \in [n]$
- $c^1_a \in \mathbb{R}^{(m \times 1)}$ for all $a \in \mathcal{I}$.

Algorithm: (calculate the f^i terms bottom-up in the tree)

- For all $i \in [N]$ such that $a_i \in \mathcal{P}$, $f^i = c^{\infty}_{r_i}$
- For all $i \in [N]$ such that $a_i \in \mathcal{I}$, $f^i = f^{\gamma} C^{r_i} (f^{\beta})$ where β is the index of the left child of node i in the tree, and γ is the index of the right child.

Return: $f^1 c^1_{a_1} = p(r_1 \ldots r_N)$

Figure 2: The tensor form for calculation of $p(r_1 \ldots r_N)$.

1. For a given s-tree $r_1 \ldots r_N$, calculate $p(r_1 \ldots r_N)$.
2. For a given input sentence $x = x_1 \ldots x_N$, calculate the marginal probabilities

$$\mu(a, i, j) = \sum_{\tau \in \mathcal{T}(x) : (a,i,j) \in \tau} p(\tau)$$

for each non-terminal $a \in \mathcal{N}$, for each (i, j) such that $1 \leq i \leq j \leq N$.

Here $\mathcal{T}(x)$ denotes the set of all possible s-trees for the sentence x, and we write $(a, i, j) \in \tau$ if non-terminal a spans words $x_i \ldots x_j$ in the parse tree τ.

The marginal probabilities have a number of uses. Perhaps most importantly, for a given sentence $x = x_1 \ldots x_N$, the parsing algorithm of Goodman (1996) can be used to find

$$\arg \max_{\tau \in \mathcal{T}(x)} \sum_{(a,i,j) \in \tau} \mu(a, i, j)$$

This is the parsing algorithm used by Petrov et al. (2006), for example. In addition, we can calculate the probability for an input sentence, $p(x) = \sum_{\tau \in \mathcal{T}(x)} p(\tau)$, as $p(x) = \sum_{a \in \mathcal{I}} \mu(a, 1, N)$.

Variants of the inside-outside algorithm can be used for problems 1 and 2. This section introduces a novel form of these algorithms, using tensors. This is the first step in deriving the spectral estimation method.

The algorithms are shown in figures 2 and 3. Each algorithm takes the following inputs:

1. A tensor $C^{a \rightarrow b\ c} \in \mathbb{R}^{(m \times m \times m)}$ for each rule $a \rightarrow b\ c$.

2. A vector $c^{\infty}_{a \rightarrow x} \in \mathbb{R}^{(1 \times m)}$ for each rule $a \rightarrow x$.

3. A vector $c^1_a \in \mathbb{R}^{(m \times 1)}$ for each $a \in \mathcal{I}$.

The following theorem gives conditions under which the algorithms are correct:

Theorem 1 *Assume that we have an L-PCFG with parameters $Q^{a \to x}, Q^{a \to b\ c}, T^{a \to b\ c}, S^{a \to b\ c}, \pi^a$, and that there exist matrices $G^a \in \mathbb{R}^{(m \times m)}$ for all $a \in \mathcal{N}$ such that each G^a is invertible, and such that:*

1. *For all rules $a \to b\ c$, $C^{a \to b\ c}(y) = G^c T^{a \to b\ c} \text{diag}(y G^b S^{a \to b\ c}) Q^{a \to b\ c} (G^a)^{-1}$*
2. *For all rules $a \to x$, $c^\infty_{a \to x} = 1^\top Q^{a \to x} (G^a)^{-1}$*
3. *For all $a \in \mathcal{I}$, $c^1_a = G^a \pi^a$*

Then: 1) The algorithm in figure 2 correctly computes $p(r_1 \ldots r_N)$ under the L-PCFG. 2) The algorithm in figure 3 correctly computes the marginals $\mu(a, i, j)$ under the L-PCFG.

Proof: See section 9.1. □

6 Estimating the Tensor Model

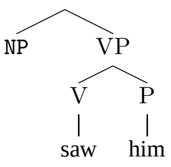
A crucial result is that it is possible to directly estimate parameters $C^{a \to b\ c}$, $c^\infty_{a \to x}$ and c^1_a that satisfy the conditions in theorem 1, from a training sample consisting of s-trees (i.e., trees where hidden variables are unobserved). We first describe random variables underlying the approach, then describe observable representations based on these random variables.

6.1 Random Variables Underlying the Approach

Each s-tree with N rules $r_1 \ldots r_N$ has N nodes. We will use the s-tree in figure 1 as a running example.

Each node has an associated rule: for example, node 2 in the tree in figure 1 has the rule NP → D N. If the rule at a node is of the form $a \to b\ c$, then there are left and right *inside trees* below the left child and right child of the rule. For example, for node 2 we have a left inside tree rooted at node 3, and a right inside tree rooted at node 4 (in this case the left and right inside trees both contain only a single rule production, of the form $a \to x$; however in the general case they might be arbitrary subtrees).

