Spectral Dependency Parsing with Latent Variables

Paramveer S. Dhillon[1], **Jordan Rodu**[2], **Michael Collins**[3], **Dean P. Foster**[2] **and Lyle H. Ungar**[1]
[1]Computer & Information Science/ [2]Statistics, University of Pennsylvania, Philadelphia, PA, U.S.A
[3] Computer Science, Columbia University, New York, NY, U.S.A
{dhillon|ungar@cis.upenn.edu}, {jrodu|foster@wharton.upenn.edu}
mcollins@cs.columbia.edu

Abstract

Recently there has been substantial interest in using spectral methods to learn generative sequence models like HMMs. Spectral methods are attractive as they provide globally consistent estimates of the model parameters and are very fast and scalable, unlike EM methods, which can get stuck in local minima. In this paper, we present a novel extension of this class of spectral methods to learn dependency tree structures. We propose a simple yet powerful latent variable generative model for dependency parsing, and a spectral learning method to efficiently estimate it. As a pilot experimental evaluation, we use the spectral tree probabilities estimated by our model to re-rank the outputs of a near state-of-the-art parser. Our approach gives us a moderate reduction in error of upto 4.6% over the baseline re-ranker.

1 Introduction

Markov models have been for two decades a workhorse of statistical pattern recognition with applications ranging from speech to vision to language. Adding latent variables to these models gives us additional modeling power and have shown success in applications like POS tagging (Merialdo, 1994), speech recognition (Rabiner, 1989) and object recognition (Quattoni et al., 2004). However, this comes at the cost that the resulting parameter estimation problem becomes non-convex and techniques like EM (Dempster et al., 1977) which are used to estimate the parameters can only lead to locally optimal solutions.

Recent work by Hsu et al. (2008) has shown that globally consistent estimates of the parameters of the HMMs can be found by using spectral methods, particularly by singular value decomposition (SVD) of appropriately defined linear systems. They avoid the NP Hard problem of the global optimization problem of the HMM parameters (Terwijn, 2002), by putting restrictions on the smallest singular value of the HMM parameters. The main intuition behind the model is that, although the observed data (i.e. words) seem to live in a very high dimensional space, but in reality they live in a very low dimensional space (size $k \sim 30 - 50$) and an appropriate eigen decomposition of the observed data will reveal the underlying low dimensional dynamics and thereby revealing the parameters of the model. Besides ducking the NP hard problem, the spectral methods are very fast and scalable to train compared to EM methods.

In this paper we generalize the approach of Hsu et al. (2008) to learn dependency tree structures with latent variables.[1] Petrov et al. (2006); Musillo and Merlo (2008) have shown that learning PCFGs and dependency grammars respectively with latent variables can produce parsers with very good generalization performance. However, both these approaches rely on EM for parameter estimation and can benefit from using spectral methods.

We propose a simple yet powerful latent variable generative model for use with dependency pars-

[1] Actually, instead of using the model by Hsu et al. (2008) we work with a related model proposed by Foster et al. (2012) which addresses some of the shortcomings of the earlier model which we detail below.

ing which has one hidden node for each word in the sentence, like the one shown in Figure 1 and work out the details for the parameter estimation of the corresponding spectral learning model. At a very high level, the parameter estimation of our model involves collecting unigram, bigram and trigram counts sensitive to the underlying dependency structure of the given sentence.

Recently, Luque et al. (2012) have also proposed a spectral method for dependency parsing, however they deal with *horizontal markovization* and use hidden states to model sequential dependencies within a word's sequence of children. In contrast with that, in this paper, we propose a spectral learning algorithm where latent states are not restricted to HMM-like distributions of modifier sequences for a particular head, but instead allow information to be propagated through the entire tree.

More recently, Cohen et al. (2012) have proposed a spectral method for learning PCFGs.

Its worth noting that recent work by Parikh et al. (2011) also extends Hsu et al. (2008) to latent variable dependency trees like us but under the restrictive conditions that model parameters are trained for a specified, albeit arbitrary, tree topology.[2] In other words, all training sentences and test sentences must have identical tree topologies. By doing this they allow for node-specific model parameters, but must retrain the model entirely when a different tree topology is encountered. Our model on the other hand allows the flexibility and efficiency of processing sentences with a variety of tree topologies from a single training run.

