Spectral Learning of Latent-Variable PCFGs

Shay B. Cohen[1], Karl Stratos[1], Michael Collins[1], Dean P. Foster[2], 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, Barrett, Thibaux, & Klein, 2006; Matsuzaki, Miyao, & Tsujii, 2005). Under a separability (singular value) condition, we prove that the method provides consistent parameter estimates. Our result rests on three theorems: the first gives a tensor form of the inside-outside algorithm for PCFGs; the second shows that the required tensors can be estimated directly from training examples where hidden-variable values are missing; the third gives a PAC-style convergence bound for the estimation method.

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 a polynomial-time learning algorithm (and a consistent estimation method) for an important case of hidden-variable models: hidden Markov models (Hsu, Kakade, & Zhang, 2009). This algorithm uses a spectral method: that is, an algorithm based on eigenvector decompositions of linear systems, in particular singular value decomposition (SVD). In the general case, learning of HMMs is intractable (e.g., see Terwijn, 2002). The spectral method finesses the problem of intractibility by assuming separability conditions. More precisely, 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. The HMM learning algorithm is not susceptible to problems with local maxima, and gives 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 7) 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 4 and 5) 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 6.1 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 7 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 8 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 developed 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, Rodu, & Ungar, 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, Quattoni, and Carreras (2011) consider spectral learning of finite-state transducers; Lugue, Quattoni, Balle, and Carreras (2012) considers spectral learning of head automata for dependency parsing. Parikh, Song, and Xing (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 $\{1, 2, \ldots n\}$.

We use $\mathbb{R}^{m \times 1}$ to denote the space of m-dimensional column vectors, and $\mathbb{R}^{1 \times m}$ to denote the space of m-dimensional row vectors. We use $\mathbb{R}^m$ to denote the space of m-dimensional vectors, where the vector in question can be either a row or column vector. 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 $h = 1 \ldots m$, 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 vectors $y^1 \in \mathbb{R}^m$ and $y^2 \in \mathbb{R}^m$, we define $C(y^1, y^2)$ to be the m-dimensional row vector with components*

$$[C(y^1, y^2)]_i = \sum_{j \in [m], k \in [m]} C_{i,j,k} y^1_j y^2_k$$

Hence C can be interpreted as a function $C : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}^{1 \times m}$ that maps vectors y^1 and y^2 to a row vector $C(y^1, y^2) \in \mathbb{R}^{1 \times m}$.

In addition, we define the tensor $C_{(1,2)} \in \mathbb{R}^{(m \times m \times m)}$ for any tensor $C \in \mathbb{R}^{(m \times m \times m)}$ to be the function $C_{(1,2)} : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}^{m \times 1}$ defined as

$$[C_{(1,2)}(y^1, y^2)]_k = \sum_{i \in [m], j \in [m]} C_{i,j,k} y^1_i y^2_j$$

Similarly, for any tensor C we define $C_{(1,3)} : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}^{m \times 1}$ as

$$[C_{(1,3)}(y^1, y^2)]_j = \sum_{i \in [m], k \in [m]} C_{i,j,k} y^1_i y^2_k$$

Note that $C_{(1,2)}(y^1, y^2)$ and $C_{(1,3)}(y^1, y^2)$ are both **column** *vectors.*

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 $D_{i,j,k} = x_i y_j z_k$ (this is analogous to the outer product: $[xy^\top]_{i,j} = x_i y_j$).

4. L-PCFGs

In this section we describe latent-variable PCFGs (L-PCFGs), as used for example by (Matsuzaki et al., 2005; Petrov et al., 2006). We first give the basic definitions for L-PCFGs, and then describe the underlying motivation for them.

4.1 Basic Definitions

This section gives a definition of the L-PCFG formalism used in this paper. An L-PCFG is an 8-tuple $(\mathcal{N}, \mathcal{I}, \mathcal{P}, m, n, t, q, \pi)$ 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 $a(h) \rightarrow x$.
- For all $a \in \mathcal{I}, b \in \mathcal{N}, c \in \mathcal{N}$, and $h_1, h_2, h_3 \in [m]$, we have a parameter $t(a \rightarrow b\ c, h_2, h_3 | h_1, a)$.
- For all $a \in \mathcal{P}$, $x \in [n]$, and $h \in [m]$, we have a parameter $q(a \rightarrow x | h, a)$.
- For all $a \in \mathcal{I}$ and $h \in [m]$, 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.

Note that 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.

For convenience we define the set of possible "skeletal rules" as $\mathcal{R} = \{a \rightarrow b\ c : a \in \mathcal{I}, b \in \mathcal{N}, c \in \mathcal{N}\}$.

These definitions give a PCFG, with rule probabilities

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

and

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

Remark 1 *In the previous paper on this work (Cohen, Stratos, Collins, Foster, & Ungar, 2012), we considered an L-PCFG model where*

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

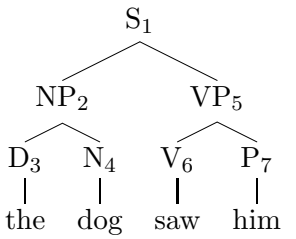
In this model the random variables h_2 and h_3 are assumed to be conditionally independent given h_1 and $a \rightarrow b\ c$.

In this paper we consider a model where

$$p(a(h_1) \rightarrow b(h_2)\ c(h_3) | a(h_1)) = t(a \rightarrow b\ c, h_2, h_3, | h_1, a) \tag{1}$$

That is, we do **not** *assume that the random variables h_2 and h_3 are independent when conditioning on h_1 and $a \rightarrow b\ c$. This is also the model considered by (Petrov et al., 2006; Matsuzaki et al., 2005).*

Note however that the algorithms in this paper are the same as those in (Cohen et al., 2012): we have simply proved that the algorithms give consistent estimators for the model form in Eq. 1.



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

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

As in usual PCFGs, the probability of an entire tree is calculated as the product of its rule probabilities. We now give more detail for these calculations.

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 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 $i \in [N]$ such that $a_i \in \mathcal{I}$ (i.e., a_i is an in-terminal, and rule r_i is of the form $a \rightarrow b\ c$) define $h_i^{(2)}$ to be the hidden variable value associated with the left child of the rule r_i, and $h_i^{(3)}$ to be the hidden variable value associated with the right child. The probability mass function (PMF) over full trees is then

$$p(r_1 \dots r_N, h_1 \dots h_N) = \pi(a_1, h_1) \times \prod_{i:a_i \in \mathcal{I}} t(r_i, h_i^{(2)}, h_i^{(3)} | h_i, a_i) \times \prod_{i:a_i \in \mathcal{P}} q(r_i | h_i, a_i) \quad (2)$$

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

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 $T^{a \rightarrow b\ c} \in \mathbb{R}^{m \times m \times m}$ to be the tensor with values

$$T^{a \rightarrow b\ c}_{h_1, h_2, h_3} = t(a \rightarrow b\ c, h_2, h_3 | a, h_1)$$

- For each $a \in \mathcal{P}$, $x \in [n]$, we define $q_{a \rightarrow x} \in \mathbb{R}^{1 \times m}$ to be the row vector with values

$$[q_{a \rightarrow x}]_h = q(a \rightarrow x | h, a)$$

  for $h = 1, 2, \dots m$.

- For each $a \in \mathcal{I}$, we define the column vector $\pi^a \in \mathbb{R}^{m \times 1}$ where $[\pi^a]_h = \pi(a, h)$.

4.2 Application of L-PCFGs to Natural Language Parsing

L-PCFGs have been shown to be a very useful model for natural language parsing (Matsuzaki et al., 2005; Petrov et al., 2006). In this section we describe the basic approach.

We assume a training set consisting of sentences paired with parse trees, which are similar to the skeletal tree shown in figure 1. A naive approach to parsing would simply read off a PCFG from the training set: the resulting grammar would have rules such as

$$\begin{aligned}
\texttt{S} &\rightarrow \texttt{NP VP} \\
\texttt{NP} &\rightarrow \texttt{D N} \\
\texttt{VP} &\rightarrow \texttt{V NP} \\
\texttt{D} &\rightarrow \texttt{the} \\
\texttt{N} &\rightarrow \texttt{dog}
\end{aligned}$$

and so on. Given a test sentence, the most likely parse under the PCFG can be found using dynamic programming algorithms.