In addition, each node has an *outside* tree. For node 2, the outside tree is



Inputs: Sentence $x_1 \ldots x_N$, L-PCFG $(\mathcal{N}, \mathcal{I}, \mathcal{P}, m, n)$, parameters $C^{a \to b\ c} \in \mathbb{R}^{(m \times m \times m)}$ for all $a \to b\ c \in \mathcal{R}$, $c^\infty_{a \to x} \in \mathbb{R}^{(1 \times m)}$ for all $a \in \mathcal{P}, x \in [n]$, $c^1_a \in \mathbb{R}^{(m \times 1)}$ for all $a \in \mathcal{I}$.

Data structures:

- Each $\alpha^{a,i,j} \in \mathbb{R}^{1 \times m}$ for $a \in \mathcal{N}$, $1 \le i \le j \le N$ is a row vector of inside terms.
- Each $\beta^{a,i,j} \in \mathbb{R}^{m \times 1}$ for $a \in \mathcal{N}$, $1 \le i \le j \le N$ is a column vector of outside terms.
- Each $\mu(a, i, j) \in \mathbb{R}$ for $a \in \mathcal{N}$, $1 \le i \le j \le N$ is a marginal probability.

Algorithm:

(Inside base case) $\forall a \in \mathcal{P}, i \in [N]$, $\quad \alpha^{a,i,i} = c^\infty_{a \to x_i}$
(Inside recursion) $\forall a \in \mathcal{I}, 1 \le i < j \le N$,

$$\alpha^{a,i,j} = \sum_{k=i}^{j-1} \sum_{a \to b\ c} \alpha^{c,k+1,j} C^{a \to b\ c}(\alpha^{b,i,k})$$

(Outside base case) $\forall a \in \mathcal{I}$, $\quad \beta^{a,1,n} = c^1_a$
(Outside recursion) $\forall a \in \mathcal{N}, 1 \le i \le j \le N$,

$$\beta^{a,i,j} = \sum_{k=1}^{i-1} \sum_{b \to c\ a} C^{b \to c\ a}(\alpha^{c,k,i-1}) \beta^{b,k,j}$$

$$+ \sum_{k=j+1}^{N} \sum_{b \to a\ c} C^{b \to a\ c}_*(\alpha^{c,j+1,k}) \beta^{b,i,k}$$

(Marginals) $\forall a \in \mathcal{N}, 1 \le i \le j \le N$,

$$\mu(a, i, j) = \alpha^{a,i,j} \beta^{a,i,j} = \sum_{h \in [m]} \alpha^{a,i,j}_h \beta^{a,i,j}_h$$

Figure 3: The tensor form of the inside-outside algorithm, for calculation of marginal terms $\mu(a, i, j)$.

The outside tree contains everything in the s-tree $r_1 \ldots r_N$, excluding the subtree below node i.

Our random variables are defined as follows. First, we select a random internal node, from a random tree, as follows:

- Sample an s-tree $r_1 \ldots r_N$ from the PMF $p(r_1 \ldots r_N)$. Choose a node i uniformly at random from $[N]$.

If the rule r_i for the node i is of the form $a \to b\ c$, we define random variables as follows:

- R_1 is equal to the rule r_i (e.g., NP → D N).
- T_1 is the inside tree rooted at node i. T_2 is the inside tree rooted at the left child of node i, and T_3 is the inside tree rooted at the right child of node i.
- H_1, H_2, H_3 are the hidden variables associated with node i, the left child of node i, and the right child of node i respectively.

- A_1, A_2, A_3 are the labels for node i, the left child of node i, and the right child of node i respectively. (E.g., $A_1 = \texttt{NP}$, $A_2 = \texttt{D}$, $A_3 = \texttt{N}$.)
- O is the outside tree at node i.
- B is equal to 1 if node i is at the root of the tree (i.e., $i = 1$), 0 otherwise.

If the rule r_i for the selected node i is of the form $a \rightarrow x$, we have random variables R_1, T_1, H_1, A_1, O, B as defined above, but H_2, H_3, T_2, T_3, A_2, and A_3 are not defined.

We assume a function ψ that maps outside trees o to feature vectors $\psi(o) \in \mathbb{R}^{d'}$. For example, the feature vector might track the rule directly above the node in question, the word following the node in question, and so on. We also assume a function ϕ that maps inside trees t to feature vectors $\phi(t) \in \mathbb{R}^d$. As one example, the function ϕ might be an indicator function tracking the rule production at the root of the inside tree. Later we give formal criteria for what makes good definitions of $\psi(o)$ of $\phi(t)$. One requirement is that $d' \geq m$ and $d \geq m$.