Most of the current state-of-the-art dependency parsers are discriminative parsers (Koo et al., 2008; McDonald, 2006) due to the flexibility of representations which can be used as features leading to better accuracies and the ease of reproducibility of results. However, unlike discriminative models, generative models can exploit unlabeled data. Also, as is common in statistical parsing, re-ranking the outputs of a parser leads to significant reductions in error (Collins and Koo, 2005).

Since our spectral learning algorithm uses a generative model of words given a tree structure, it can score a tree structure i.e. its probability of generation. Thus, it can be used to re-rank the n-best outputs of a given parser.

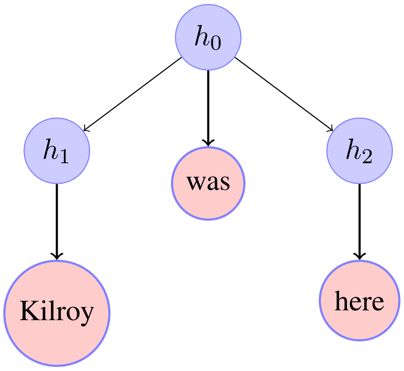

Figure 1: Sample dependency parsing tree for “Kilroy was here”

The remainder of the paper is organized as follows. In the next section we introduce the notation and give a brief overview of the spectral algorithm for learning HMMs (Hsu et al., 2008; Foster et al., 2012). In Section 3 we describe our proposed model for dependency parsing in detail and work out the theory behind it. Section 4 provides experimental evaluation of our model on Penn Treebank data. We conclude with a brief summary and future avenues for research.

2 Spectral algorithm for learning HMMs

In this section we describe the spectral algorithm for learning HMMs.[3]

2.1 Notation

The HMM that we consider in this section is a sequence of hidden states $h \in \{1, \ldots, k\}$ that follow the Markov property:

$$p(h_t|h_1, \ldots, h_{t-1}) = p(h_t|h_{t-1})$$

and a sequence of observations $x \in \{1, \ldots, n\}$ such that

$$p(x_t|x_1, \ldots, x_{t-1}, h_1, \ldots, h_t) = p(x_t|h_t)$$

[2] This can be useful in modeling phylogeny trees for instance, but precludes most NLP applications, since there is a need to model the full set of different tree topologies possible in parsing.

[3] As mentioned earlier, we use the model by Foster et al. (2012) which is conceptually similar to the one by Hsu et al. (2008), but does further dimensionality reduction and thus has lower sample complexity. Also, critically, the fully reduced dimension model that we use generalizes much more cleanly to trees.

The parameters of this HMM are:

- A vector π of length k where $\pi_i = p(h_1 = i)$: The probability of the start state in the sequence being i.

- A matrix T of size $k \times k$ where $T_{i,j} = p(h_{t+1} = i | h_t = j)$: The probability of transitioning to state i, given that the previous state was j.

- A matrix O of size $n \times k$ where $O_{i,j} = p(x = i | h = j)$: The probability of state h emitting observation x.

Define δ_j to be the vector of length n with a 1 in the j^{th} entry and 0 everywhere else, and $\text{diag}(v)$ to be the matrix with the entries of v on the diagonal and 0 everywhere else.

The joint distribution of a sequence of observations $x_1, \dots, x_m$ and a sequence of hidden states $h_1, \dots, h_m$ is:

$$p(x_1, \dots, x_m, h_1, \dots, h_m) = \pi_{h_1} \prod_{j=2}^{m-1} T_{h_j, h_{j-1}} \prod_{j=1}^{m} O_{x_j, h_j}$$

Now, we can write the marginal probability of a sequence of observations as

$$p(x_1, \dots x_m) = \sum_{h_1, \dots, h_m} p(x_1, \dots, x_m, h_1, \dots, h_m)$$

which can be expressed in matrix form[4] as:

$$p(x_1, \dots, x_m) = \mathbf{1}^\top A_{x_m} A_{x_{m-1}} \cdots A_{m_1} \pi$$

where $A_{x_m} \equiv T\text{diag}(O^\top \delta_{x_m})$, and $\mathbf{1}$ is a k-dimensional vector with every entry equal to 1.