Unfortunately, simple "vanilla" PCFGs induced from treebanks such as the Penn treebank (Marcus, Santorini, & Marcinkiewicz, 1993) typically give very poor parsing performance. A critical issue is that the set of non-terminals in the resulting grammar (`S, NP, VP, PP, D, N`, etc.) is often quite small. The resulting PCFG therefore makes very strong independence assumptions, failing to capture important statistical properties of parse trees.

In response to this issue, a number of PCFG-based models have been developed which make use of grammars with *refined* non-terminals. For example, in lexicalized models (Collins, 1997; Charniak, 1997), non-terminals such as `S` are replaced with non-terminals such as `S-sleeps`: the non-terminals track some lexical item (in this case *sleeps*), in addition to the syntactic category. For example, the parse tree in figure 1 would include rules

$$\begin{aligned}
\texttt{S-saw} &\rightarrow \texttt{NP-dog VP-saw} \\
\texttt{NP-dog} &\rightarrow \texttt{D-the N-dog} \\
\texttt{VP-saw} &\rightarrow \texttt{V-saw P-him} \\
\texttt{D-the} &\rightarrow \texttt{the} \\
\texttt{N-dog} &\rightarrow \texttt{dog} \\
\texttt{V-saw} &\rightarrow \texttt{saw} \\
\texttt{P-him} &\rightarrow \texttt{him}
\end{aligned}$$

In this case the number of non-terminals in the grammar increases dramatically, but with appropriate smoothing of parameter estimates lexicalized models perform at much higher accuracy than vanilla PCFGs.

As another example, Johnson (1998) describes an approach where non-terminals are refined to also include the non-terminal one level up in the tree; for example rules such as

$$\texttt{S} \rightarrow \texttt{NP VP}$$

are replaced by rules such as

$$\texttt{S-ROOT} \rightarrow \texttt{NP-S VP-S}$$

Here NP-S corresponds to an NP non-terminal whose parent is S; VP-S corresponds to a VP whose parent is S; S-ROOT corresponds to an S which is at the root of the tree. This simple modification leads to significant improvements over a vanilla PCFG.

Klein and Manning (2003) develop this approach further, introducing annotations corresponding to parents and siblings in the tree, together with other information, resulting in a parser whose performance is just below the lexicalized models of Collins (1997), Charniak (1997).

The approaches of Collins (1997), Charniak (1997), Johnson (1998), Klein and Manning (2003) all use hand-constructed rules to enrich the set of non-terminals in the PCFG. A natural question is whether refinements to non-terminals can be learned automatically. Matsuzaki et al. (2005), Petrov et al. (2006) addressed this question through the use of L-PCFGs in conjunction with the EM algorithm. The basic idea is to allow each non-terminal in the grammar to have m possible *latent* values. For example, with $m = 8$ we would replace the non-terminal S with non-terminals S-1, S-2, ..., S-8, and we would replace rules such as

$$\texttt{S} \rightarrow \texttt{NP VP}$$

with rules such as

$$\texttt{S-4} \rightarrow \texttt{NP-3 VP-2}$$

The latent values are of course unobserved in the training data (the treebank), but they can be treated as latent variables in a PCFG-based model, and the parameters of the model can be estimated using the EM algorithm. More specifically, given training examples consisting of skeletal trees of the form $t^{(i)} = (r_1^{(i)}, r_2^{(i)}, \ldots, r_{N_i}^{(i)})$, for $i = 1 \ldots M$, where N_i is the number of rules in the i'th tree, the log-likelihood of the training data is

$$\sum_{i=1}^{M} \log p(r_1^{(i)} \ldots r_{N_i}^{(i)}) = \sum_{i=1}^{M} \log \sum_{h_1 \ldots h_{N_i}} p(r_1^{(i)} \ldots r_{N_i}^{(i)}, h_1 \ldots h_{N_i})$$

where $p(r_1^{(i)} \ldots r_{N_i}^{(i)}, h_1 \ldots h_{N_i})$ is as defined in Eq. 2. The EM algorithm is guaranteed to converge to a local maximum of the log-likelihood function. Once the parameters of the L-PCFG have been estimated, the algorithm of Goodman (1996) can be used to parse test-data sentences using the L-PCFG: see section 4.3 for more details. Matsuzaki et al. (2005), Petrov et al. (2006) show very good performance for these methods.

4.3 Basic Algorithms for L-PCFGs: Variants of the Inside-Outside Algorithm

Variants of the inside-outside algorithm (Baker, 1979) can be used for basic calculations in L-PCFGs, in particular for calculations that involve marginalization over the values for the hidden variables.

To be more specific, given an L-PCFG, two calculations are central:

1. For a given s-tree $r_1 \ldots r_N$, calculate $p(r_1 \ldots r_N) = \sum_{h_1 \ldots h_N} p(r_1 \ldots r_N, h_1 \ldots h_N)$.
2. For a given input sentence $x = x_1 \ldots x_N$, calculate the marginal probabilities

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

Inputs: s-tree $r_1 \dots r_N$, L-PCFG $(\mathcal{N}, \mathcal{I}, \mathcal{P}, m, n, t, q, \pi)$, with parameters

- $t(a \rightarrow b\ c, h_2, h_3 | h_1, a)$ for all $a \rightarrow b\ c \in \mathcal{R}$, $h_1, h_2, h_3 \in [m]$.
- $q(a \rightarrow x | h, a)$ for all $a \in \mathcal{P}, x \in [n], h \in [m]$
- $\pi(a, h)$ for all $a \in \mathcal{I}$, $h \in [m]$.

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

- For all $i \in [N]$ such that $a_i \in \mathcal{P}$, for all $h \in [m]$, $b^i_h = q(r_i | h, a_i)$
- For all $i \in [N]$ such that $a_i \in \mathcal{I}$, for all $h \in [m]$, $b^i_h = \sum_{h_2, h_3} t(r_i, h_2, h_3 | h, a_i) b^\beta_{h_2} b^\gamma_{h_3}$ 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: $\sum_h b^1_h \pi(a, h) = p(r_1 \dots r_N)$

Figure 2: The conventional inside-outside algorithm for calculation of $p(r_1 \dots r_N)$.

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 $x_i \dots x_j$ in the parse tree τ.

The marginal probabilities have a number of uses. Perhaps most importantly, for a given sentence $x = x_1 \dots x_N$, 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.[1] In addition, we can calculate the probability for an input sentence, $p(x) = \sum_{\tau \in \mathcal{T}(x)} p(\tau)$, as $p(x) = \sum_{a \in \mathcal{I}} \mu(a, 1, N)$.

Figures 2 and 3 give the conventional (as opposed to tensor) form of inside-outside algorithms for these two problems. In the next section we describe the tensor form. The algorithm in figure 2 uses dynamic programming to compute

$$p(r_1 \dots r_N) = \sum_{h_1 \dots h_N} p(r_1 \dots r_N, h_1 \dots h_N)$$

for a given parse tree $r_1 \dots r_N$. The algorithm in figure 3 uses dynamic programming to compute marginal terms.

5. Roadmap

The next three sections of the paper derive the spectral algorithm for learning of L-PCFGs. The structure of these sections is as follows:

- Section 6 introduces a *tensor form* of the inside-outside algorithms for L-PCFGs. This is analogous to the matrix form for hidden Markov models (see (Jaeger, 2000), and in particular Lemma 1 of (Hsu et al., 2009)), and is also related to the use of tensors in spectral algorithms for directed graphical models (Parikh et al., 2011).

1. Note that finding $\arg\max_{\tau \in \mathcal{T}(x)} p(\tau)$, where $p(\tau) = \sum_{h_1 \dots h_N} p(\tau, h_1 \dots h_N)$, is NP hard, hence the use of Goodman's algorithm.

Inputs: Sentence $x_1 \ldots x_N$, L-PCFG $(\mathcal{N}, \mathcal{I}, \mathcal{P}, m, n, t, q, \pi)$, with parameters

- $t(a \rightarrow b\ c, h_2, h_3 | h_1, a)$ for all $a \rightarrow b\ c \in \mathcal{R}$, $h_1, h_2, h_3 \in [m]$.
- $q(a \rightarrow x | h, a)$ for all $a \in \mathcal{P}, x \in [n], h \in [m]$
- $\pi(a, h)$ for all $a \in \mathcal{I}$, $h \in [m]$.

