Viterbi Training for PCFGs: Hardness Results and Competitiveness of Uniform Initialization

Shay B. Cohen and **Noah A. Smith**
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, USA
{scohen,nasmith}@cs.cmu.edu

Abstract

We consider the search for a maximum likelihood assignment of hidden derivations and grammar weights for a probabilistic context-free grammar, the problem approximately solved by "Viterbi training." We show that solving and even approximating Viterbi training for PCFGs is NP-hard. We motivate the use of uniform-at-random initialization for Viterbi EM as an optimal initializer in absence of further information about the correct model parameters, providing an approximate bound on the log-likelihood.

1 Introduction

Probabilistic context-free grammars are an essential ingredient in many natural language processing models (Charniak, 1997; Collins, 2003; Johnson et al., 2006; Cohen and Smith, 2009, *inter alia*). Various algorithms for training such models have been proposed, including unsupervised methods. Many of these are based on the expectation-maximization (EM) algorithm.

There are alternatives to EM, and one such alternative is Viterbi EM, also called "hard" EM or "sparse" EM (Neal and Hinton, 1998). Instead of using the parameters (which are maintained in the algorithm's current state) to find the true posterior over the derivations, Viterbi EM algorithm uses a posterior focused on the Viterbi parse of those parameters. Viterbi EM and variants have been used in various settings in natural language processing (Yejin and Cardie, 2007; Wang et al., 2007; Goldwater and Johnson, 2005; DeNero and Klein, 2008; Spitkovsky et al., 2010).

Viterbi EM can be understood as a coordinate ascent procedure that locally optimizes a function; we call this optimization goal "Viterbi training."

In this paper, we explore Viterbi training for probabilistic context-free grammars. We first show that under the assumption that P $\neq$ NP, solving and even approximating the Viterbi training problem is hard. This result holds even for hidden Markov models. We extend the main hardness result to the EM algorithm (giving an alternative proof to this known result), as well as the problem of *conditional* Viterbi training. We then describe a "competitiveness" result for uniform initialization of Viterbi EM: we show that initialization of the trees in an E-step which uses uniform distributions over the trees is optimal with respect to a certain approximate bound.

The rest of this paper is organized as follows. §2 gives background on PCFGs and introduces some notation. §3 explains Viterbi training, the declarative form of Viterbi EM. §4 describes a hardness result for Viterbi training. §5 extends this result to a hardness result of approximation and §6 further extends these results for other cases. §7 describes the advantages in using uniform-at-random initialization for Viterbi training. We relate these results to work on the k-means problem in §8.

2 Background and Notation

We assume familiarity with probabilistic context-free grammars (PCFGs). A PCFG $\boldsymbol{G}$ consists of:

- A finite set of nonterminal symbols $\mathcal{N}$;
- A finite set of terminal symbols Σ;
- For each $A \in \mathcal{N}$, a set of rewrite rules $R(A)$ of the form $A \rightarrow \alpha$, where $\alpha \in (\mathcal{N} \cup \Sigma)^*$, and $\mathcal{R} = \cup_{A \in \mathcal{N}} R(A)$;
- For each rule $A \rightarrow \alpha$, a probability $\theta_{A \rightarrow \alpha}$. The collection of probabilities is denoted $\boldsymbol{\theta}$, and they are constrained such that:

$$\begin{aligned} \forall (A \rightarrow \alpha) \in R(A), \ \theta_{A \rightarrow \alpha} &\geq 0 \\ \forall A \in \mathcal{N}, \ \sum_{\alpha : (A \rightarrow \alpha) \in R(A)} \theta_{A \rightarrow \alpha} &= 1 \end{aligned}$$

That is, $\boldsymbol{\theta}$ is grouped into $|\mathcal{N}|$ multinomial distributions.

Under the PCFG, the joint probability of a string $x \in \Sigma^*$ and a grammatical derivation z is[1]

$$p(x, z \mid \boldsymbol{\theta}) = \prod_{(A\to\alpha)\in\mathcal{R}} (\theta_{A\to\alpha})^{f_{A\to\alpha}(z)} \quad (1)$$
$$= \exp \sum_{(A\to\alpha)\in\mathcal{R}} f_{A\to\alpha}(z) \log \theta_{A\to\alpha}$$

where $f_{A\to\alpha}(z)$ is a function that "counts" the number of times the rule $A \to \alpha$ appears in the derivation z. $f_A(z)$ will similarly denote the number of times that nonterminal A appears in z. Given a sample of derivations $\boldsymbol{z} = \langle z_1, \ldots, z_n \rangle$, let:

$$F_{A\to\alpha}(\boldsymbol{z}) = \sum_{i=1}^{n} f_{A\to\alpha}(z_i) \quad (2)$$

$$F_A(\boldsymbol{z}) = \sum_{i=1}^{n} f_A(z_i) \quad (3)$$

We use the following notation for $\boldsymbol{G}$:

- $L(\boldsymbol{G})$ is the set of all strings (sentences) x that can be generated using the grammar $\boldsymbol{G}$ (the "language of $\boldsymbol{G}$").
- $D(\boldsymbol{G})$ is the set of all possible derivations z that can be generated using the grammar $\boldsymbol{G}$.
- $D(\boldsymbol{G}, x)$ is the set of all possible derivations z that can be generated using the grammar $\boldsymbol{G}$ and have the yield x.