A is called an "observation operator", and is effectively a third order tensor, and A_{x_m} which is a matrix, gives the distribution vector over states at time $m+1$ as a function of the state distribution vector at the current time m and the current observation δ_{x_m}. Since A_{x_m} depends on the hidden state, it is not observable, and hence cannot be directly estimated. However Hsu et al. (2008); Foster et al. (2012) showed that under certain conditions there exists a fully observable representation of the observable operator model.

2.2 Fully Observable Representation

Before presenting the model, we need to address a few more points. First, let U be a "representation matrix" (eigenfeature dictionary) which maps each observation to a reduced dimension space ($n \to k$) that satisfies the conditions:

- $U^\top O$ is invertible

- $|U_{ij}| < 1$.

Hsu et al. (2008); Foster et al. (2012) discuss U in more detail, but U can, for example, be obtained by the SVD of the bigram probability matrix (where $P_{ij} = p(x_{t+1} = i | x_t = j)$) or by doing CCA on neighboring n-grams (Dhillon et al., 2011).

Letting $y_i = U^\top \delta_{x_i}$, we have

$$p(x_1, \dots, x_m) = c_\infty^\top C(y_m) C(y_{m-1}) \dots C(y_1) c_1 \qquad (1)$$

where

$$\begin{aligned} c_1 &= \mu \\ c_\infty &= \mu^\top \Sigma^{-1} \\ C(y) &= K(y) \Sigma^{-1} \end{aligned}$$

and μ, Σ and K, described in more detail below, are quantities estimated by frequencies of unigrams, bigrams, and trigrams in the observed (training) data.

Under the assumption that data is generated by an HMM, the distribution $\hat{p}$ obtained by substituting the estimated values $\hat{c}_1$, $\hat{c}_\infty$, and $\widehat{C}(y)$ into equation (1) converges to p sufficiently fast as the amount of training data increases, giving us consistent parameter estimates. For details of the convergence proof, please see Hsu et al. (2008); Foster et al. (2012).

3 Spectral algorithm for Learning Dependency Trees

In this section, we first describe a simple latent variable generative model for dependency parsing. We then define some extra notation and finally present

[4] This is essentially the matrix form of the standard dynamic program (forward algorithm) used to estimate HMMs.

the details of the corresponding spectral learning algorithm for dependency parsing, and prove that our learning algorithm provides a consistent estimation of the marginal probabilities.

It is worth mentioning that an alternate way of approaching the spectral estimation of latent states for dependency parsing is by converting the dependency trees into linear sequences from root-to-leaf and doing a spectral estimation of latent states using Hsu et al. (2008). However, this approach would not give us the correct probability distribution over trees as the probability calculations for different paths through the trees are not independent. Thus, although one could calculate the probability of a path from the root to a leaf, one cannot generalize from this probability to say anything about the neighboring nodes or words. Put another way, when a parent has more than the one descendant, one has to be careful to take into account that the hidden variables at each child node are all conditioned on the hidden variable of the parent.

3.1 A latent variable generative model for Dependency Parsing

In the standard setting, we are given training examples where each training example consists of a sequence of words $x_1, \ldots, x_m$ together with a dependency structure over those words, and we want to estimate the probability of the observed structure. This marginal probability estimates can then be used to build an actual generative dependency parser or, since the marginal probability is conditioned on the tree structure, it can be used re-rank the outputs of a parser.

As in the conventional HMM described in the previous section, we can define a simple latent variable first order dependency parsing model by introducing a hidden variable h_i for each word x_i. The joint probability of a sequence of observed nodes $x_1, \ldots, x_m$ together with hidden nodes $h_1, \ldots, h_m$ can be written as

$$p(x_1, \ldots, x_m, h_1, \ldots, h_m) = \pi_{h_1} \prod_{j=2}^{m} t_{d(j)}(h_j|h_{pa(j)}) \prod_{j=1}^{m} o(x_j|h_j) \quad (2)$$

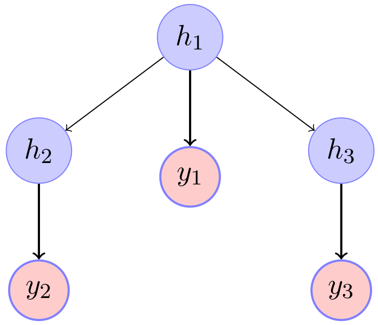

Figure 2: Dependency parsing tree with observed variables y_1, y_2, and y_3.

where $pa(j)$ is the parent of node j and $d(j) \in \{L, R\}$ indicates whether h_j is a left or a right node of $h_{pa(j)}$. For simplicity, the number of hidden and observed nodes in our tree are the same, however they are not required to be so.