In tandem with these definitions, we assume projection matices $U^a \in \mathbb{R}^{(d \times m)}$ and $V^a \in \mathbb{R}^{(d' \times m)}$ for all $a \in \mathcal{N}$. We then define additional random variables Y_1, Y_2, Y_3, Z as

$$Y_1 = (U^{a_1})^\top \phi(T_1) \quad Z = (V^{a_1})^\top \psi(O)$$

$$Y_2 = (U^{a_2})^\top \phi(T_2) \quad Y_3 = (U^{a_3})^\top \phi(T_3)$$

where a_i is the value of the random variable A_i. Note that Y_1, Y_2, Y_3, Z are all in $\mathbb{R}^m$.

6.2 Observable Representations

Given the definitions in the previous section, our representation is based on the following matrix, tensor and vector quantities, defined for all $a \in \mathcal{N}$, for all rules of the form $a \rightarrow b\ c$, and for all rules of the form $a \rightarrow x$ respectively:

$$\begin{aligned}
\Sigma^a &= \mathbf{E}[Y_1 Z^\top | A_1 = a] \\
D^{a \rightarrow b\ c} &= \mathbf{E}\left[[[R_1 = a \rightarrow b\ c]] Y_3 Z^\top Y_2^\top | A_1 = a\right] \\
d^\infty_{a \rightarrow x} &= \mathbf{E}\left[[[R_1 = a \rightarrow x]] Z^\top | A_1 = a\right]
\end{aligned}$$

Assuming access to functions ϕ and ψ, and projection matrices U^a and V^a, these quantities can be estimated directly from training data consisting of a set of s-trees (see section 7).

Our observable representation then consists of:

$$\begin{aligned}
C^{a \rightarrow b\ c}(y) &= D^{a \rightarrow b\ c}(y)(\Sigma^a)^{-1} && (2) \\
c^\infty_{a \rightarrow x} &= d^\infty_{a \rightarrow x}(\Sigma^a)^{-1} && (3) \\
c^1_a &= \mathbf{E}\left[[[A_1 = a]] Y_1 | B = 1\right] && (4)
\end{aligned}$$

We next introduce conditions under which these quantities satisfy the conditions in theorem 1.

The following definition will be important:

Definition 2 *For all $a \in \mathcal{N}$, we define the matrices $I^a \in \mathbb{R}^{(d \times m)}$ and $J^a \in \mathbb{R}^{(d' \times m)}$ as*

$$[I^a]_{i,h} = \mathbf{E}[\phi_i(T_1) \mid H_1 = h, A_1 = a]$$

$$[J^a]_{i,h} = \mathbf{E}[\psi_i(O) \mid H_1 = h, A_1 = a]$$

In addition, for any $a \in \mathcal{N}$, we use $\gamma^a \in \mathbb{R}^m$ to denote the vector with $\gamma^a_h = P(H_1 = h | A_1 = a)$.

The correctness of the representation will rely on the following conditions being satisfied (these are parallel to conditions 1 and 2 in Hsu et al. (2009)):

Condition 1 *$\forall a \in \mathcal{N}$, the matrices I^a and J^a are of full rank (i.e., they have rank m). For all $a \in \mathcal{N}$, for all $h \in [m]$, $\gamma^a_h > 0$.*

Condition 2 *$\forall a \in \mathcal{N}$, the matrices $U^a \in \mathbb{R}^{(d \times m)}$ and $V^a \in \mathbb{R}^{(d' \times m)}$ are such that the matrices $G^a = (U^a)^\top I^a$ and $K^a = (V^a)^\top J^a$ are invertible.*

The following lemma justifies the use of an SVD calculation as one method for finding values for U^a and V^a that satisfy condition 2:

Lemma 1 *Assume that condition 1 holds, and for all $a \in \mathcal{N}$ define*

$$\Omega^a = \mathbf{E}[\phi(T_1)\,(\psi(O))^\top | A_1 = a] \qquad (5)$$

Then if U^a is a matrix of the m left singular vectors of Ω^a corresponding to non-zero singular values, and V^a is a matrix of the m right singular vectors of Ω^a corresponding to non-zero singular values, then condition 2 is satisfied.

Proof sketch: It can be shown that $\Omega^a = I^a \mathrm{diag}(\gamma^a)(J^a)^\top$. The remainder is similar to the proof of lemma 2 in Hsu et al. (2009). □

The matrices Ω^a can be estimated directly from a training set consisting of s-trees, assuming that we have access to the functions ϕ and ψ.