3 Viterbi Training

Viterbi EM, or "hard" EM, is an unsupervised learning algorithm, used in NLP in various settings (Yejin and Cardie, 2007; Wang et al., 2007; Goldwater and Johnson, 2005; DeNero and Klein, 2008; Spitkovsky et al., 2010). In the context of PCFGs, it aims to select parameters $\boldsymbol{\theta}$ and phrase-structure trees $\boldsymbol{z}$ jointly. It does so by iteratively updating a state consisting of $(\boldsymbol{\theta}, \boldsymbol{z})$. The state is initialized with some value, then the algorithm alternates between (i) a "hard" E-step, where the strings $x_1, \ldots, x_n$ are parsed according to a current, fixed $\boldsymbol{\theta}$, giving new values for $\boldsymbol{z}$, and (ii) an M-step, where the $\boldsymbol{\theta}$ are selected to maximize likelihood, with $\boldsymbol{z}$ fixed.

With PCFGs, the E-step requires running an algorithm such as (probabilistic) CKY or Earley's algorithm, while the M-step normalizes frequency counts $F_{A\to\alpha}(\boldsymbol{z})$ to obtain the maximum likelihood estimate's closed-form solution.

We can understand Viterbi EM as a coordinate ascent procedure that approximates the solution to the following declarative problem:

Problem 1. ViterbiTrain
Input: $\boldsymbol{G}$ *context-free grammar,* $x_1, \ldots, x_n$ *training instances from* $L(\boldsymbol{G})$
Output: $\boldsymbol{\theta}$ *and* $z_1, \ldots, z_n$ *such that*

$$(\boldsymbol{\theta}, z_1, \ldots, z_n) = \operatorname*{argmax}_{\boldsymbol{\theta}, \boldsymbol{z}} \prod_{i=1}^{n} p(x_i, z_i \mid \boldsymbol{\theta}) \quad (4)$$

The optimization problem in Eq. 4 is non-convex and, as we will show in §4, hard to optimize. Therefore it is necessary to resort to approximate algorithms like Viterbi EM.

Neal and Hinton (1998) use the term "sparse EM" to refer to a version of the EM algorithm where the E-step finds the modes of hidden variables (rather than marginals as in standard EM). Viterbi EM is a variant of this, where the E-step finds the mode for each x_i's derivation, $\operatorname{argmax}_{z\in D(\boldsymbol{G},x_i)} p(x_i, z \mid \boldsymbol{\theta})$.

We will refer to

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{z}) = \prod_{i=1}^{n} p(x_i, z_i \mid \boldsymbol{\theta}) \quad (5)$$

as "the objective function of ViterbiTrain."

Viterbi training and Viterbi EM are closely related to **self-training**, an important concept in semi-supervised NLP (Charniak, 1997; McClosky et al., 2006a; McClosky et al., 2006b). With self-training, the model is learned with some seed annotated data, and then iterates by labeling new, unannotated data and adding it to the original annotated training set. McClosky et al. consider self-training to be "one round of Viterbi EM" with supervised initialization using labeled seed data. We refer the reader to Abney (2007) for more details.

4 Hardness of Viterbi Training

We now describe hardness results for Problem 1. We first note that the following problem is known to be NP-hard, and in fact, NP-complete (Sipser, 2006):

Problem 2. 3-SAT
Input: *A formula* $\phi = \bigwedge_{i=1}^{m} (a_i \vee b_i \vee c_i)$ *in conjunctive normal form, such that each clause has* 3

[1] Note that $x = \text{yield}(z)$; if the derivation is known, the string is also known. On the other hand, there may be many derivations with the same yield, perhaps even infinitely many.

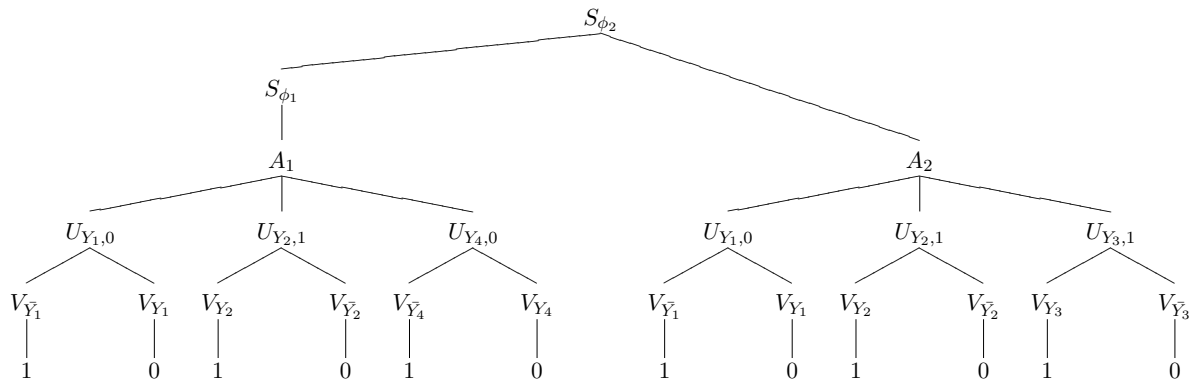

Figure 1: An example of a Viterbi parse tree which represents a satisfying assignment for $\phi = (Y_1 \vee Y_2 \vee \bar{Y_4}) \wedge (\bar{Y_1} \vee \bar{Y_2} \vee Y_3)$. In $\boldsymbol{\theta}_\phi$, all rules appearing in the parse tree have probability 1. The extracted assignment would be $Y_1 = 0, Y_2 = 1, Y_3 = 1, Y_4 = 0$. Note that there is no usage of two different rules for a single nonterminal.

literals.

Output: *1 if there is a satisfying assignment for ϕ and 0 otherwise.*

We now describe a reduction of 3-SAT to Problem 1. Given an instance of the 3-SAT problem, the reduction will, in polynomial time, create a grammar and a single string such that solving the ViterbiTrain problem for this grammar and string will yield a solution for the instance of the 3-SAT problem.