As is the case with the conventional HMM, the parameters used to calculate this joint probability are unobservable, but it turns out that under suitable conditions a fully observable model is also possible for the dependency tree case with the parameterization as described below.

3.2 Parameters

We will define both the theoretical representations of our observable parameters, and the sampling versions of these parameters. Note that in all the cases, the estimated versions are unbiased estimates of the theoretical quantities.

Define T_d and T_d^u where $d \in \{L, R\}$ to be the hidden state transition matrices from parent to left or right child, and from left or right child to parent (hence the u for 'up'), respectively. In other words (referring to Figure 2)

$$\begin{aligned} T_R &= t(h_3|h_1) \\ T_L &= t(h_2|h_1) \\ T_R^u &= t(h_1|h_3) \\ T_L^u &= t(h_1|h_2) \end{aligned}$$

Let $U_{x(i)}$ be the i^{th} entry of vector $U^\top \delta_x$ and $G = U^\top O$. Further, recall the notation $\operatorname{diag}(v)$, which is a matrix with elements of v on its diagonal, then:

- Define the k-dimensional vector μ (unigram counts):

$$\mu = G\pi$$
$$[\hat{\mu}]_i = \sum_{u=1}^{n} \bar{c}(u) U_{u(i)}$$

where $\bar{c}(u) = \frac{c(u)}{N_1}$, $c(u)$ is the count of observation u in the training sample, and $N_1 = \sum_{u \in n} c(u)$.

- Define the $k \times k$ matrices Σ_L and Σ_R (*left child-parent* and *right child-parent* bigram counts):

$$\begin{aligned}
[\hat{\Sigma}_L]_{i,j} &= \sum_{u=1}^{n} \sum_{v=1}^{n} \bar{c}_L(u, v) U_{u(j)} U_{v(i)} \\
\Sigma_L &= G T_L^u \text{diag}(\pi) G^\top \\
[\hat{\Sigma}_R]_{i,j} &= \sum_{u=1}^{n} \sum_{v=1}^{n} \bar{c}_R(u, v) U_{u(j)} U_{v(i)} \\
\Sigma_R &= G T_R^u \text{diag}(\pi) G^\top
\end{aligned}$$

where $\bar{c}_L(u, v) = \frac{c_L(u,v)}{N_{2_L}}$, $c_L(u, v)$ is the count of bigram (u, v) where u is the left child and v is the parent in the training sample, and $N_{2_L} = \sum_{(u,v) \in n \times n} c_L(u, v)$. Define $\bar{c}_R(u, v)$ similarly.

- Define $k \times k \times k$ tensor K (*left child-parent-right child* trigram counts):

$$\begin{aligned}
\hat{K}_{i,j,l} &= \sum_{u=1}^{n} \sum_{v=1}^{n} \sum_{w=1}^{n} \bar{c}(u, v, w) U_{w(i)} U_{u(j)} U_{v(l)} \\
K(y) &= G T_L \text{diag}(G^\top y) T_R^u \text{diag}(\pi) G^\top
\end{aligned}$$

where $\bar{c}(w, u, v) = \frac{c(w,u,v)}{N_3}$, $c(w, u, v)$ is the count of bigram (w, u, v) where w is the left child, u is the parent and v is the right child in the training sample, and $N_3 = \sum_{(w,u,v) \in n \times n \times n} c(w, u, v)$.