We can now state the following theorem:

Theorem 2 *Assume conditions 1 and 2 are satisfied. For all $a \in \mathcal{N}$, define $G^a = (U^a)^\top I^a$. Then under the definitions in Eqs. 2-4:*

1. *For all rules $a \to b\ c$, $C^{a \to b\ c}(y) = G^c T^{a \to b\ c} \mathrm{diag}(y G^b S^{a \to b\ c}) Q^{a \to b\ c} (G^a)^{-1}$*
2. *For all rules $a \to x$, $c^\infty_{a \to x} = 1^\top Q^{a \to x} (G^a)^{-1}$.*
3. *For all $a \in \mathcal{N}$, $c^1_a = G^a \pi^a$*

Proof: The following identities hold (see section 9.2):

$$D^{a \to b\ c}(y) = G^c T^{a \to b\ c} \mathrm{diag}(y G^b S^{a \to b\ c}) Q^{a \to b\ c} \mathrm{diag}(\gamma^a)(K^a)^\top \quad (6)$$

$$d^\infty_{a \to x} = 1^\top Q^{a \to x} \mathrm{diag}(\gamma^a)(K^a)^\top \quad (7)$$

$$\Sigma^a = G^a \mathrm{diag}(\gamma^a)(K^a)^\top \quad (8)$$

$$c^1_a = G^a \pi^a \quad (9)$$

Under conditions 1 and 2, Σ^a is invertible, and $(\Sigma^a)^{-1} = ((K^a)^\top)^{-1} (\mathrm{diag}(\gamma^a))^{-1} (G^a)^{-1}$. The identities in the theorem follow immediately. □

7 Deriving Empirical Estimates

Figure 4 shows an algorithm that derives estimates of the quantities in Eqs 2, 3, and 4. As input, the algorithm takes a sequence of tuples $(r^{(i,1)}, t^{(i,1)}, t^{(i,2)}, t^{(i,3)}, o^{(i)}, b^{(i)})$ for $i \in [M]$.

These tuples can be derived from a training set consisting of s-trees $\tau_1 \dots \tau_M$ as follows:

• $\forall i \in [M]$, choose a single node j_i uniformly at random from the nodes in τ_i. Define $r^{(i,1)}$ to be the rule at node j_i. $t^{(i,1)}$ is the inside tree rooted at node j_i. If $r^{(i,1)}$ is of the form $a \to b\ c$, then $t^{(i,2)}$ is the inside tree under the left child of node j_i, and $t^{(i,3)}$ is the inside tree under the right child of node j_i. If $r^{(i,1)}$ is of the form $a \to x$, then $t^{(i,2)} = t^{(i,3)} =$ `NULL`. $o^{(i)}$ is the outside tree at node j_i. $b^{(i)}$ is 1 if node j_i is at the root of the tree, 0 otherwise.

Under this process, assuming that the s-trees $\tau_1 \dots \tau_M$ are i.i.d. draws from the distribution $p(\tau)$ over s-trees under an L-PCFG, the tuples $(r^{(i,1)}, t^{(i,1)}, t^{(i,2)}, t^{(i,3)}, o^{(i)}, b^{(i)})$ are i.i.d. draws from the joint distribution over the random variables R_1, T_1, T_2, T_3, O, B defined in the previous section.

The algorithm first computes estimates of the projection matrices U^a and V^a: following lemma 1, this is done by first deriving estimates of Ω^a, and then taking SVDs of each Ω^a. The matrices are then used to project inside and outside trees $t^{(i,1)}, t^{(i,2)}, t^{(i,3)}, o^{(i)}$ down to m-dimensional vectors $y^{(i,1)}, y^{(i,2)}, y^{(i,3)}, z^{(i)}$; these vectors are used to derive the estimates of $C^{a \to b\ c}$, $c^\infty_{a \to x}$, and c^1_a.

We now state a PAC-style theorem for the learning algorithm. First, for a given L-PCFG, we need a couple of definitions:

• Λ is the minimum absolute value of any element of the vectors/matrices/tensors c^1_a, $d^\infty_{a \to x}$, $D^{a \to b\ c}$, $(\Sigma^a)^{-1}$. (Note that Λ is a function of the projection matrices U^a and V^a as well as the underlying L-PCFG.)

• For each $a \in \mathcal{N}$, σ^a is the value of the m'th largest singular value of Ω^a. Define $\sigma = \min_a \sigma^a$.