Let $\phi = \bigwedge_{i=1}^{m} (a_i \vee b_i \vee c_i)$ be an instance of the 3-SAT problem, where a_i, b_i and c_i are literals over the set of variables $\{Y_1, \ldots, Y_N\}$ (a literal refers to a variable Y_j or its negation, $\bar{Y_j}$). Let C_j be the jth clause in ϕ, such that $C_j = a_j \vee b_j \vee c_j$. We define the following context-free grammar $\boldsymbol{G}_\phi$ and string to parse s_ϕ:

1. The terminals of $\boldsymbol{G}_\phi$ are the binary digits $\Sigma = \{0, 1\}$.
2. We create N nonterminals V_{Y_r}, $r \in \{1, \ldots, N\}$ and rules $V_{Y_r} \to 0$ and $V_{Y_r} \to 1$.
3. We create N nonterminals $V_{\bar{Y}_r}$, $r \in \{1, \ldots, N\}$ and rules $V_{\bar{Y}_r} \to 0$ and $V_{\bar{Y}_r} \to 1$.
4. We create $U_{Y_r,1} \to V_{Y_r} V_{\bar{Y}_r}$ and $U_{Y_r,0} \to V_{\bar{Y}_r} V_{Y_r}$.
5. We create the rule $S_{\phi_1} \to A_1$. For each $j \in \{2, \ldots, m\}$, we create a rule $S_{\phi_j} \to S_{\phi_{j-1}} A_j$ where S_{ϕ_j} is a new nonterminal indexed by $\phi_j \triangleq \bigwedge_{i=1}^{j} C_i$ and A_j is also a new nonterminal indexed by $j \in \{1, \ldots, m\}$.
6. Let $C_j = a_j \vee b_j \vee c_j$ be clause j in ϕ. Let $Y(a_j)$ be the variable that a_j mentions. Let (y_1, y_2, y_3) be a satisfying assignment for C_j where $y_k \in \{0, 1\}$ and is the value of $Y(a_j)$, $Y(b_j)$ and $Y(c_j)$ respectively for $k \in \{1, 2, 3\}$. For each such clause-satisfying assignment, we add the rule:
$$A_j \to U_{Y(a_j),y_1} U_{Y(b_j),y_2} U_{Y(c_j),y_3} \qquad (6)$$
For each A_j, we would have at most 7 rules of that form, since one rule will be logically inconsistent with $a_j \vee b_j \vee c_j$.
7. The grammar's start symbol is S_{ϕ_n}.
8. The string to parse is $s_\phi = (10)^{3m}$, i.e. $3m$ consecutive occurrences of the string 10.

A parse of the string s_ϕ using $\boldsymbol{G}_\phi$ will be used to get an assignment by setting $Y_r = 0$ if the rule $V_{Y_r} \to 0$ or $V_{\bar{Y}_r} \to 1$ are used in the derivation of the parse tree, and 1 otherwise. Notice that at this point we do not exclude "contradictions" coming from the parse tree, such as $V_{Y_3} \to 0$ used in the tree together with $V_{Y_3} \to 1$ or $V_{\bar{Y}_3} \to 0$. The following lemma gives a condition under which the assignment is consistent (so contradictions do not occur in the parse tree):

Lemma 1. *Let ϕ be an instance of the 3-SAT problem, and let $\boldsymbol{G}_\phi$ be a probabilistic CFG based on the above grammar with weights $\boldsymbol{\theta}_\phi$. If the (multiplicative) weight of the Viterbi parse of s_ϕ is 1, then the assignment extracted from the parse tree is consistent.*

Proof. Since the probability of the Viterbi parse is 1, all rules of the form $\{V_{Y_r}, V_{\bar{Y}_r}\} \to \{0, 1\}$ which appear in the parse tree have probability 1 as well. There are two possible types of inconsistencies. We show that neither exists in the Viterbi parse:

1. For any r, an appearance of both rules of the form $V_{Y_r} \to 0$ and $V_{Y_r} \to 1$ cannot occur because all rules that appear in the Viterbi parse tree have probability 1.
2. For any r, an appearance of rules of the form $V_{Y_r} \to 1$ and $V_{\bar{Y_r}} \to 1$ cannot occur, because whenever we have an appearance of the rule $V_{Y_r} \to 0$, we have an adjacent appearance of the rule $V_{\bar{Y_r}} \to 1$ (because we parse substrings of the form 10), and then again we use the fact that all rules in the parse tree have probability 1. The case of $V_{Y_r} \to 0$ and $V_{\bar{Y_r}} \to 0$ is handled analogously.

Thus, both possible inconsistencies are ruled out, resulting in a consistent assignment. □

Figure 1 gives an example of an application of the reduction.

Lemma 2. *Define ϕ, $\boldsymbol{G}_\phi$ as before. There exists $\boldsymbol{\theta}_\phi$ such that the Viterbi parse of s_ϕ is 1 if and only if ϕ is satisfiable. Moreover, the satisfying assignment is the one extracted from the parse tree with weight 1 of s_ϕ under $\boldsymbol{\theta}_\phi$.*

Proof. ($\Longrightarrow$) Assume that there is a satisfying assignment. Each clause $C_j = a_j \vee b_j \vee c_j$ is satisfied using a tuple (y_1, y_2, y_3) which assigns value for $Y(a_j)$, $Y(b_j)$ and $Y(c_j)$. This assignment corresponds the following rule

$$A_j \to U_{Y(a_j),y_1} U_{Y(b_j),y_2} U_{Y(c_j),y_3} \qquad (7)$$

Set its probability to 1, and set all other rules of A_j to 0. In addition, for each r, if $Y_r = y$, set the probabilities of the rules $V_{Y_r} \to y$ and $V_{\bar{Y_r}} \to 1-y$ to 1 and $V_{\bar{Y_r}} \to y$ and $V_{Y_r} \to 1 - y$ to 0. The rest of the weights for $S_{\phi_j} \to S_{\phi_{j-1}} A_j$ are set to 1. This assignment of rule probabilities results in a Viterbi parse of weight 1.