Data structures:

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

Algorithm:

(Inside base case) $\forall a \in \mathcal{P}, i \in [N], h \in [m]$ $\bar{\alpha}_h^{a,i,i} = q(a \rightarrow x_i | h, a)$

(Inside recursion) $\forall a \in \mathcal{I}, 1 \leq i < j \leq N, h \in [m]$

$$\bar{\alpha}_h^{a,i,j} = \sum_{k=i}^{j-1} \sum_{a \rightarrow b\ c} \sum_{h_2 \in [m]} \sum_{h_3 \in [m]} t(a \rightarrow b\ c, h_2, h_3 | h, a) \times \bar{\alpha}_{h_2}^{b,i,k} \times \bar{\alpha}_{h_3}^{c,k+1,j}$$

(Outside base case) $\forall a \in \mathcal{I}, h \in [m]$ $\bar{\beta}_h^{a,1,n} = \pi(a, h)$

(Outside recursion) $\forall a \in \mathcal{N}, 1 \leq i \leq j \leq N, h \in [m]$

$$\bar{\beta}_h^{a,i,j} = \sum_{k=1}^{i-1} \sum_{b \rightarrow c\ a} \sum_{h_2 \in [m]} \sum_{h_3 \in [m]} t(b \rightarrow c\ a, h_3, h | h_2, b) \times \bar{\beta}_{h_2}^{b,k,j} \times \bar{\alpha}_{h_3}^{c,k,i-1}$$

$$+ \sum_{k=j+1}^{N} \sum_{b \rightarrow a\ c} \sum_{h_2 \in [m]} \sum_{h_3 \in [m]} t(b \rightarrow a\ c, h, h_3 | h_2, b) \times \bar{\beta}_{h_2}^{b,i,k} \times \bar{\alpha}_{h_3}^{c,j+1,k}$$

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

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

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

- Section 7 derives an *observable form* for the tensors required by algorithms of section 6. The implication of this result is that the required tensors can be estimated directly from training data consisting of skeletal trees.

- Section 8 gives the algorithm for estimation of the tensors from a training sample, and gives a PAC-style generalization bound for the approach.

6. Tensor Form of the Inside-Outside Algorithm

This section first gives a tensor form of the inside-outside algorithms for L-PCFGs, then give an illustrative example.

Inputs: s-tree $r_1 \dots r_N$, 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 $a \in \mathcal{I}$.

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 $a_i \in \mathcal{I}$, $f^i = C^{r_i}(f^{\beta}, f^{\gamma})$ 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: $f^1 c^1_{a_1} = p(r_1 \dots r_N)$

Figure 4: The tensor form for calculation of $p(r_1 \dots r_N)$.

6.1 The Tensor-Form Algorithms

Recall the two calculations for L-PCFGs introduced in section 4.3:

1. For a given s-tree $r_1 \dots r_N$, calculate $p(r_1 \dots r_N)$.

2. For a given input sentence $x = x_1 \dots x_N$, 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 $1 \leq i \leq j \leq N$, where $\mathcal{T}(x)$ 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 $x_i \dots x_j$ in the parse tree τ.

The tensor form of the inside-outside algorithms for these two problems are shown in figures 4 and 5. Each algorithm takes the following inputs:

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

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

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 $q_{a \to x}$, $T^{a \to b\ c}$, π^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 $a \to b\ c$, $C^{a \to b\ c}(y^1, y^2) = \left(T^{a \to b\ c}(y^1 G^b, y^2 G^c)\right)(G^a)^{-1}$*

2. *For all rules $a \to x$, $c^{\infty}_{a \to x} = q_{a \to x}(G^a)^{-1}$*

3. *For all $a \in \mathcal{I}$, $c^{1}_{a} = G^a \pi^a$*

Then: 1) The algorithm in figure 4 correctly computes $p(r_1 \dots r_N)$ under the L-PCFG. 2) The algorithm in figure 5 correctly computes the marginals $\mu(a, i, j)$ under the L-PCFG.

Proof: see section A.1. The next section (section 6.2) gives an example that illustrates the basic intuition behind the proof. □

Remark 2 *It is easily verified (see also the example in section 6.2), that if the inputs to the tensor-form algorithms are of the following form (equivalently, the matrices G^a for all a are equal to the identity matrix):*

1. *For all rules $a \to b\ c$, $C^{a \to b\ c}(y^1, y^2) = T^{a \to b\ c}(y^1, y^2)$*
2. *For all rules $a \to x$, $c^{\infty}_{a \to x} = q_{a \to x}$*
3. *For all $a \in \mathcal{I}$, $c^1_a = \pi^a$*

then the algorithms in figures 4 and 5 are identical to the algorithms in figures 2 and 3 respectively. More precisely, we have the identities

$$b^i_h = f^i_h$$

for the quantities in figures 2 and 4, and

$$\bar{\alpha}^{a,i,j}_h = \alpha^{a,i,j}_h$$
$$\bar{\beta}^{a,i,j}_h = \beta^{a,i,j}_h$$

for the quantities in figures 3 and 5.

The theorem shows, however, that it is sufficient[2] *to have parameters that are equal to $T^{a \to b\ c}$, $q_{a \to x}$ and π^a up to linear transforms defined by the matrices G^a for all non-terminals a. The linear transformations add an extra degree of freedom that is crucial in what follows in this paper: in the next section, on observable representations, we show that it is possible to directly estimate values for $C^{a \to b\ c}$, $c^{\infty}_{a \to x}$ and c^1_a that satisfy the conditions of the theorem, but where the matrices G^a are not the identity matrix.*

The key step in the proof of the theorem (see section A.1) is to show that under the assumptions of the theorem we have the identities

$$f^i = b^i (G^a)^{-1}$$

for figures 2 and 4, and

$$\alpha^{a,i,j} = \bar{\alpha}^{a,i,j} (G^a)^{-1}$$
$$\beta^{a,i,j} = G^a \bar{\beta}^{a,i,j}$$

for figures 3 and 5. Thus the quantities calculated by the tensor-form algorithms are equivalent to the quantities calculated by the conventional algorithms, up to linear transforms. The linear transforms and their inverses cancel in useful ways: for example in the output from figure 4 we have

$$\mu(a, i, j) = \alpha^{a,i,j} \beta^{a,i,j} = \bar{\alpha}^{a,i,j} (G^a)^{-1} G^a \bar{\beta}^{a,i,j} = \sum_h \bar{\alpha}^{a,i,j}_h \bar{\beta}^{a,i,j}_h$$

showing that the marginals calculated by the conventional and tensor-form algorithms are identical.

2. Assuming that the goal is to calculate $p(r_1 \dots r_N)$ for any skeletal tree, or marginal terms $\mu(a, i, j)$.

Inputs: Sentence $x_1 \dots x_N$, 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 $a \in \mathcal{N}$, $1 \leq i \leq j \leq N$ is a row vector of inside terms.
- Each $\beta^{a,i,j} \in \mathbb{R}^{m \times 1}$ for $a \in \mathcal{N}$, $1 \leq i \leq j \leq N$ is a column vector of outside terms.
- Each $\mu(a, i, j) \in \mathbb{R}$ for $a \in \mathcal{N}$, $1 \leq i \leq j \leq N$ is a marginal probability.

Algorithm:

(Inside base case) $\forall a \in \mathcal{P}, i \in [N]$, $\alpha^{a,i,i} = c^{\infty}_{a \to x_i}$

(Inside recursion) $\forall a \in \mathcal{I}, 1 \leq i < j \leq N$,

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

(Outside base case) $\forall a \in \mathcal{I}$, $\beta^{a,1,n} = c^{1}_{a}$

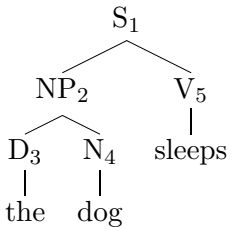
(Outside recursion) $\forall a \in \mathcal{N}, 1 \leq i \leq j \leq N$,

$$\beta^{a,i,j} = \sum_{k=1}^{i-1} \sum_{b \to c\ a} C^{b \to c\ a}_{(1,2)}(\beta^{b,k,j}, \alpha^{c,k,i-1}) + \sum_{k=j+1}^{N} \sum_{b \to a\ c} C^{b \to a\ c}_{(1,3)}(\beta^{b,i,k}, \alpha^{c,j+1,k})$$

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

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

Figure 5: The tensor form of the inside-outside algorithm, for calculation of marginal terms $\mu(a, i, j)$.