- Define $k \times k$ matrices Ω_L and Ω_R (skip-bigram counts (left child-right child) and (right child-left child)) [5]:

$$\begin{aligned}
[\hat{\Omega}_L]_{i,j} &= \sum_{u=1}^{n} \sum_{v=1}^{n} \sum_{w=1}^{n} \bar{c}(u, v, w) U_{w(i)} U_{u(j)} \\
\Omega_L &= G T_L T_R^u \text{diag}(\pi) G^\top \\
[\hat{\Omega}_R]_{i,j} &= \sum_{u=1}^{n} \sum_{v=1}^{n} \sum_{w=1}^{n} \bar{c}(u, v, w) U_{w(j)} U_{u(i)} \\
\Omega_R &= G T_R T_L^u \text{diag}(\pi) G^\top
\end{aligned}$$

3.3 Model estimation algorithm

Using the above definitions, we can estimate the parameters of the model, namely $\mu, \Sigma_L, \Sigma_R, \Omega_L, \Omega_R$ and K, from the training data and define observables useful for the dependency model as[6]

$$\begin{aligned}
c_1 &= \mu \\
c_\infty^T &= \mu^T \Sigma_R^{-1} \\
E_L &= \Omega_L \Sigma_R^{-1} \\
E_R &= \Omega_R \Sigma_L^{-1} \\
D(y) &= E_L^{-1} K(y) \Sigma_R^{-1}
\end{aligned}$$

As we will see, these quantities allow us to recursively compute the marginal probability of the dependency tree, $\hat{p}(x_1, \ldots, x_m)$, in a bottom up manner by using belief propagation.

To see this, let $hch(i)$ be the set of hidden children of hidden node i (in Figure 2 for instance, $hch(1) = \{2, 3\}$) and let $och(i)$ be the set of observed children of hidden node i (in the same figure $och(i) = \{1\}$). Then compute the marginal probability $p(x_1, \ldots, x_m)$ from Equation 2 as

$$r_i(h) = \prod_{j \in hch(i)} \alpha_j(h) \prod_{j \in och(i)} o(x_j | h) \qquad (3)$$

where $\alpha_i(h)$ is defined by summing over all the hidden random variables i.e., $\alpha_i(h) = \sum_{h'} p(h'|h) r_i(h')$.

This can be written in a compact matrix form as

$$\begin{aligned}
\overrightarrow{r_i}^\top = 1^\top &\prod_{j \in hch(i)} diag(T_{d_j}^\top \overrightarrow{r_j}) \\
&\cdot \prod_{j \in och(i)} diag(O^\top \delta_{x_j}) \qquad (4)
\end{aligned}$$

[5] Note than $\Omega_R = \Omega_L^T$, which is not immediately obvious from the matrix representations.

[6] The details of the derivation follow directly from the matrix versions of the variables.

where $\overrightarrow{r_i}$ is a vector of size k (the dimensionality of the hidden space) of values $r_i(h)$. Note that since in Equation 2 we condition on whether x_j is the left or right child of its parent, we have separate transition matrices for left and right transitions from a given hidden node $d_j \in \{L, R\}$.

The recursive computation can be written in terms of observables as:

$$\overrightarrow{r_i}^\top = c_\infty^\top \prod_{j \in hch(i)} D(E_{d_j}^\top \overrightarrow{r_j}) \cdot \prod_{j \in och(i)} D((U^\top U)^{-1} U^\top \delta_{x_j})$$

The final calculation for the marginal probability of a given sequence is

$$\hat{p}(x_1, \ldots, x_m) = \overrightarrow{r_1}^\top c_1 \tag{5}$$

The spectral estimation procedure is described below in Algorithm 1.

Algorithm 1 Spectral dependency parsing (Computing marginal probability of a tree.)

```
Input: Training examples- $x^{(i)}$ for $i \in \{1, \ldots, M\}$ along with dependency structures where each sequence $x^{(i)} = x_1^{(i)}, \ldots, x_{m_i}^{(i)}$.
Compute the spectral parameters $\hat{\mu}$, $\hat{\Sigma}_R$, $\hat{\Sigma}_L$, $\hat{\Omega}_R$, $\hat{\Omega}_L$, and $\hat{K}$
#Now, for a given sentence, we can recursively compute the following:
for $x_j^{(i)}$ for $j \in \{m_i, \ldots, 1\}$ do
    Compute:
    $$\overrightarrow{r_i}^\top = c_\infty^\top \prod_{j \in hch(i)} D(E_{d_j}^\top \overrightarrow{r_j}) \cdot \prod_{j \in och(i)} D((U^\top U)^{-1} U^\top \delta_{x_j})$$
end for
Finally compute
    $$\hat{p}(x_1, \ldots, x_{m_i}) = \overrightarrow{r_1}^\top c_1$$
#The marginal probability of an entire tree.
```