($\Longleftarrow$) Assume that the Viterbi parse has probability 1. From Lemma 1, we know that we can extract a consistent assignment from the Viterbi parse. In addition, for each clause C_j we have a rule

$$A_j \to U_{Y(a_j),y_1} U_{Y(b_j),y_2} U_{Y(c_j),y_3} \qquad (8)$$

that is assigned probability 1, for some (y_1, y_2, y_3). One can verify that (y_1, y_2, y_3) are the values of the assignment for the corresponding variables in clause C_j, and that they satisfy this clause. This means that each clause is satisfied by the assignment we extracted. □

In order to show an NP-hardness result, we need to "convert" ViterbiTrain to a decision problem. The natural way to do it, following Lemmas 1 and 2, is to state the decision problem for ViterbiTrain as "given $\boldsymbol{G}$ and $x_1, \ldots, x_n$ and $\alpha \geq 0$, is the optimized value of the objective function $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{z}) \geq \alpha$?" and use $\alpha = 1$ together with Lemmas 1 and 2. (Naturally, an algorithm for solving ViterbiTrain can easily be used to solve its decision problem.)

Theorem 3. *The decision version of the* ViterbiTrain *problem is NP-hard.*

5 Hardness of Approximation

A natural path of exploration following the hardness result we showed is determining whether an *approximation* of ViterbiTrain is also hard. Perhaps there is an efficient approximation algorithm for ViterbiTrain we could use instead of coordinate ascent algorithms such as Viterbi EM. Recall that such algorithms' main guarantee is identifying a local maximum; we know nothing about how far it will be from the global maximum.

We next show that approximating the objective function of ViterbiTrain with a constant factor of ρ is hard for any $\rho \in (\frac{1}{2}, 1]$ (i.e., $1/2 + \epsilon$ approximation is hard for any $\epsilon \leq 1/2$). This means that, under the P $\neq$ NP assumption, there is no efficient algorithm that, given a grammar $\boldsymbol{G}$ and a sample of sentences $x_1, \ldots, x_n$, returns $\boldsymbol{\theta}'$ and $\boldsymbol{z}'$ such that:

$$\mathcal{L}(\boldsymbol{\theta}', \boldsymbol{z}') \quad \geq \quad \rho \cdot \max_{\boldsymbol{\theta}, \boldsymbol{z}} \prod_{i=1}^{n} p(x_i, z_i \mid \boldsymbol{\theta}) \qquad (9)$$

We will continue to use the same reduction from §4. Let s_ϕ be the string from that reduction, and let $(\boldsymbol{\theta}, z)$ be the optimal solution for ViterbiTrain given $\boldsymbol{G}_\phi$ and s_ϕ. We first note that if $p(s_\phi, z \mid \boldsymbol{\theta}) < 1$ (implying that there is no satisfying assignment), then there must be a nonterminal which appears along with two different rules in z.

This means that we have a nonterminal $B \in \mathcal{N}$ with some rule $B \to \alpha$ that appears k times, while the nonterminal appears in the parse $r \geq k + 1$ times. Given the tree z, the $\boldsymbol{\theta}$ that maximizes the objective function is the maximum likelihood estimate (MLE) for z (counting and normalizing the rules).[2] We therefore know that the ViterbiTrain objective function, $\mathcal{L}(\boldsymbol{\theta}, z)$, is at

[2] Note that we can only make $p(z \mid \boldsymbol{\theta}, x)$ greater by using $\boldsymbol{\theta}$ to be the MLE for the derivation z.

most $\left(\frac{k}{r}\right)^k$, because it includes a factor equal to $\left(\frac{f_{B\to\alpha}(z)}{f_B(z)}\right)^{f_{B\to\alpha}(z)}$, where $f_B(z)$ is the number of times nonterminal B appears in z (hence $f_B(z) = r$) and $f_{B\to\alpha}(z)$ is the number of times $B \to \alpha$ appears in z (hence $f_{B\to\alpha}(z) = k$). For any $k \geq 1, r \geq k+1$:

$$\left(\frac{k}{r}\right)^k \leq \left(\frac{k}{k+1}\right)^k \leq \frac{1}{2} \qquad (10)$$

This means that if the value of the objective function of ViterbiTrain is not 1 using the reduction from §4, then it is at most $\frac{1}{2}$. If we had an efficient approximate algorithm with approximation coefficient $\rho > \frac{1}{2}$ (Eq. 9 holds), then in order to solve 3-SAT for formula ϕ, we could run the algorithm on $\boldsymbol{G}_\phi$ and s_ϕ and check whether the assignment to $(\boldsymbol{\theta}, z)$ that the algorithm returns satisfies ϕ or not, and return our response accordingly.

If ϕ were satisfiable, then the true maximal value of $\mathfrak{L}$ would be 1, and the approximation algorithm would return $(\boldsymbol{\theta}, z)$ such that $\mathfrak{L}(\boldsymbol{\theta}, z) \geq \rho > \frac{1}{2}$. z would have to correspond to a satisfying assignment, and in fact $p(z \mid \boldsymbol{\theta}) = 1$, because in any other case, the probability of a derivation which does not represent a satisfying assignment is smaller than $\frac{1}{2}$. If ϕ were not satisfiable, then the approximation algorithm would never return a $(\boldsymbol{\theta}, z)$ that results in a satisfying assignment (because such a $(\boldsymbol{\theta}, z)$ does not exist).