$r_1 = \text{S} \to \text{NP V}$
$r_2 = \text{NP} \to \text{D N}$
$r_3 = \text{D} \to \text{the}$
$r_4 = \text{N} \to \text{dog}$
$r_5 = \text{V} \to \text{sleeps}$

Figure 6: An s-tree, and its sequence of rules. (For convenience we have numbered the nodes in the tree.)

6.2 An Example

In the remainder of this section we give an example that illustrates how the algorithm in figure 4 is correct, and gives the basic intuition behind the proof in section A.1. While we concentrate on the algorithm in figure 4, the intuition behind the algorithm in figure 5 is very similar.

Consider the skeletal tree in figure 6. We will demonstrate how the algorithm in figure 4, under the assumptions in the theorem, correctly calculates the probability of this tree. In brief, the argument involves the following steps:

1. We first show that the algorithm in figure 4, when run on the tree in figure 6, calculates the probability of the tree as

$$C^{S \to NP\ V}(C^{NP \to D\ N}(c^{\infty}_{D \to the}, c^{\infty}_{N \to dog}), c^{\infty}_{V \to sleeps})c^{1}_{S}$$

   Note that this expression mirrors the structure of the tree, with $c^{\infty}_{a \to x}$ terms for the leaves, $C^{a \to b\ c}$ terms for each rule production $a \to b\ c$ in the tree, and a c^{1}_{S} term for the root.

2. We then show that under the assumptions in the theorem, the following identity holds:

$$\begin{aligned} & C^{S \to NP\ V}(C^{NP \to D\ N}(c^{\infty}_{D \to the}, c^{\infty}_{N \to dog}), c^{\infty}_{V \to sleeps})c^{1}_{S} \\ = \quad & T^{S \to NP\ V}(T^{NP \to D\ N}(q_{D \to the}, q_{N \to dog}), q_{V \to sleeps})\pi^{S} \end{aligned} \tag{3}$$

   This follows because the G^a and $(G^a)^{-1}$ terms for the various non-terminals in the tree cancel. Note that the expression in Eq. 3 again follows the structure of the tree, but with $q_{a \to x}$ terms for the leaves, $T^{a \to b\ c}$ terms for each rule production $a \to b\ c$ in the tree, and a π^S term for the root.

3. Finally, we show that the expression in Eq. 3 implements the conventional dynamic-programming method for calculation of the tree probability, as described in Eqs. 11–13 below.

We now go over these three points in detail. The algorithm in figure 4 calculates the following terms (each f^i is an m-dimensional row vector):

$$\begin{aligned} f^3 &= c^{\infty}_{D \to the} \\ f^4 &= c^{\infty}_{N \to dog} \\ f^5 &= c^{\infty}_{V \to sleeps} \\ f^2 &= C^{NP \to D\ N}(f^3, f^4) \\ f^1 &= C^{S \to NP\ V}(f^2, f^5) \end{aligned}$$

The final quantity returned by the algorithm is

$$f^1 c^1_S = \sum_h f^1_h [c^1_S]_h$$

Combining the definitions above, it can be seen that

$$f^1 c^1_S = C^{S \to NP\ V}(C^{NP \to D\ N}(c^{\infty}_{D \to the}, c^{\infty}_{N \to dog}), c^{\infty}_{V \to sleeps})c^1_S$$

demonstrating that point 1 above holds.

Next, given the assumptions in the theorem, we show point 2, that is, that

$$\begin{aligned} & C^{S\to NP\ V}(C^{NP\to D\ N}(c^{\infty}_{D\to the}, c^{\infty}_{N\to dog}), c^{\infty}_{V\to sleeps})c^{1}_{S} \\ = \ & T^{S\to NP\ V}(T^{NP\to D\ N}(q_{D\to the}, q_{N\to dog}), q_{V\to sleeps})\pi^{S} \end{aligned} \tag{4}$$

This follows because the G^a and $(G^a)^{-1}$ terms in the theorem cancel. More specifically, we have

$$f^3 = c^{\infty}_{D\to the} = q_{D\to the}(G^D)^{-1} \tag{5}$$
$$f^4 = c^{\infty}_{N\to dog} = q_{N\to dog}(G^N)^{-1} \tag{6}$$
$$f^5 = c^{\infty}_{V\to sleeps} = q_{V\to sleeps}(G^V)^{-1} \tag{7}$$
$$f^2 = C^{NP\to D\ N}(f^3, f^4) = T^{NP\to D\ N}(q_{D\to the}, q_{D\to dog})(G^{NP})^{-1} \tag{8}$$
$$f^1 = C^{S\to NP\ V}(f^2, f^5) = T^{S\to NP\ V}(T^{NP\to D\ N}(q_{D\to the}, q_{N\to dog}), q_{V\to sleeps})(G^S)^{-1} \tag{9}$$

Eqs. 5, 6, 7 follow by the assumptions in the theorem. Eq. 8 follows because by the assumptions in the theorem

$$C^{NP\to D\ N}(f^3, f^4) = T^{NP\to D\ N}(f^3 G^D, f^4 G^N)(G^{NP})^{-1}$$

hence

$$\begin{aligned} C^{NP\to D\ N}(f^3, f^4) &= T^{NP\to D\ N}(q_{D\to the}(G^D)^{-1}G^D, q_{N\to dog}(G^N)^{-1}G^N)(G^{NP})^{-1} \\ &= T^{NP\to D\ N}(q_{D\to the}, q_{N\to dog})(G^{NP})^{-1} \end{aligned}$$

Eq. 9 follows in a similar manner.

It follows by the assumption that $c^1_S = G^S\pi^S$ that

$$\begin{aligned} & C^{S\to NP\ V}(C^{NP\to D\ N}(c^{\infty}_{D\to the}, c^{\infty}_{N\to dog}), c^{\infty}_{V\to sleeps})c^{1}_{S} \\ = \ & T^{S\to NP\ V}(T^{NP\to D\ N}(q_{D\to the}, q_{N\to dog}), q_{V\to sleeps})(G^S)^{-1}G^S\pi^S \\ = \ & T^{S\to NP\ V}(T^{NP\to D\ N}(q_{D\to the}, q_{N\to dog}), q_{V\to sleeps})\pi^S \end{aligned} \tag{10}$$

The final step (point 3) is to show that the expression in Eq. 10 correctly calculates the probability of the example tree. First consider the term $T^{NP\to D\ N}(q_{D\to the}, q_{N\to dog})$—this is an m-dimensional row vector, call this b^2. By the definition of the tensor $T^{NP\to D\ N}$, we have

$$\begin{aligned} b^2_h &= \left[T^{NP\to D\ N}(q_{D\to the}, q_{N\to dog})\right]_h \\ &= \sum_{h_2,h_3} t(NP \to D\ N, h_2, h_3|h, NP) \times q(D \to the|h_2, D) \times q(N \to dog|h_3, N) \end{aligned} \tag{11}$$

By a similar calculation, $T^{S\to NP\ V}(T^{NP\to D\ N}(q_{D\to the}, q_{N\to dog}), q_{V\to sleeps})$—call this vector b^1—is

$$b^1_h = \sum_{h_2,h_3} t(S \to NP\ V, h_2, h_3|h, S) \times b^2_{h_2} \times q(V \to sleeps|h_3, V) \tag{12}$$

Finally, the probability of the full tree is calculated as

$$\sum_h b_h^1 \pi_h^S \tag{13}$$

It can be seen that the expression in Eq. 4 implements the calculations in Eqs. 11, 12 and 13, which are precisely the calculations used in the conventional dynamic programming algorithm for calculation of the probability of the tree.