3.4 Sample complexity

Our main theoretical result states that the above scheme for spectral estimation of marginal probabilities provides a guaranteed consistent estimation scheme for the marginal probabilities:

Theorem 3.1. *Let the sequence $\{x_1, \ldots, x_m\}$ be generated by an $k \geq 2$ state HMM. Suppose we are given a U which has the property that $U^\top O$ is invertible, and $|U_{ij}| \leq 1$. Suppose we use equation (5) to estimate the probability based on N independent triples. Then*

$$N \quad \geq \quad C_m \frac{k^2}{\epsilon^2} \log\left(\frac{k}{\delta}\right) \tag{6}$$

where C_m is specified in the appendix, implies that

$$1 - \epsilon \leq \left| \frac{\widehat{p}(x_1, \ldots, x_m)}{p(x_1, \ldots, x_m)} \right| \leq 1 + \epsilon$$

holds with probability at least $1 - \delta$.

Proof. A sketch of the proof, in the case without directional transition parameters, can be found in the appendix. The proof with directional transition parameters is almost identical. □

4 Experimental Evaluation

Since our algorithm can score any given tree structure by computing its marginal probability, a natural way to benchmark our parser would be to generate n-best dependency trees using some standard parser and then use our algorithm to re-rank the candidate dependency trees, e.g. using the log spectral probability as described in Algorithm 1 as a feature in a discriminative re-ranker.

4.1 Experimental Setup

Our base parser was the discriminatively trained MSTParser (McDonald, 2006), which implements both first and second order parsers and is trained using MIRA (Crammer et al., 2006) and used the standard baseline features as described in McDonald (2006).

We tested our methods on the English Penn Treebank (Marcus et al., 1993). We use the standard splits of Penn Treebank; i.e., we used sections 2-21 for training, section 22 for development and section 23 for testing. We used the PennConverter[7] tool to convert Penn Treebank from constituent to dependency format. Following (McDonald, 2006; Koo

[7] http://nlp.cs.lth.se/software/treebank_converter/

et al., 2008), we used the POS tagger by Ratnaparkhi (1996) trained on the full training data to provide POS tags for development and test sets and used 10-way jackknifing to generate tags for the training set. As is common practice we stripped our sentences of all the punctuation. We evaluated our approach on sentences of all lengths.

4.2 Details of spectral learning

For the spectral learning phase, we need to just collect word counts from the training data as described above, so there are no tunable parameters as such. However, we need to have access to an attribute dictionary U which contains a k dimensional representation for each word in the corpus. A possible way of generating U as suggested by Hsu et al. (2008) is by performing SVD on bigrams P_{21} and using the left eigenvectors as U. We instead used the eigenfeature dictionary proposed by Dhillon et al. (2011) (LR-MVL) which is obtained by performing CCA on neighboring words and has provably better sample complexity for rare words compared to the SVD alternative.

We induced the LR-MVL embeddings for words using the Reuters RCV1 corpus which contains about 63 million tokens in 3.3 million sentences and used their context oblivious embeddings as our estimate of U. We experimented with different choices of k (the size of the low dimensional projection) on the development set and found $k = 10$ to work reasonably well and fast. Using $k = 10$ we were able to estimate our spectral learning parameters $\mu, \Sigma_{L,R}, \Omega_{L,R}, K$ from the entire training data in under 2 minutes on a 64 bit Intel 2.4 Ghz processor.

4.3 Re-ranking the outputs of MST parser

We could not find any previous work which describes features for discriminative re-ranking for dependency parsing, which is due to the fact that unlike constituency parsing, the base parsers for dependency parsing are discriminative (e.g. MST parser) which obviates the need for re-ranking as one could add a variety of features to the baseline parser itself. However, parse re-ranking is a good testbed for our spectral dependency parser which can score a given tree. So, we came up with a baseline set of features to use in an averaged perceptron reranker (Collins, 2002). Our baseline features comprised of two main features which capture information that varies across the different n-best parses and moreover were not used as features by the baseline MST parser, $< POS-left-modifier,\ POS-head,\ POS-right-modifier >$ and $< POS-left/right-modifier,\ POS-head,\ POS-grandparent >$ [8]. In addition to that we used the log of spectral probability ($\hat{p}(x_1, \ldots, x_m)$ - as calculated using Algorithm 1) as a feature.