The conclusion is that an efficient algorithm for approximating the objective function of ViterbiTrain (Eq. 4) within a factor of $\frac{1}{2} + \epsilon$ is unlikely to exist. If there were such an algorithm, we could use it to solve 3-SAT using the reduction from §4.

6 Extensions of the Hardness Result

An alternative problem to Problem 1, a variant of Viterbi-training, is the following (see, for example, Klein and Manning, 2001):

Problem 3. ConditionalViterbiTrain
Input: $\boldsymbol{G}$ *context-free grammar,* $x_1, \ldots, x_n$ *training instances from* $L(\boldsymbol{G})$
Output: $\boldsymbol{\theta}$ *and* $z_1, \ldots, z_n$ *such that*

$$(\boldsymbol{\theta}, z_1, \ldots, z_n) = \operatorname*{argmax}_{\boldsymbol{\theta}, \boldsymbol{z}} \prod_{i=1}^{n} p(z_i \mid \boldsymbol{\theta}, x_i) \quad (11)$$

Here, instead of maximizing the likelihood, we maximize the *conditional* likelihood. Note that there is a hidden assumption in this problem definition, that x_i can be parsed using the grammar $\boldsymbol{G}$. Otherwise, the quantity $p(z_i \mid \boldsymbol{\theta}, x_i)$ is not well-defined. We can extend ConditionalViterbiTrain to return $\perp$ in the case of not having a parse for one of the x_i—this can be efficiently checked using a run of a cubic-time parser on each of the strings x_i with the grammar $\boldsymbol{G}$.

An approximate technique for this problem is similar to Viterbi EM, only modifying the M-step to maximize the conditional, rather than joint, likelihood. This new M-step will not have a closed form and may require auxiliary optimization techniques like gradient ascent.

Our hardness result for ViterbiTrain applies to ConditionalViterbiTrain as well. The reason is that if $p(z, s_\phi \mid \boldsymbol{\theta}_\phi) = 1$ for a ϕ with a satisfying assignment, then $L(\boldsymbol{G}) = \{s_\phi\}$ and $D(\boldsymbol{G}) = \{z\}$. This implies that $p(z \mid \boldsymbol{\theta}_\phi, s_\phi) = 1$. If ϕ is unsatisfiable, then for the optimal $\boldsymbol{\theta}$ of ViterbiTrain we have z and z' such that $0 < p(z, s_\phi \mid \boldsymbol{\theta}_\phi) < 1$ and $0 < p(z', s_\phi \mid \boldsymbol{\theta}_\phi) < 1$, and therefore $p(z \mid \boldsymbol{\theta}_\phi, s_\phi) < 1$, which means the conditional objective function will not obtain the value 1. (Note that there always exist some parameters $\boldsymbol{\theta}_\phi$ that generate s_ϕ.) So, again, given an algorithm for ConditionalViterbiTrain, we can discern between a satisfiable formula and an unsatisfiable formula, using the reduction from §4 with the given algorithm, and identify whether the value of the objective function is 1 or strictly less than 1. We get the result that:

Theorem 4. *The decision problem of* ConditionalViterbiTrain *problem is NP-hard.*

where the decision problem of ConditionalViterbiTrain is defined analogously to the decision problem of ViterbiTrain.

We can similarly show that finding the global maximum of the marginalized likelihood:

$$\max_{\boldsymbol{\theta}} \frac{1}{n} \sum_{i=1}^{n} \log \sum_{z} p(x_i, z \mid \boldsymbol{\theta}) \qquad (12)$$

is NP-hard. The reasoning follows. Using the reduction from before, if ϕ is satisfiable, then Eq. 12 gets value 0. If ϕ is unsatisfiable, then we would still get value 0 only if $L(\boldsymbol{G}) = \{s_\phi\}$. If $\boldsymbol{G}_\phi$ generates a single derivation for $(10)^{3m}$, then we actually do have a satisfying assignment from

Lemma 1. Otherwise (more than a single derivation), the optimal $\boldsymbol{\theta}$ would have to give fractional probabilities to rules of the form $V_{Y_r} \rightarrow \{0, 1\}$ (or $V_{\bar{Y}_r} \rightarrow \{0, 1\}$). In that case, it is no longer true that $(10)^{3m}$ is the only generated sentence, which is a contradiction.

The quantity in Eq. 12 can be maximized approximately using algorithms like EM, so this gives a hardness result for optimizing the objective function of EM for PCFGs. Day (1983) previously showed that maximizing the marginalized likelihood for hidden Markov models is NP-hard.

We note that the grammar we use for all of our results is not recursive. Therefore, we can encode this grammar as a hidden Markov model, strengthening our result from PCFGs to HMMs.[3]

7 Uniform-at-Random Initialization

In the previous sections, we showed that solving Viterbi training is hard, and therefore requires an approximation algorithm. Viterbi EM, which is an example of such algorithm, is dependent on an initialization of either $\boldsymbol{\theta}$ to start with an E-step or $\boldsymbol{z}$ to start with an M-step. In the absence of a better-informed initializer, it is reasonable to initialize $\boldsymbol{z}$ using a uniform distribution over $D(\boldsymbol{G}, x_i)$ for each i. If $D(\boldsymbol{G}, x_i)$ is finite, it can be done efficiently by setting $\boldsymbol{\theta} = \mathbf{1}$ (ignoring the normalization constraint), running the inside algorithm, and sampling from the (unnormalized) posterior given by the chart (Johnson et al., 2007). We turn next to an analysis of this initialization technique that suggests it is well-motivated.