We then have the following theorem:

Theorem 3 *Assume that the inputs to the algorithm in figure 4 are i.i.d. draws from the joint distribution over the random variables R_1, T_1, T_2, T_3, O, B, under an L-PCFG with distribution $p(r_1 \dots r_N)$ over s-trees. Define m to be the number of latent states in the L-PCFG. Assume that the algorithm in figure 4 has projection matrices $\hat{U}^a$ and $\hat{V}^a$ derived as left and right singular vectors of Ω^a, as defined in Eq. 5. Assume that the L-PCFG, together with $\hat{U}^a$ and $\hat{V}^a$, has coefficients $\Lambda > 0$ and $\sigma > 0$. In addition, assume that all elements in c^1_a, $d^\infty_{a \to x}$, $D^{a \to b\ c}$, and Σ^a are in $[-1, +1]$. For any s-tree $r_1 \dots r_N$ define $\hat{p}(r_1 \dots r_N)$ to be the value calculated by the algorithm in figure 3 with inputs $\hat{c}^1_a$, $\hat{c}^\infty_{a \to x}$, $\hat{C}^{a \to b\ c}$ derived from the algorithm in figure 4. Define R to be the total number of rules in the grammar of the form $a \to b\ c$ or $a \to x$. Define M_a to be the number of training examples in the input to the algorithm in figure 4 where $r^{i,1}$ has non-terminal a on its left-hand-side. Under these assumptions, if for all a*

$$M_a \geq \frac{128 m^2}{\left(\sqrt[2N+1]{1+\epsilon} - 1 \right)^2 \Lambda^2 \sigma^4} \log \left(\frac{2mR}{\delta} \right)$$

Then

$$1 - \epsilon \leq \left| \frac{\hat{p}(r_1 \dots r_N)}{p(r_1 \dots r_N)} \right| \leq 1 + \epsilon$$

A similar theorem (omitted for space) states that $1 - \epsilon \leq \left| \frac{\hat{\mu}(a,i,j)}{\mu(a,i,j)} \right| \leq 1 + \epsilon$ for the marginals.

The condition that $\hat{U}^a$ and $\hat{V}^a$ are derived from Ω^a, as opposed to the sample estimate $\hat{\Omega}^a$, follows Foster et al. (2012). As these authors note, similar techniques to those of Hsu et al. (2009) should be applicable in deriving results for the case where $\hat{\Omega}^a$ is used in place of Ω^a.

Proof sketch: The proof is similar to that of Foster et al. (2012). The basic idea is to first show that under the assumptions of the theorem, the estimates $\hat{c}^1_a$, $\hat{d}^\infty_{a\to x}$, $\hat{D}^{a\to b\ c}$, $\hat{\Sigma}^a$ are all close to the underlying values being estimated. The second step is to show that this ensures that $\frac{\hat{p}(r_1 \ldots r_{N'})}{p(r_1 \ldots r_{N'})}$ is close to 1. □

The method described of selecting a single tuple $(r^{(i,1)}, t^{(i,1)}, t^{(i,2)}, t^{(i,3)}, o^{(i)}, b^{(i)})$ for each s-tree ensures that the samples are i.i.d., and simplifies the analysis underlying theorem 3. In practice, an implementation should most likely use all nodes in all trees in training data; by Rao-Blackwellization we know such an algorithm would be better than the one presented, but the analysis of how much better would be challenging. It would almost certainly lead to a faster rate of convergence of $\hat{p}$ to p.

8 Discussion

There are several potential applications of the method. The most obvious is parsing with L-PCFGs.[1] The approach should be applicable in other cases where EM has traditionally been used, for example in semi-supervised learning. Latent-variable HMMs for sequence labeling can be derived as special case of our approach, by converting tagged sequences to right-branching skeletal trees.

The sample complexity of the method depends on the minimum singular values of Ω^a; these singular values are a measure of how well correlated ψ and ϕ are with the unobserved hidden variable H_1. Experimental work is required to find a good choice of values for ψ and ϕ for parsing.

9 Proofs

This section gives proofs of theorems 1 and 2. Due to space limitations we cannot give full proofs; instead we provide proofs of some key lemmas. A long version of this paper will give the full proofs.

9.1 Proof of Theorem 1

First, the following lemma leads directly to the correctness of the algorithm in figure 2:

Inputs: Training examples $(r^{(i,1)}, t^{(i,1)}, t^{(i,2)}, t^{(i,3)}, o^{(i)}, b^{(i)})$ for $i \in \{1 \ldots M\}$, where $r^{(i,1)}$ is a context free rule; $t^{(i,1)}$, $t^{(i,2)}$ and $t^{(i,3)}$ are inside trees; $o^{(i)}$ is an outside tree; and $b^{(i)} = 1$ if the rule is at the root of tree, 0 otherwise. A function ϕ that maps inside trees t to feature-vectors $\phi(t) \in \mathbb{R}^d$. A function ψ that maps outside trees o to feature-vectors $\psi(o) \in \mathbb{R}^{d'}$.

Algorithm:

Define a_i to be the non-terminal on the left-hand side of rule $r^{(i,1)}$. If $r^{(i,1)}$ is of the form $a \to b\ c$, define b_i to be the non-terminal for the left-child of $r^{(i,1)}$, and c_i to be the non-terminal for the right-child.