Method	Accuracy	Complete
I Order		
MST Parser	90.8	37.2
+Base. Features	91.3	37.5
+Base. Features +$\log \hat{p}$	**91.7**	**37.8**
II Order		
MST Parser	91.8	40.6
+Base. Features	92.4	41.0
+Base. Features +$\log \hat{p}$	**92.7**	**41.3**

Table 1: (Unlabeled) Dependency Parse re-ranking results for English test set (Section 23). **Note:** 1). *Accuracy* is the number of words which correctly identified their parent and *Complete* is the number of sentences for which the entire dependency tree was correct. 2). Base. Features are the two re-ranking features described in Section 4.3. 3). $\log \hat{p}$ is the spectral log probability feature.

We used the MST parser trained on entire training data to generate a list of n-best parses for the development and test sets. The n-best parses for the training set were generated by 3-fold cross validation, where we train on 2 folds to get the parses for the third fold. In all our experiments we used $n = 50$. The results are shown in Table 1. As can be seen, the best results give up to 4.6% reduction in error over the re-ranker which uses just the baseline set of features.

5 Discussion and Future Work

Spectral learning of structured latent variable models in general is a promising direction as has been shown by the recent interest in this area. It allows us to circumvent the ubiquitous problem of getting stuck in local minima when estimating the la-

[8] Its conceivable to come up with better and more linguistically informed features for dependency parse re-ranking. For the purpose of this paper, we just wanted to get a decent baseline.

tent variable models via EM. In this paper we extended the spectral learning ideas to learn a simple yet powerful dependency parser. As future work, we are working on building an end-to-end parser which would involve coming up with a spectral version of the inside-outside algorithm for our setting and further extend it to learn more powerful grammars e.g. split head-automata grammars (SHAG) (Eisner and Satta, 1999).

6 Conclusion

In this paper we proposed a novel spectral method for dependency parsing. Unlike EM trained generative latent variable models, our method does not get stuck in local optima, it gives consistent parameter estimates, and it is extremely fast to train. We worked out the theory of a simple yet powerful generative model and showed how it can be learned using a spectral method. As pilot experimental evaluations we showed the efficacy of our approach by using the spectral probabilities output by our model as features in the MST parser as well as for re-ranking its outputs. Our method reduced the error of the baseline parser by up to a moderate 4.6%.

7 Appendix

This appendix offers a sketch of the proof of Theorem 1. The proof uses the following definitions, which are slightly modified from those of Foster et al. (2012).

Definition 1. *Define Λ as the smallest element of μ, Σ^{-1}, Ω^{-1}, and $K()$. In other words,*

$$\Lambda \equiv \min\{\min_i |\mu_i|, \min_{i,j} |\Sigma_{ij}^{-1}|, \min_{i,j} |\Omega_{ij}^{-1}|, \min_{i,j,k} |K_{ijk}|, \min_{i,j} |\Sigma_{ij}|, \min_{i,j} |\Omega_{ij}|, \}$$

where $K_{ijk} = K(\delta_j)_{ik}$ are the elements of the tensor $K()$.

Definition 2. *Define σ_k as the smallest singular value of Σ and Ω.*

The proof relies on the fact that a row vector multiplied by a series of matrices, and finally multiplied by a column vector amounts to a sum over all possible products of individual entries in the vectors and matrices. With this in mind, if we bound the largest relative error of any particular entry in the matrix by, say, ω, and there are, say, s parameters (vectors and matrices) being multiplied together, then by simple algebra the total relative error of the sum over the products is bounded by ω^s.

The proof then follows from two basic steps. First, one must bound the maximal relative error, ω for any particular entry in the parameters, which can be done using central limit-type theorems and the quantity Λ described above. Then, to calculate the exponent s one simply counts the number of parameters multiplied together when calculating the probability of a particular sequence of observations.