The sketch of our result is as follows: we first give an asymptotic upper bound for the log-likelihood of derivations and sentences. This bound, which has an information-theoretic interpretation, depends on a parameter λ, which depends on the distribution from which the derivations were chosen. We then show that this bound is minimized when we pick λ such that this distribution is (conditioned on the sentence) a uniform distribution over derivations.

Let $q(x)$ be any distribution over $L(\boldsymbol{G})$ and $\boldsymbol{\theta}$ some parameters for $\boldsymbol{G}$. Let $f(z)$ be some feature function (such as the one that counts the number of appearances of a certain rule in a derivation), and then:

$$\mathbb{E}_{q,\boldsymbol{\theta}}[f] \triangleq \sum_{x \in L(\boldsymbol{G})} q(x) \sum_{z \in D(\boldsymbol{G},x)} p(z \mid \boldsymbol{\theta}, x) f(z)$$

which gives the expected value of the feature function $f(z)$ under the distribution $q(x) \times p(z \mid \boldsymbol{\theta}, x)$. We will make the following assumption about $\boldsymbol{G}$:

Condition 1. *There exists some $\boldsymbol{\theta}_I$ such that $\forall x \in L(\boldsymbol{G}), \forall z \in D(\boldsymbol{G}, x)$, $p(z \mid \boldsymbol{\theta}_I, x) = 1/|D(\boldsymbol{G}, x)|$.*

This condition is satisfied, for example, when $\boldsymbol{G}$ is in Chomsky normal form and for all $A, A' \in \mathcal{N}$, $|R(A)| = |R(A')|$. Then, if we set $\theta_{A \rightarrow \alpha} = 1/|R(A)|$, we get that all derivations of x will have the same number of rules and hence the same probability. This condition does not hold for grammars with unary cycles because $|D(\boldsymbol{G}, x)|$ may be infinite for some derivations. Such grammars are not commonly used in NLP.

Let us assume that some "correct" parameters $\boldsymbol{\theta}^*$ exist, and that our data were drawn from a distribution parametrized by $\boldsymbol{\theta}^*$. The goal of this section is to motivate the following initialization for $\boldsymbol{\theta}$, which we call UniformInit:

1. Initialize $\boldsymbol{z}$ by sampling from the uniform distribution over $D(\boldsymbol{G}, x_i)$ for each x_i.
2. Update the grammar parameters using maximum likelihood estimation.

7.1 Bounding the Objective

To show our result, we require first the following definition due to Freund et al. (1997):

Definition 5. *A distribution p_1 is within $\lambda \geq 1$ of a distribution p_2 if for every event A, we have*

$$\frac{1}{\lambda} \leq \frac{p_1(A)}{p_2(A)} \leq \lambda \tag{13}$$

For any feature function $f(z)$ and any two sets of parameters $\boldsymbol{\theta}_2$ and $\boldsymbol{\theta}_1$ for $\boldsymbol{G}$ and for any marginal $q(x)$, if $p(z \mid \boldsymbol{\theta}_1, x)$ is within λ of $p(z \mid \boldsymbol{\theta}_2, x)$ for all x then:

$$\frac{\mathbb{E}_{q,\boldsymbol{\theta}_1}[f]}{\lambda} \leq \mathbb{E}_{q,\boldsymbol{\theta}_2}[f] \leq \lambda \mathbb{E}_{q,\boldsymbol{\theta}_1}[f] \tag{14}$$

Let $\boldsymbol{\theta}_0$ be a set of parameters such that we perform the following procedure in initializing Viterbi EM: first, we sample from the posterior distribution $p(z \mid \boldsymbol{\theta}_0, x)$, and then update the parameters with maximum likelihood estimate, in a regular M-step. Let λ be such that $p(z \mid \boldsymbol{\theta}_0, x)$ is within λ of $p(z \mid \boldsymbol{\theta}^*, x)$ (for all $x \in L(\boldsymbol{G})$). (Later we will show that UniformInit is a wise choice for making λ small. Note that UniformInit is equivalent to the procedure mentioned above with $\boldsymbol{\theta}_0 = \boldsymbol{\theta}_I$.)

[3] We thank an anonymous reviewer for pointing this out.

Consider $\tilde{p}_n(x)$, the empirical distribution over $x_1, \ldots, x_n$. As $n \to \infty$, we have that $\tilde{p}_n(x) \to p^*(x)$, almost surely, where p^* is:

$$p^*(x) = \sum_z p^*(x, z \mid \boldsymbol{\theta}^*) \tag{15}$$

This means that as $n \to \infty$ we have $\mathbb{E}_{\tilde{p}_n, \boldsymbol{\theta}}[f] \to \mathbb{E}_{p^*, \boldsymbol{\theta}}[f]$. Now, let $\boldsymbol{z}_0 = (z_{0,1}, \ldots, z_{0,n})$ be samples from $p(z \mid \boldsymbol{\theta}_0, x_i)$ for $i \in \{1, \ldots, n\}$. Then, from simple MLE computation, we know that the value

$$\begin{aligned} &\max_{\boldsymbol{\theta}'} \prod_{i=1}^{n} p(x_i, z_{0,i} \mid \boldsymbol{\theta}') \qquad (16) \\ &= \prod_{(A \to \alpha) \in \mathcal{R}} \left(\frac{F_{A \to \alpha}(\boldsymbol{z}_0)}{F_A(\boldsymbol{z}_0)} \right)^{F_{A \to \alpha}(\boldsymbol{z}_0)} \end{aligned}$$