(Step 0: Singular Value Decompositions)

- Use the algorithm in figure 5 to calculate matrices $\hat{U}^a \in \mathbb{R}^{(d\times m)}$ and $\hat{V}^a \in \mathbb{R}^{(d'\times m)}$ for each $a \in \mathcal{N}$.

(Step 1: Projection)

- For all $i \in [M]$, compute $y^{(i,1)} = (\hat{U}^{a_i})^\top \phi(t^{(i,1)})$.
- For all $i \in [M]$ such that $r^{(i,1)}$ is of the form $a \to b\ c$, compute $y^{(i,2)} = (\hat{U}^{b_i})^\top \phi(t^{(i,2)})$ and $y^{(i,3)} = (\hat{U}^{c_i})^\top \phi(t^{(i,3)})$.
- For all $i \in [M]$, compute $z^{(i)} = (\hat{V}^{a_i})^\top \psi(o^{(i)})$.

(Step 2: Calculate Correlations)

- For each $a \in \mathcal{N}$, define $\delta_a = 1/\sum_{i=1}^M [[a_i = a]]$
- For each rule $a \to b\ c$, compute $\hat{D}^{a\to b\ c} = \delta_a \times \sum_{i=1}^M [[r^{(i,1)} = a \to b\ c]] y^{(i,3)} (z^{(i)})^\top (y^{(i,2)})^\top$
- For each rule $a \to x$, compute $\hat{d}^\infty_{a\to x} = \delta_a \times \sum_{i=1}^M [[r^{(i,1)} = a \to x]] (z^{(i)})^\top$
- For each $a \in \mathcal{N}$, compute $\hat{\Sigma}^a = \delta_a \times \sum_{i=1}^M [[a_i = a]] y^{(i,1)} (z^{(i)})^\top$

(Step 3: Compute Final Parameters)

- For all $a \to b\ c$, $\hat{C}^{a\to b\ c}(y) = \hat{D}^{a\to b\ c}(y)(\hat{\Sigma}^a)^{-1}$
- For all $a \to x$, $\hat{c}^\infty_{a\to x} = \hat{d}^\infty_{a\to x}(\hat{\Sigma}^a)^{-1}$
- For all $a \in \mathcal{I}$, $\hat{c}^1_a = \frac{\sum_{i=1}^M [[a_i = a \text{ and } b^{(i)} = 1]] y^{(i,1)}}{\sum_{i=1}^M [[b^{(i)} = 1]]}$

Figure 4: The spectral learning algorithm.

Inputs: Identical to algorithm in figure 4.

Algorithm:

- For each $a \in \mathcal{N}$, compute $\hat{\Omega}^a \in \mathbb{R}^{(d'\times d)}$ as

$$\hat{\Omega}^a = \frac{\sum_{i=1}^M [[a_i = a]] \phi(t^{(i,1)})(\psi(o^{(i)}))^\top}{\sum_{i=1}^M [[a_i = a]]}$$

and calculate a singular value decomposition of $\hat{\Omega}^a$.

- For each $a \in \mathcal{N}$, define $\hat{U}^a \in \mathbb{R}^{m\times d}$ to be a matrix of the left singular vectors of $\hat{\Omega}^a$ corresponding to the m largest singular values. Define $\hat{V}^a \in \mathbb{R}^{m\times d'}$ to be a matrix of the right singular vectors of $\hat{\Omega}^a$ corresponding to the m largest singular values.

Figure 5: Singular value decompositions.

[1] Parameters can be estimated using the algorithm in figure 4; for a test sentence $x_1 \ldots x_N$ we can first use the algorithm in figure 3 to calculate marginals $\mu(a, i, j)$, then use the algorithm of Goodman (1996) to find $\arg\max_{\tau \in \mathcal{T}(x)} \sum_{(a,i,j)\in\tau} \mu(a, i, j)$.

Lemma 2 *Assume that conditions 1-3 of theorem 1 are satisfied, and that the input to the algorithm in figure 2 is an s-tree $r_1 \dots r_N$. Define a_i for $i \in [N]$ to be the non-terminal on the left-hand-side of rule r_i, and t_i for $i \in [N]$ to be the s-tree with rule r_i at its root. Finally, for all $i \in [N]$, define the row vector $b^i \in \mathbb{R}^{(1 \times m)}$ to have components*

$$b^i_h = P(T_i = t_i | H_i = h, A_i = a_i)$$

for $h \in [m]$. Then for all $i \in [N]$, $f^i = b^i (G^{(a_i)})^{-1}$. It follows immediately that

$$f^1 c^1_{a_1} = b^1 (G^{(a_1)})^{-1} G^{a_1} \pi_{a_1} = p(r_1 \dots r_N) \quad \Box$$

This lemma shows a direct link between the vectors f^i calculated in the algorithm, and the terms b^i_h, which are terms calculated by the conventional inside algorithm: each f^i is a linear transformation (through G^{a_i}) of the corresponding vector b^i.