Since each hidden node is associated with exactly one observed node, it follows that $s = 12m + 2L$, where L is the number of levels (for instance in our example “Kilroy was here” there are two levels). s can be easily computed for arbitrary tree topologies.

It follows from Foster et al. (2012) that we achieve a sample complexity

$$N \geq \frac{128k^2s^2}{\epsilon^2\, \Lambda^2\sigma_k^4} \log\left(\frac{2k}{\delta}\right) \cdot \overbrace{\frac{\epsilon^2/s^2}{(\sqrt[s]{1+\epsilon}-1)^2}}^{\approx 1} \tag{7}$$

leading to the theorem stated above.

Lastly, note that in reality one does not see Λ and σ_k but instead estimates of these quantities; Foster et al. (2012) shows how to incorporate the accuracy of the estimates into the sample complexity.

Acknowledgement: We would like to thank Emily Pitler for valuable feedback on the paper.

References

Shay Cohen, Karl Stratos, Michael Collins, Dean Foster, and Lyle Ungar. Spectral learning of latent-variable pcfgs. In *Association of Computational Linguistics (ACL)*, volume 50, 2012.

Michael Collins. Ranking algorithms for named-entity extraction: boosting and the voted perceptron. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL ’02, pages 489–496, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics. URL http://dx.doi.org/10.3115/1073083.1073165.

Michael Collins and Terry Koo. Discriminative reranking for natural language parsing. *Comput. Linguist.*, 31(1):25–70, March 2005. ISSN 0891-2017.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585, 2006.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *JRSS, SERIES B*, 39(1):1–38, 1977.

Paramveer S. Dhillon, Dean Foster, and Lyle Ungar. Multi-view learning of word embeddings via cca. In *Advances in Neural Information Processing Systems (NIPS)*, volume 24, 2011.

Jason Eisner and Giorgio Satta. Efficient parsing for bilexical context-free grammars and head-automaton grammars. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 457–464, University of Maryland, June 1999. URL `http://cs.jhu.edu/~jason/papers/#acl99`.

Dean Foster, Jordan Rodu, and Lyle Ungar. Spectral dimensionality reduction for HMMs. *ArXiV http://arxiv.org/abs/1203.6130*, 2012.

D Hsu, S M. Kakade, and Tong Zhang. A spectral algorithm for learning hidden markov models. *arXiv:0811.4413v2*, 2008.

Terry Koo, Xavier Carreras, and Michael Collins. Simple semi-supervised dependency parsing. In *In Proc. ACL/HLT*, 2008.

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

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of english: the penn treebank. *Comput. Linguist.*, 19:313–330, June 1993. ISSN 0891-2017.

Ryan McDonald. *Discriminative learning and spanning tree algorithms for dependency parsing*. PhD thesis, University of Pennsylvania, Philadelphia, PA, USA, 2006. AAI3225503.

Bernard Merialdo. Tagging english text with a probabilistic model. *Comput. Linguist.*, 20:155–171, June 1994. ISSN 0891-2017.

Gabriele Antonio Musillo and Paola Merlo. Unlexicalised hidden variable models of split dependency grammars. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers*, HLT-Short '08, pages 213–216, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics.

Ankur P. Parikh, Le Song, and Eric P. Xing. A spectral algorithm for latent tree graphical models. In *ICML*, pages 1065–1072, 2011.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, ACL-44, pages 433–440, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.

Ariadna Quattoni, Michael Collins, and Trevor Darrell. Conditional random fields for object recognition. In *In NIPS*, pages 1097–1104. MIT Press, 2004.

Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. In *Proceedings of the IEEE*, pages 257–286, 1989.

Adwait Ratnaparkhi. A Maximum Entropy Model for Part-Of-Speech Tagging. In Eric Brill and Kenneth Church, editors, *Proceedings of the Empirical Methods in Natural Language Processing*, pages 133–142, 1996.

Sebastiaan Terwijn. On the learnability of hidden markov models. In *Proceedings of the 6th International Colloquium on Grammatical Inference: Algorithms and Applications*, ICGI '02, pages 261–268, London, UK, UK, 2002. Springer-Verlag. ISBN 3-540-44239-1.