We also know that for $\boldsymbol{\theta}_0$, from the consistency of MLE, for large enough samples:

$$\frac{F_{A \to \alpha}(\boldsymbol{z}_0)}{F_A(\boldsymbol{z}_0)} \approx \frac{\mathbb{E}_{\tilde{p}_n, \boldsymbol{\theta}_0}[f_{A \to \alpha}]}{\mathbb{E}_{\tilde{p}_n, \boldsymbol{\theta}_0}[f_A]} \tag{17}$$

which means that we have the following as n grows (starting from the ViterbiTrain objective with initial state $\boldsymbol{z} = \boldsymbol{z}_0$):

$$\begin{aligned} &\max_{\boldsymbol{\theta}'} \prod_{i=1}^{n} p(x_i, z_{0,i} \mid \boldsymbol{\theta}') \qquad (18) \\ &\overset{\text{(Eq. 16)}}{=} \prod_{(A \to \alpha) \in \mathcal{R}} \left(\frac{F_{A \to \alpha}(\boldsymbol{z}_0)}{F_A(\boldsymbol{z}_0)} \right)^{F_{A \to \alpha}(\boldsymbol{z}_0)} \qquad (19) \\ &\overset{\text{(Eq. 17)}}{\approx} \prod_{(A \to \alpha) \in \mathcal{R}} \left(\frac{\mathbb{E}_{\tilde{p}_n, \boldsymbol{\theta}_0}[f_{A \to \alpha}]}{\mathbb{E}_{\tilde{p}_n, \boldsymbol{\theta}_0}[f_A]} \right)^{F_{A \to \alpha}(\boldsymbol{z}_0)} \qquad (20) \end{aligned}$$

We next use the fact that $\tilde{p}_n(x) \approx p^*(x)$ for large n, and apply Eq. 14, noting again our assumption that $p(z \mid \boldsymbol{\theta}_0, x)$ is within λ of $p(z \mid \boldsymbol{\theta}^*, x)$. We also let $B = \sum_i |z_i|$, where $|z_i|$ is the number of nodes in the derivation z_i. Note that $F_A(z_i) \leq B$. The above quantity (Eq. 20) is approximately bounded above by

$$\begin{aligned} &\prod_{(A \to \alpha) \in \mathcal{R}} \frac{1}{\lambda^{2B}} \left(\frac{\mathbb{E}_{p^*, \boldsymbol{\theta}^*}[f_{A \to \alpha}]}{\mathbb{E}_{p^*, \boldsymbol{\theta}^*}[f_A]} \right)^{F_{A \to \alpha}(\boldsymbol{z}_0)} \qquad (21) \\ &= \frac{1}{\lambda^{2|\mathcal{R}|B}} \prod_{(A \to \alpha) \in \mathcal{R}} (\theta^*_{A \to \alpha})^{F_{A \to \alpha}(\boldsymbol{z}_0)} \qquad (22) \end{aligned}$$

Eq. 22 follows from:

$$\theta^*_{A \to \alpha} = \frac{\mathbb{E}_{p^*, \boldsymbol{\theta}^*}[f_{A \to \alpha}]}{\mathbb{E}_{p^*, \boldsymbol{\theta}^*}[f_A]} \tag{23}$$

If we continue to develop Eq. 22 and apply Eq. 17 and Eq. 23 again, we get that:

$$\begin{aligned} &\frac{1}{\lambda^{2|\mathcal{R}|B}} \prod_{(A \to \alpha) \in \mathcal{R}} (\theta^*_{A \to \alpha})^{F_{A \to \alpha}(\boldsymbol{z}_0)} \\ &= \frac{1}{\lambda^{2|\mathcal{R}|B}} \prod_{(A \to \alpha) \in \mathcal{R}} (\theta^*_{A \to \alpha})^{F_{A \to \alpha}(\boldsymbol{z}_0) \cdot \frac{F_A(\boldsymbol{z}_0)}{F_A(\boldsymbol{z}_0)}} \\ &\approx \frac{1}{\lambda^{2|\mathcal{R}|B}} \prod_{(A \to \alpha) \in \mathcal{R}} (\theta^*_{A \to \alpha})^{\frac{\mathbb{E}_{p^*, \boldsymbol{\theta}_0}[f_{A \to \alpha}]}{\mathbb{E}_{p^*, \boldsymbol{\theta}_0}[f_A]} \cdot F_A(\boldsymbol{z}_0)} \\ &\geq \frac{1}{\lambda^{2|\mathcal{R}|B}} \prod_{(A \to \alpha) \in \mathcal{R}} (\theta^*_{A \to \alpha})^{\lambda^2 \theta^*_{A \to \alpha} F_A(\boldsymbol{z}_0)} \\ &\geq \frac{1}{\lambda^{2|\mathcal{R}|B}} \underbrace{\left(\prod_{(A \to \alpha) \in \mathcal{R}} (\theta^*_{A \to \alpha})^{n \theta^*_{A \to \alpha}} \right)}_{T(\boldsymbol{\theta}^*, n)}^{B\lambda^2/n} \qquad (24) \\ &= \left(\frac{1}{\lambda^{2|\mathcal{R}|B}} \right) T(\boldsymbol{\theta}^*, n)^{B\lambda^2/n} \qquad (25) \\ &\triangleq d(\lambda; \boldsymbol{\theta}^*, |\mathcal{R}|, B) \qquad (26) \end{aligned}$$

where Eq. 24 is the result of $F_A(\boldsymbol{z}_0) \leq B$.