Proof: The proof is by induction.

First consider the base case. For any leaf—i.e., for any i such that $a_i \in \mathcal{P}$—we have $b^i_h = q(r_i | h, a_i)$, and it is easily verified that $f^i = b^i (G^{(a_i)})^{-1}$.

The inductive case is as follows. For all $i \in [N]$ such that $a_i \in \mathcal{I}$, by the definition in the algorithm,

$$\begin{aligned} f^i &= f^\gamma C^{r_i}(f^\beta) \\ &= f^\gamma G^{a_\gamma} T^{r_i} \mathrm{diag}(f^\beta G^{a_\beta} S^{r_i}) Q^{r_i} (G^{a_i})^{-1} \end{aligned}$$

Assuming by induction that $f^\gamma = b^\gamma (G^{(a_\gamma)})^{-1}$ and $f^\beta = b^\beta (G^{(a_\beta)})^{-1}$, this simplifies to

$$f^i = \kappa^r \mathrm{diag}(\kappa^l) Q^{r_i} (G^{a_i})^{-1} \tag{10}$$

where $\kappa^r = b^\gamma T^{r_i}$, and $\kappa^l = b^\beta S^{r_i}$. κ^r is a row vector with components $\kappa^r_h = \sum_{h' \in [m]} b^\gamma_{h'} T^{r_i}_{h',h} = \sum_{h' \in [m]} b^\gamma_{h'} t(h' | h, r_i)$. Similarly, κ^l is a row vector with components equal to $\kappa^l_h = \sum_{h' \in [m]} b^\beta_{h'} S^{r_i}_{h',h} = \sum_{h' \in [m]} b^\beta_{h'} s(h' | h, r_i)$. It can then be verified that $\kappa^r \mathrm{diag}(\kappa^l) Q^{r_i}$ is a row vector with components equal to $\kappa^r_h \kappa^l_h q(r_i | h, a_i)$.

But $b^i_h = q(r_i | h, a_i) \times \left(\sum_{h' \in [m]} b^\gamma_{h'} t(h' | h, r_i) \right) \times \left(\sum_{h' \in [m]} b^\beta_{h'} s(h' | h, r_i) \right) = q(r_i | h, a_i) \kappa^r_h \kappa^l_h$, hence $\kappa^r \mathrm{diag}(\kappa^l) Q^{r_i} = b^i$ and the inductive case follows immediately from Eq. 10. □

Next, we give a similar lemma, which implies the correctness of the algorithm in figure 3:

Lemma 3 *Assume that conditions 1-3 of theorem 1 are satisfied, and that the input to the algorithm in figure 3 is a sentence $x_1 \dots x_N$. For any $a \in \mathcal{N}$, for any $1 \leq i \leq j \leq N$, define $\bar{\alpha}^{a,i,j} \in \mathbb{R}^{(1 \times m)}$ to have components $\bar{\alpha}^{a,i,j}_h = p(x_i \dots x_j | h, a)$ for $h \in [m]$. In addition, define $\bar{\beta}^{a,i,j} \in \mathbb{R}^{(m \times 1)}$ to have components $\bar{\beta}^{a,i,j}_h = p(x_1 \dots x_{i-1}, a(h), x_{j+1} \dots x_N)$ for $h \in [m]$. Then for all $i \in [N]$, $\alpha^{a,i,j} = \bar{\alpha}^{a,i,j} (G^a)^{-1}$ and $\beta^{a,i,j} = G^a \bar{\beta}^{a,i,j}$. It follows that for all (a, i, j),*

$$\begin{aligned} \mu(a,i,j) &= \bar{\alpha}^{a,i,j} (G^a)^{-1} G^a \bar{\beta}^{a,i,j} = \bar{\alpha}^{a,i,j} \bar{\beta}^{a,i,j} \\ &= \sum_h \bar{\alpha}^{a,i,j}_h \bar{\beta}^{a,i,j}_h = \sum_{\tau \in \mathcal{T}(x) : (a,i,j) \in \tau} p(\tau) \quad \Box \end{aligned}$$

Thus the vectors $\alpha^{a,i,j}$ and $\beta^{a,i,j}$ are linearly related to the vectors $\bar{\alpha}^{a,i,j}$ and $\bar{\beta}^{a,i,j}$, which are the inside and outside terms calculated by the conventional form of the inside-outside algorithm.

The proof is by induction, and is similar to the proof of lemma 2; for reasons of space it is omitted.

9.2 Proof of the Identity in Eq. 6

We now prove the identity in Eq. 6, used in the proof of theorem 2. For reasons of space, we do not give the proofs of identities 7-9: the proofs are similar.