7. Estimating the Tensor Model

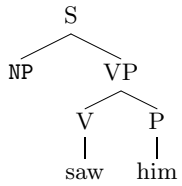
A crucial result is that it is possible to directly estimate parameters $C^{a \to b\, c}$, $c_{a \to x}^{\infty}$ and c_a^1 that satisfy the conditions in theorem 1, from a training sample consisting of s-trees (i.e., trees where hidden variables are unobserved). We first describe random variables underlying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

7.1 Random Variables Underlying the Approach

Each s-tree with N rules $r_1 \dots r_N$ 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` $\to$ `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



The outside tree contains everything in the s-tree $r_1 \dots r_N$, 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 a full tree $r_1 \dots r_N, h_1 \dots h_N$ from the PMF $p(r_1 \dots r_N, h_1 \dots h_N)$. 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` $\to$ `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

$$Y_1 = (U^{a_1})^\top \phi(T_1) \;\; Z = (V^{a_1})^\top \psi(O)$$
$$Y_2 = (U^{a_2})^\top \phi(T_2) \;\; Y_3 = (U^{a_3})^\top \phi(T_3)$$

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

7.2 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:

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

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 8).

Our observable representation then consists of:

$$C^{a \rightarrow b\ c}(y^1, y^2) = D^{a \rightarrow b\ c}(y^1, y^2)(\Sigma^a)^{-1} \quad (14)$$
$$c^\infty_{a \rightarrow x} = d^\infty_{a \rightarrow x}(\Sigma^a)^{-1} \quad (15)$$
$$c^1_a = \mathbf{E}\left[[[A_1 = a]] Y_1 | B = 1\right] \quad (16)$$

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.*

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. 14-16:*

1. *For all rules $a \rightarrow b\ c$, $C^{a \rightarrow b\ c}(y^1, y^2) = \left(T^{a \rightarrow b\ c}(y^1 G^b, y^2 G^c)\right)(G^a)^{-1}$*
2. *For all rules $a \rightarrow x$, $c^\infty_{a \rightarrow x} = q_{a \rightarrow x}(G^a)^{-1}$.*
3. *For all $a \in \mathcal{N}$, $c^1_a = G^a \pi^a$*

Proof: The following identities hold (see section A.2):

$$D^{a \rightarrow b\ c}(y^1, y^2) = \left(T^{a \rightarrow b\ c}(y^1 G^b, y^2 G^c)\right) \text{diag}(\gamma^a)(K^a)^\top \quad (17)$$

$$d^\infty_{a \rightarrow x} = q_{a \rightarrow x} \text{diag}(\gamma^a)(K^a)^\top \quad (18)$$

$$\Sigma^a = G^a \text{diag}(\gamma^a)(K^a)^\top \quad (19)$$

$$c^1_a = G^a \pi^a \quad (20)$$

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

This theorem leads directly to the spectral learning algorithm, which we describe in the next section. We give a sketch of the approach here. Assume that we have a training set consisting of skeletal trees (no latent variables are observed) generated from some underlying L-PCFG. Assume in addition that we have definitions of ϕ, ψ, U^a and V^a such that conditions 1 and 2 are satisfied for the L-PCFG. Then it is straightforward to use the training examples to derive i.i.d. samples from the joint distribution over the random variables $(A_1, R_1, Y_1, Y_2, Y_3, Z, B)$ used in the definitions in Eqs. 14–16. These samples can be used to estimate the quantities in Eqs. 14–16; the estimated quantities $\hat{C}^{a \rightarrow b\ c}$, $\hat{c}^\infty_{a \rightarrow x}$ and $\hat{c}^1_a$ can then be used as inputs to the algorithms in figures 4 and 5. By standard arguments, the estimates $\hat{C}^{a \rightarrow b\ c}$, $\hat{c}^\infty_{a \rightarrow x}$ and $\hat{c}^1_a$ will converge to the values in Eqs. 14–16.

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, assuming that condition 1 holds:

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

$$\Omega^a = \mathbf{E}[\phi(T_1)\left(\psi(O)\right)^\top | A_1 = a] \tag{21}$$

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 $\Omega^a = I^a \text{diag}(\gamma^a)(J^a)^\top$. 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. Similar arguments to those of (Hsu et al., 2009) can be used to show that with a sufficient number of samples, the resulting estimates of U^a and V^a satisfy condition 2 with high probability.

8. Deriving Empirical Estimates

Figure 7 shows an algorithm that derives estimates of the quantities in Eqs 14, 15, and 16. 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 $\tau_1 \ldots \tau_M$ 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 $a \rightarrow b\ c$, 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 $a \rightarrow x$, then $t^{(i,2)} = t^{(i,3)} = \texttt{NULL}$. $o^{(i)}$ 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 $\tau_1 \ldots \tau_M$ 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 $C^{a \rightarrow b\ c}$, $c^\infty_{a \rightarrow x}$, and c^1_a. For example, the quantities

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

can be estimated as

$$\hat{\Sigma}^a = \delta_a \times \sum_{i=1}^{M} [[a_i = a]] y^{(i,1)} (z^{(i)})^\top$$

$$\hat{D}^{a \rightarrow b\ c} = \delta_a \times \sum_{i=1}^{M} [[r^{(i,1)} = a \rightarrow b\ c]] z^{(i)} (y^{(i,2)})^\top (y^{(i,3)})^\top$$

$$\hat{d}^{\infty}_{a\to x} = \delta_a \times \sum_{i=1}^{M} [[r^{(i,1)} = a \to x]](z^{(i)})^{\top}$$

where $\delta_a = 1/\sum_{i=1}^{M}[[a_i = a]]$, and we can then set

$$\hat{C}^{a\to b\ c}(y^1, y^2) = \hat{D}^{a\to b\ c}(y^1, y^2)(\hat{\Sigma}^a)^{-1}$$

$$\hat{c}^{\infty}_{a\to x} = \hat{d}^{\infty}_{a\to x}(\hat{\Sigma}^a)^{-1}$$

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 7 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. 21. 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 5 with inputs $\hat{c}^1_a$, $\hat{c}^{\infty}_{a\to x}, \hat{C}^{a\to b\ c}$ derived from the algorithm in figure 7. 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 7 where $r^{i,1}$ has non-terminal a on its left-hand-side. Under these assumptions, if for all a*

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

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 $\hat{c}^1_a$, $\hat{d}^{\infty}_{a\to x}, \hat{D}^{a\to b\ c}$, $\hat{\Sigma}^a$

Inputs: Training examples $(r^{(i,1)}, t^{(i,1)}, t^{(i,2)}, t^{(i,3)}, o^{(i)}, b^{(i)})$ for $i \in \{1 \dots M\}$, 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 8 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 $\hat{D}^{a \to b\ c} = \delta_a \times \sum_{i=1}^M [[r^{(i,1)} = a \to b\ c]] z^{(i)} (y^{(i,2)})^\top (y^{(i,3)})^\top$
- 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 $a \to b\ c$, $\hat{C}^{a \to b\ c}(y^1, y^2) = \hat{D}^{a \to b\ c}(y^1, y^2)(\hat{\Sigma}^a)^{-1}$
- 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 7: The spectral learning algorithm.

are all close to the underlying values being estimated. The second step is to show that this ensures that $\frac{\hat{p}(r_1 \dots r_N)}{p(r_1 \dots r_N)}$ 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.

Inputs: Identical to algorithm in figure 7.
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 8: Singular value decompositions.

9. Discussion

There are several potential applications of the method. The most obvious is parsing with L-PCFGs.[3] 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.

In terms of efficiency, the first step of the algorithm in figure 7 requires an SVD calculation: modern methods for calculating SVDs are very efficient (e.g., see Dhillon et al., 2011 and Tropp et al., 2009). The remaining steps of the algorithm require manipulation of tensors or vectors, and require $O(Mm^3)$ time.

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.

For simplicity we have considered the case where each non-terminal has the same number, m, of possible hidden values. It is simple to generalize the algorithms to the case where the number of hidden values varies depending on the non-terminal; this may be important in applications.

Appendix A. Proofs

This section gives proofs of theorems 1 and 2.

A.1 Proof of Theorem 1

The key idea behind the proof of theorem 1 is to show that the algorithms in figures 4 and 5 compute the same quantities as the conventional version of the inside outside algorithms, as shown in figures 2 and 3.