The following identities can be verified:

$$\begin{aligned} P(R_1 = a \rightarrow b\ c | H_1 = h, A_1 = a) &= q(a \rightarrow b\ c | h, a) \\ \mathbf{E}\left[Y_{3,j} | H_1 = h, R_1 = a \rightarrow b\ c \right] &= E^{a \rightarrow b\ c}_{j,h} \\ \mathbf{E}\left[Z_k | H_1 = h, R_1 = a \rightarrow b\ c \right] &= K^a_{k,h} \\ \mathbf{E}\left[Y_{2,l} | H_1 = h, R_1 = a \rightarrow b\ c \right] &= F^{a \rightarrow b\ c}_{l,h} \end{aligned}$$

where $E^{a \rightarrow b\ c} = G^c T^{a \rightarrow b\ c}$, $F^{a \rightarrow b\ c} = G^b S^{a \rightarrow b\ c}$.

Y_3, Z and Y_2 are independent when conditioned on H_1, R_1 (this follows from the independence assumptions in the L-PCFG), hence

$$\begin{aligned} & \mathbf{E}\left[[[R_1 = a \rightarrow b\ c]] Y_{3,j} Z_k Y_{2,l} \mid H_1 = h, A_1 = a \right] \\ = \; & q(a \rightarrow b\ c | h, a) E^{a \rightarrow b\ c}_{j,h} K^a_{k,h} F^{a \rightarrow b\ c}_{l,h} \end{aligned}$$

Hence (recall that $\gamma^a_h = P(H_1 = h | A_1 = a)$),

$$\begin{aligned} & D^{a \rightarrow b\ c}_{j,k,l} = \mathbf{E}\left[[[R_1 = a \rightarrow b\ c]] Y_{3,j} Z_k Y_{2,l} \mid A_1 = a \right] \\ = \; & \sum_h \gamma^a_h \mathbf{E}\left[[[R_1 = a \rightarrow b\ c]] Y_{3,j} Z_k Y_{2,l} \mid H_1 = h, A_1 = a \right] \\ = \; & \sum_h \gamma^a_h q(a \rightarrow b\ c | h, a) E^{a \rightarrow b\ c}_{j,h} K^a_{k,h} F^{a \rightarrow b\ c}_{l,h} \end{aligned} \tag{11}$$

from which Eq. 6 follows. □

Acknowledgements: Columbia University gratefully acknowledges the support of the Defense Advanced Research Projects Agency (DARPA) Machine Reading Program under Air Force Research Laboratory (AFRL) prime contract no. FA8750-09-C-0181. 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 DARPA, AFRL, or the US government. Shay Cohen was supported by the National Science Foundation under Grant #1136996 to the Computing Research Association for the CIFellows Project. Dean Foster was supported by National Science Foundation grant 1106743.

References

B. Balle, A. Quattoni, and X. Carreras. 2011. A spectral learning algorithm for finite state transducers. In *Proceedings of ECML*.

S. Dasgupta. 1999. Learning mixtures of Gaussians. In *Proceedings of FOCS*.

Dean P. Foster, Jordan Rodu, and Lyle H. Ungar. 2012. Spectral dimensionality reduction for hmms. arXiv:1203.6130v1.

J. Goodman. 1996. Parsing algorithms and metrics. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, pages 177–183. Association for Computational Linguistics.

D. Hsu, S. M. Kakade, and T. Zhang. 2009. A spectral algorithm for learning hidden Markov models. In *Proceedings of COLT*.

H. Jaeger. 2000. Observable operator models for discrete stochastic time series. *Neural Computation*, 12(6).

F. M. Lugue, A. Quattoni, B. Balle, and X. Carreras. 2012. Spectral learning for non-deterministic dependency parsing. In *Proceedings of EACL*.

T. Matsuzaki, Y. Miyao, and J. 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.

A. Parikh, L. Song, and E. P. Xing. 2011. A spectral algorithm for latent tree graphical models. In *Proceedings of The 28th International Conference on Machine Learningy (ICML 2011)*.

F. Pereira and Y. Schabes. 1992. Inside-outside reestimation from partially bracketed corpora. In *Proceedings of the 30th Annual Meeting of the Association for Computational Linguistics*, pages 128–135, Newark, Delaware, USA, June. Association for Computational Linguistics.

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

S. A. Terwijn. 2002. On the learnability of hidden markov models. In *Grammatical Inference: Algorithms and Applications (Amsterdam, 2002)*, volume 2484 of *Lecture Notes in Artificial Intelligence*, pages 261–268, Berlin. Springer.

S. Vempala and G. Wang. 2004. A spectral algorithm for learning mixtures of distributions. *Journal of Computer and System Sciences*, 68(4):841–860.