First, the following lemma leads directly to the correctness of the algorithm in figure 4:

3. Parameters can be estimated using the algorithm in figure 7; for a test sentence $x_1 \ldots x_N$ we can first use the algorithm in figure 5 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 4 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. For all $i \in [N]$, define the row vector $b^i \in \mathbb{R}^{(1\times m)}$ to be the vector computed by the conventional inside-outside algorithm, as shown in figure 2, on the s-tree $r_1 \dots r_N$. Define $f^i \in \mathbb{R}^{(1\times m)}$ to be the vector computed by the tensor-based inside-outside algorithm, as shown in figure 4, on the s-tree $r_1 \dots r_N$.*

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

$$f^1 c^1_{a_1} = b^1 (G^{(a_1)})^{-1} G^{a_1} \pi_{a_1} = b^1 \pi_{a_1} = \sum_h b^1_h \pi(a, h)$$

Hence the output from the algorithms in figures 2 and 4 is the same, and it follows that the tensor-based algorithm in figure 4 is correct.

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,

$$\begin{aligned} f^i &= C^{r_i}(f^\beta, f^\gamma) \\ &= \left(T^{r_i}(f^\beta G^{a_\beta}, f^\gamma G^{a_\gamma})\right)(G^{a_i})^{-1} \end{aligned}$$

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

$$f^i = \left(T^{r_i}(b^\beta, b^\gamma)\right)(G^{a_i})^{-1} \tag{22}$$

By the definition of the tensor T^{r_i},

$$\left[T^{r_i}(b^\beta, b^\gamma)\right]_h = \sum_{h_2 \in [m], h_3 \in [m]} t(r_i, h_2, h_3 | a_i, h) b^\beta_{h_2} b^\gamma_{h_3}$$

But by definition (see the algorithm in figure 2),

$$b^i_h = \sum_{h_2 \in [m], h_3 \in [m]} t(r_i, h_2, h_3 | a_i, h) b^\beta_{h_2} b^\gamma_{h_3}$$

hence $b^i = T^{r_i}(b^\beta, b^\gamma)$ and the inductive case follows immediately from Eq. 22. □

Next, we give a similar lemma, which implies the correctness of the algorithm in figure 5:

Lemma 3 *Assume that conditions 1-3 of theorem 1 are satisfied, and that the input to the algorithm in figure 5 is a sentence $x_1 \dots x_N$. For any $a \in \mathcal{N}$, for any $1 \leq i \leq j \leq N$, define $\bar{\alpha}^{a,i,j} \in \mathbb{R}^{(1\times m)}$, $\bar{\beta}^{a,i,j} \in \mathbb{R}^{(m\times 1)}$ and $\bar{\mu}(a, i, j) \in \mathbb{R}$ to be the quantities computed by the conventional inside-outside algorithm in figure 3 on the input $x_1 \dots x_N$. Define*

$\alpha^{a,i,j} \in \mathbb{R}^{(1\times m)}$, $\beta^{a,i,j} \in \mathbb{R}^{(m\times 1)}$ *and* $\mu(a,i,j) \in \mathbb{R}$ *to be the quantities computed by the algorithm in figure 3.*

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),

$$\mu(a,i,j) = \alpha^{a,i,j}\beta^{a,i,j} = \bar{\alpha}^{a,i,j}(G^a)^{-1}G^a\bar{\beta}^{a,i,j} = \bar{\alpha}^{a,i,j}\bar{\beta}^{a,i,j} = \bar{\mu}(a,i,j)$$

Hence the outputs from the algorithms in figures 3 and 5 are the same, and it follows that the tensor-based algorithm in figure 5 is correct.

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.

Proof: The proof is by induction, and is similar to the proof of lemma 2.

First, we prove that the inside terms satisfy the relation $\alpha^{a,i,j} = \bar{\alpha}^{a,i,j}(G^a)^{-1}$.

The base case of the induction is as follows. By definiton, for any $a \in \mathcal{P}, i \in [N], h \in [m]$, we have $\bar{\alpha}_h^{a,i,i} = q(a \to x_i|h,a)$. We also have for any $a \in \mathcal{P}, i \in [N]$, $\alpha^{a,i,i} = c^{\infty}_{a\to x_i} = q_{a\to x_i}(G^a)^{-1}$. It follows directly that $\alpha^{a,i,i} = \bar{\alpha}^{a,i,i}(G^a)^{-1}$ for any $a \in \mathcal{P}, i \in [N]$.

The inductive case is as follows. By definition, we have $\forall a \in \mathcal{I}, 1 \le i < j \le N, h \in [m]$

$$\bar{\alpha}_h^{a,i,j} = \sum_{k=i}^{j-1}\sum_{b,c}\sum_{h_2\in[m]}\sum_{h_3\in[m]} t(a \to b\ c, h_2, h_3|h,a) \times \bar{\alpha}_{h_2}^{b,i,k} \times \bar{\alpha}_{h_3}^{c,k+1,j}$$

We also have $\forall a \in \mathcal{I}, 1 \le i < j \le N$,

$$\begin{aligned}
\alpha^{a,i,j} &= \sum_{k=i}^{j-1}\sum_{b,c} C^{a\to b\ c}(\alpha^{b,i,k}, \alpha^{c,k+1,j}) && (23)\\
&= \sum_{k=i}^{j-1}\sum_{b,c}\left(T^{a\to b\ c}(\alpha^{b,i,k}G^b, \alpha^{c,k+1,j}G^c)\right)(G^a)^{-1} && (24)\\
&= \sum_{k=i}^{j-1}\sum_{b,c}\left(T^{a\to b\ c}(\bar{\alpha}^{b,i,k}, \bar{\alpha}^{c,k+1,j}\right)(G^a)^{-1} && (25)\\
&= \bar{\alpha}^{a,i,j}(G^a)^{-1} && (26)
\end{aligned}$$

Eq. 23 follows by the definitions in algorithm 5. Eq. 24 follows by the assumption in the theorem that

$$C^{a\to b\ c}(y^1, y^2) = \left(T^{a\to b\ c}(y^1G^b, y^2G^c)\right)(G^a)^{-1}$$

Eq. 25 follows because by the inductive hypothesis, $\alpha^{b,i,k} = \bar{\alpha}^{b,i,k}(G^b)^{-1}$ and $\alpha^{c,k+1,j} = \bar{\alpha}^{c,k+1,j}(G^c)^{-1}$. Eq. 26 follows because

$$\left[T^{a\to b\ c}(\bar{\alpha}^{b,i,k}, \bar{\alpha}^{c,k+1,j})\right]_h = \sum_{h_2,h_3} t(a \to b\ c, h_2, h_3|h,a)\bar{\alpha}_{h_2}^{b,i,k}\bar{\alpha}_{h_3}^{c,k+1,j}$$

hence

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

We now turn the outside terms, proving that $\beta^{a,i,j} = G^a \bar{\beta}^{a,i,j}$. The proof is again by induction.

The base case is as follows. By the definitions in the algorithms, for all $a \in \mathcal{I}$, $\beta^{a,1,n} = c_a^1 = G^a \pi^a$, and for all $a \in \mathcal{I}, h \in [m]$, $\bar{\beta}_h^{a,1,n} = \pi(a,h)$. It follows directly that for all $a \in \mathcal{I}$, $\beta^{a,1,n} = G^a \bar{\beta}^{a,1,n}$.

The inductive case is as follows. By the definitions in the algorithms, we have $\forall a \in \mathcal{N}, 1 \leq i \leq j \leq N, h \in [m]$

$$\bar{\beta}_h^{a,i,j} = \gamma_h^{1,a,i,j} + \gamma_h^{2,a,i,j}$$

where

$$\gamma_h^{1,a,i,j} = \sum_{k=1}^{i-1} \sum_{b \to c\ a} \sum_{h_2 \in [m]} \sum_{h_3 \in [m]} t(b \to c\ a, h_3, h | h_2, b) \times \bar{\beta}_{h_2}^{b,k,j} \times \bar{\alpha}_{h_3}^{c,k,i-1}$$

$$\gamma_h^{2,a,i,j} = \sum_{k=j+1}^{N} \sum_{b \to a\ c} \sum_{h_2 \in [m]} \sum_{h_3 \in [m]} t(b \to a\ c, h, h_3 | h_2, b) \times \bar{\beta}_{h_2}^{b,i,k} \times \bar{\alpha}_{h_3}^{c,j+1,k}$$

and $\forall a \in \mathcal{N}, 1 \leq i \leq j \leq N$,

$$\beta^{a,i,j} = \sum_{k=1}^{i-1} \sum_{b \to c\ a} C_{(1,2)}^{b \to c\ a}(\beta^{b,k,j}, \alpha^{c,k,i-1}) + \sum_{k=j+1}^{N} \sum_{b \to a\ c} C_{(1,3)}^{b \to a\ c}(\beta^{b,i,k}, \alpha^{c,j+1,k})$$

Critical identities are

$$\sum_{k=1}^{i-1} \sum_{b \to c\ a} C_{(1,2)}^{b \to c\ a}(\beta^{b,k,j}, \alpha^{c,k,i-1}) = G^a \gamma^{1,a,i,j} \quad (27)$$

$$\sum_{k=j+1}^{N} \sum_{b \to a\ c} C_{(1,3)}^{b \to a\ c}(\beta^{b,i,k}, \alpha^{c,j+1,k}) = G^a \gamma^{2,a,i,j} \quad (28)$$

from which $\beta^{a,i,j} = G^a \bar{\beta}^{a,i,j}$ follows immediately.

The identities in Eq. 29 and 30 are proved through straightforward algebraic manipulation, based on the following properties:

- By the inductive hypothesis, $\beta^{b,k,j} = G^b \bar{\beta}^{b,k,j}$ and $\beta^{b,i,k} = G^b \bar{\beta}^{b,i,k}$.
- By correctness of the inside terms, as shown earlier in this proof, $\alpha^{c,k,i-1} = \bar{\alpha}^{c,k,i-1}(G^c)^{-1}$, $\alpha^{c,j+1,k} = \bar{\alpha}^{c,j+1,k}(G^c)^{-1}$.
- By the assumptions in the theorem,

$$C^{a \to b\ c}(y^1, y^2) = \left(T^{a \to b\ c}(y^1 G^b, y^2 G^c)\right)(G^a)^{-1}$$

It follows (see Lemma 4) that

$$\begin{aligned} C_{(1,2)}^{b\to c\ a}(\beta^{b,k,j},\alpha^{c,k,i-1}) &= G^a\left(T_{(1,2)}^{b\to c\ a}((G^b)^{-1}\beta^{b,k,j},\alpha^{c,k,i-1}G^c)\right) \\ &= G^a\left(T_{(1,2)}^{b\to c\ a}(\bar{\beta}^{b,k,j},\bar{\alpha}^{c,k,i-1})\right) \end{aligned}$$

and

$$C_{(1,3)}^{b\to a\ c}(\beta^{b,i,k},\alpha^{c,j+1,k}) = G^a\left(T_{(1,3)}^{b\to a\ c}(\bar{\beta}^{b,i,k},\bar{\alpha}^{c,j+1,k})\right)$$

□

Finally, we give the following Lemma, as used above:

Lemma 4 *Assume we have tensors $C \in \mathbb{R}^{m\times m\times m}$ and $T \in \mathbb{R}^{m\times m\times m}$ such that for any y^2, y^3,*

$$C(y^2,y^3) = \left(T(y^2A, y^3B)\right) D$$

where A, B, D are matrices in $\mathbb{R}^{m\times m}$. Then for any y^1, y^2,

$$C_{(1,2)}(y^1,y^2) = B\left(T_{(1,2)}(Dy^1, y^2A)\right) \tag{29}$$

and for any y^1, y^3,

$$C_{(1,3)}(y^1,y^3) = A\left(T_{(1,3)}(Dy^1, y^3B)\right) \tag{30}$$

Proof: Consider first Eq. 29. We will prove the following statement:

$$\forall y^1, y^2, y^3, \quad y^3 C_{(1,2)}(y^1,y^2) = y^3 B\left(T_{(1,2)}(Dy^1, y^2A)\right)$$

This statement is equivalent to Eq. 29.

First, for all y^1, y^2, y^3, by the assumption that $C(y^2,y^3) = \left(T(y^2A, y^3B)\right) D$,

$$C(y^2,y^3)y^1 = T(y^2A, y^3B)Dy^1$$

hence

$$\sum_{i,j,k} C_{i,j,k} y_i^1 y_j^2 y_k^3 = \sum_{i,j,k} T_{i,j,k} z_i^1 z_j^2 z_k^3 \tag{31}$$

where $z^1 = Dy^1$, $z^2 = y^2A$, $z^3 = y^3B$.

In addition, it is easily verified that

$$y^3 C_{(1,2)}(y^1,y^2) = \sum_{i,j,k} C_{i,j,k} y_i^1 y_j^2 y_k^3 \tag{32}$$

$$y^3 B\left(T_{(1,2)}(Dy^1, y^2A)\right) = \sum_{i,j,k} T_{i,j,k} z_i^1 z_j^2 z_k^3 \tag{33}$$

where again $z^1 = Dy^1$, $z^2 = y^2A$, $z^3 = y^3B$. Combining Eqs. 31, 32, and 33 gives

$$y^3 C_{(1,2)}(y^1,y^2) = y^3 B\left(T_{(1,2)}(Dy^1, y^2A)\right)$$

thus proving the identity in Eq. 29.

The proof of the identity in Eq. 30 is similar, and is omitted for brevity. □

A.2 Proof of the Identity in Eq. 17

We now prove the identity in Eq. 17, repeated here:

$$D^{a \to b\ c}(y^1, y^2) = \left(T^{a \to b\ c}(y^1 G^b, y^2 G^c)\right) \text{diag}(\gamma^a)(K^a)^\top$$

Recall that

$$D^{a \to b\ c} = \mathbf{E}\left[[[R_1 = a \to b\ c]] Z Y_2^\top Y_3^\top | A_1 = a\right]$$

or equivalently

$$D^{a \to b\ c}_{i,j,k} = \mathbf{E}\left[[[R_1 = a \to b\ c]] Z_i Y_{2,j} Y_{3,k} | A_1 = a\right]$$

Using the chain rule, and marginalizing over hidden variables, we have

$$\begin{aligned} D^{a \to b\ c}_{i,j,k} &= \mathbf{E}\left[[[R_1 = a \to b\ c]] Z_i Y_{2,j} Y_{3,k} | A_1 = a\right] \\ &= \sum_{h_1, h_2, h_3 \in [m]} p(a \to b\ c, h_1, h_2, h_3 | a) \mathbf{E}\left[Z_i Y_{2,j} Y_{3,k} | R_1 = a \to b\ c, h_1, h_2, h_3\right] \end{aligned}$$

By definition, we have

$$p(a \to b\ c, h_1, h_2, h_3 | a) = \gamma^a_{h_1} \times t(a \to b\ c, h_2, h_3 | h_1, a)$$

In addition, under the independence assumptions in the L-PCFG, and using the definitions of K^a and G^a, we have

$$\begin{aligned} & \mathbf{E}\left[Z_i Y_{2,j} Y_{3,k} | R_1 = a \to b\ c, h_1, h_2, h_3\right] \\ = \ & \mathbf{E}\left[Z_i | A_1 = a, H_1 = h_1\right] \times \mathbf{E}\left[Y_{2,j} | A_2 = b, H_2 = h_2\right] \times \mathbf{E}\left[Y_{3,k} | A_3 = c, H_3 = h_3\right] \\ = \ & K^a_{i,h_1} \times G^b_{j,h_2} \times G^c_{k,h_3} \end{aligned}$$

Putting this all together gives

$$\begin{aligned} D^{a \to b\ c}_{i,j,k} &= \sum_{h_1, h_2, h_3 \in [m]} \gamma^a_{h_1} \times t(a \to b\ c, h_2, h_3 | h_1, a) \times K^a_{i,h_1} \times G^b_{j,h_2} \times G^c_{k,h_3} \\ &= \sum_{h_1 \in [m]} \gamma^a_{h_1} \times K^a_{i,h_1} \times \sum_{h_2, h_3 \in [m]} t(a \to b\ c, h_2, h_3 | h_1, a) \times G^b_{j,h_2} \times G^c_{k,h_3} \end{aligned}$$

By the definition of tensors,

$$\begin{aligned} & [D^{a \to b\ c}(y^1, y^2)]_i \\ = \ & \sum_{j,k} D^{a \to b\ c}_{i,j,k} y^1_j y^2_k \\ = \ & \sum_{h_1 \in [m]} \gamma^a_{h_1} \times K^a_{i,h_1} \times \sum_{h_2, h_3 \in [m]} t(a \to b\ c, h_2, h_3 | h_1, a) \times \left(\sum_j y^1_j G^b_{j,h_2}\right) \times \left(\sum_k y^2_k G^c_{k,h_3}\right) \\ = \ & \sum_{h_1 \in [m]} \gamma^a_{h_1} \times K^a_{i,h_1} \times \left[T^{a \to b\ c}(y^1 G^b, y^2 G^c)\right]_{h_1} \end{aligned} \tag{34}$$

The last line follows because by the definition of tensors,

$$\left[T^{a\to b\ c}(y^1 G^b, y^2 G^c)\right]_{h_1} = \sum_{h_2,h_3} T^{a\to b\ c}_{h_1,h_2,h_3} \left[y^1 G^b\right]_{h_2} \left[y^2 G^c\right]_{h_3}$$

and we have

$$\begin{aligned} T^{a\to b\ c}_{h_1,h_2,h_3} &= t(a \to b\ c, h_2, h_3 | h_1, a) \\ \left[y^1 G^b\right]_{h_2} &= \sum_j y^1_j G^b_{j,h_2} \\ \left[y^2 G^c\right]_{h_3} &= \sum_k y^2_k G^c_{k,h_3} \end{aligned}$$

Finally, the required identity

$$D^{a\to b\ c}(y^1, y^2) = \left(T^{a\to b\ c}(y^1 G^b, y^2 G^c)\right) \text{diag}(\gamma^a)(K^a)^\top$$

follows immediately from Eq. 34. □

A.3 Proof of the Identity in Eq. 18

We now prove the identity in Eq. 18, repeated below:

$$d^\infty_{a\to x} = q_{a\to x} \text{diag}(\gamma^a)(K^a)^\top$$

Recall that by definition

$$d^\infty_{a\to x} = \mathbf{E}\left[[[R_1 = a \to x]] Z^\top | A_1 = a\right]$$

or equivalently

$$[d^\infty_{a\to x}]_i = \mathbf{E}\left[[[R_1 = a \to x]] Z_i | A_1 = a\right]$$

Marginalizing over hidden variables, we have

$$\begin{aligned} [d^\infty_{a\to x}]_i &= \mathbf{E}\left[[[R_1 = a \to x]] Z_i | A_1 = a\right] \\ &= \sum_h p(a \to x, h | a) \mathbf{E}[Z_i | H_1 = h, R_1 = a \to x] \end{aligned}$$

By definition, we have

$$p(a \to x, h | a) = \gamma^a_h q(a \to x | h, a) = \gamma^a_h \left[q_{a\to x}\right]_h$$

In addition, by the independence assumptions in the L-PCFG, and the definition of K^a,

$$\mathbf{E}[Z_i | H_1 = h, R_1 = a \to x] = \mathbf{E}[Z_i | H_1 = h, A_1 = a] = K^a_{i,h}$$

Putting this all together gives

$$[d^\infty_{a\to x}]_i = \sum_h \gamma^a_h \left[q_{a\to x}\right]_h K^a_{i,h}$$

from which the required identity

$$d^\infty_{a\to x} = q_{a\to x} \text{diag}(\gamma^a)(K^a)^\top$$

follows immediately. □

A.4 Proof of the Identity in Eq. 19

We now prove the identity in Eq. 19, repeated below:

$$\Sigma^a = G^a \text{diag}(\gamma^a)(K^a)^\top$$

Recall that by definition

$$\Sigma^a = \mathbf{E}[Y_1 Z^\top | A_1 = a]$$

or equivalently

$$[\Sigma^a]_{i,j} = \mathbf{E}[Y_{1,i} Z_j | A_1 = a]$$

Marginalizing over hidden variables, we have

$$\begin{aligned} [\Sigma^a]_{i,j} &= \mathbf{E}[Y_{1,i} Z_j | A_1 = a] \\ &= \sum_h p(h|a) \mathbf{E}[Y_{1,i} Z_j | H_1 = h, A_1 = a] \end{aligned}$$

By definition, we have

$$\gamma^a_h = p(h|a)$$

In addition, under the independence assumptions in the L-PCFG, and using the definitions of K^a and G^a, we have

$$\begin{aligned} \mathbf{E}[Y_{1,i} Z_j | H_1 = h, A_1 = a] &= \mathbf{E}[Y_{1,i} | H_1 = h, A_1 = a] \times \mathbf{E}[Z_j | H_1 = h, A_1 = a] \\ &= G^a_{i,h} K^a_{j,h} \end{aligned}$$

Putting all this together gives

$$[\Sigma^a]_{i,j} = \sum_h \gamma^a_h G^a_{i,h} K^a_{j,h}$$

from which the required identity

$$\Sigma^a = G^a \text{diag}(\gamma^a)(K^a)^\top$$

follows immediately. □

A.5 Proof of the Identity in Eq. 20

We now prove the identity in Eq. 19, repeated below:

$$c^1_a = G^a \pi^a$$

Recall that by definition

$$c^1_a = \mathbf{E}\left[[[A_1 = a]] Y_1 | B = 1\right]$$

or equivalently

$$[c^1_a]_i = \mathbf{E}\left[[[A_1 = a]] Y_{1,i} | B = 1\right]$$

Marginalizing over hidden variables, we have

$$\begin{aligned}[c_a^1]_i &= \mathbf{E}\left[[[A_1 = a]]Y_{1,i}|B = 1\right] \\ &= \sum_h P(A_1 = a, H_1 = h|B = 1)\mathbf{E}\left[Y_{1,i}|A_1 = a, H_1 = h, B = 1\right]\end{aligned}$$

By definition we have

$$P(A_1 = a, H_1 = h|B = 1) = \pi(a, h)$$

By the independence assumptions in the PCFG, and the definition of G^a, we have

$$\begin{aligned}\mathbf{E}\left[Y_{1,i}|A_1 = a, H_1 = h, B = 1\right] &= \mathbf{E}\left[Y_{1,i}|A_1 = a, H_1 = h\right] \\ &= G^a_{i,h}\end{aligned}$$

Putting this together gives

$$[c_a^1]_i = \sum_h \pi(a, h)G^a_{i,h}$$

from which the required identity

$$c_a^1 = G^a\pi^a$$

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

Baker, J. (1979). Trainable Grammars for Speech Recognition. In *Proc. ASA*.

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

Charniak, E. (1997). Statistical Parsing with a Context-Free Grammar and Word Statistics. In *Proc. AAAI-IAAI*, pp. 598–603.

Cohen, S. B., Stratos, K., Collins, M., Foster, D. P., & Ungar, L. (2012). Spectral learning of latent-variable pcfgs. In *Proceedings of ACL*.

Collins, M. (1997). Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, pp. 16–23, Madrid, Spain. Association for Computational Linguistics.

Dhillon, P., Foster, D., & Ungar, L. (2011). Multi-view learning of word embeddings via CCA. In *Proceedings of NIPS 24 (Advances in Neural Information Processing Systems)*.

Foster, D. P., Rodu, J., & Ungar, L. H. (2012). Spectral dimensionality reduction for hmms. arXiv:1203.6130v1.

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

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

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

Johnson, M. (1998). PCFG Models of Linguistic Tree Representations. *Computational Linguistics*, *24*(4), 613–632.

Klein, D., & Manning, C. (2003). Accurate Unlexicalized Parsing. In *Proc. ACL*, pp. 423–430.

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

Marcus, M., Santorini, B., & Marcinkiewicz, M. (1993). Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, *19*(2), 313–330.

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

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

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

Petrov, S., Barrett, L., Thibaux, R., & Klein, D. (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*, pp. 433–440, Sydney, Australia. Association for Computational Linguistics.

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

Tropp, A., Halko, N., & Martinsson, P. G. (2009). Finding structure with randomness: Stochastic algorithms for constructing approximate matrix decompositions.. In *Technical Report No. 2009-05*.