For two series $\{a_n\}$ and $\{b_n\}$, let "$a_n \gtrapprox b_n$" denote that $\lim_{n \to \infty} a_n \geq \lim_{n \to \infty} b_n$. In other words, a_n is asymptotically larger than b_n. Then, if we changed the representation of the objective function of the ViterbiTrain problem to *log*-likelihood, for $\boldsymbol{\theta}'$ that maximizes Eq. 18 (with some simple algebra) we have:

$$\begin{aligned} &\frac{1}{n} \sum_{i=1}^{n} \log_2 p(x_i, z_{0,i} \mid \boldsymbol{\theta}') \qquad (27) \\ &\gtrapprox -\frac{2|\mathcal{R}|B}{n} \log_2 \lambda + \frac{B\lambda^2}{n} \left(\frac{1}{n} \log_2 T(\boldsymbol{\theta}^*, n) \right) \\ &= -\frac{2|\mathcal{R}|B}{n} \log_2 \lambda - |\mathcal{N}| \frac{B\lambda^2}{|\mathcal{N}|n} \sum_{A \in \mathcal{N}} H(\boldsymbol{\theta}^*, A) \qquad (28) \end{aligned}$$

where

$$H(\boldsymbol{\theta}^*, A) = -\sum_{(A \to \alpha) \in R(A)} \theta^*_{A \to \alpha} \log_2 \theta^*_{A \to \alpha} \tag{29}$$

is the entropy of the multinomial for nonterminal A. $H(\boldsymbol{\theta}^*, A)$ can be thought of as the minimal number of bits required to encode a choice of a rule from A, if chosen independently from the other rules. All together, the quantity $\frac{B}{|\mathcal{N}|n} \left(\sum_{A \in \mathcal{N}} H(\boldsymbol{\theta}^*, A) \right)$ is the average number of bits required to encode a tree in our sample using

$\boldsymbol{\theta}^*$, while removing dependence among all rules and assuming that each node at the tree is chosen uniformly.[4] This means that the log-likelihood, for large n, is bounded from above by a linear function of the (average) number of bits required to optimally encode n trees of total size B, while assuming independence among the rules in a tree. We note that the quantity B/n will tend toward the average size of a tree, which, under Condition 1, must be finite.

Our final approximate bound from Eq. 28 relates the choice of distribution, from which sample $\boldsymbol{z}_0$, to λ. The lower bound in Eq. 28 is a monotone-decreasing function of λ. We seek to make λ as small as possible to make the bound tight. We next show that the uniform distribution optimizes λ in that sense.

7.2 Optimizing λ

Note that the optimal choice of λ, for a single x and for candidate initializer $\boldsymbol{\theta}'$, is

$$\lambda_{\mathrm{opt}}(x, \boldsymbol{\theta}^*; \boldsymbol{\theta}_0) \quad = \quad \sup_{z \in D(\boldsymbol{G}, x)} \frac{p(z \mid \boldsymbol{\theta}_0, x)}{p(z \mid \boldsymbol{\theta}^*, x)} \tag{30}$$

In order to avoid degenerate cases, we will add another condition on the true model, $\boldsymbol{\theta}^*$:

Condition 2. *There exists $\tau > 0$ such that, for any $x \in L(\boldsymbol{G})$ and for any $z \in D(\boldsymbol{G}, x)$, $p(z \mid \boldsymbol{\theta}^*, x) \geq \tau$.*

This is a strong condition, forcing the cardinality of $D(\boldsymbol{G})$ to be finite, but it is not unreasonable if natural language sentences are effectively bounded in length.

Without further information about $\boldsymbol{\theta}^*$ (other than that it satisfies Condition 2), we may want to consider the worst-case scenario of possible λ, hence we seek initializer $\boldsymbol{\theta}_0$ such that

$$\Lambda(x; \boldsymbol{\theta}_0) \quad \triangleq \quad \sup_{\boldsymbol{\theta}} \lambda_{\mathrm{opt}}(x, \boldsymbol{\theta}; \boldsymbol{\theta}_0) \tag{31}$$

is minimized. If $\boldsymbol{\theta}_0 = \boldsymbol{\theta}_I$, then we have that $p(z \mid \boldsymbol{\theta}_I, x) = |D(\boldsymbol{G}, x)|^{-1} \triangleq \mu_x$. Together with Condition 2, this implies that

$$\frac{p(z \mid \boldsymbol{\theta}_I, x)}{p(z \mid \boldsymbol{\theta}^*, x)} \quad \leq \quad \frac{\mu_x}{\tau} \tag{32}$$

and hence $\lambda_{\mathrm{opt}}(x, \boldsymbol{\theta}^*) \leq \mu_x/\tau$ for any $\boldsymbol{\theta}^*$, hence $\Lambda(x; \boldsymbol{\theta}_I) \leq \mu_x/\tau$. However, if we choose $\boldsymbol{\theta}_0 \neq \boldsymbol{\theta}_I$, we have that $p(z' \mid \boldsymbol{\theta}_0, x) > \mu_x$ for some z', hence, for $\boldsymbol{\theta}^*$ such that it assigns probability τ on z', we have that

$$\sup_{z \in D(\boldsymbol{G}, x)} \frac{p(z \mid \boldsymbol{\theta}_0, x)}{p(z \mid \boldsymbol{\theta}^*, x)} \quad > \quad \frac{\mu_x}{\tau} \tag{33}$$

and hence $\lambda_{\mathrm{opt}}(x, \boldsymbol{\theta}^*; \boldsymbol{\theta}') > \mu_x/\tau$, so $\Lambda(x; \boldsymbol{\theta}') > \mu_x/\tau$. So, to optimize for the worst-case scenario over true distributions with respect to λ, we are motivated to choose $\boldsymbol{\theta}_0 = \boldsymbol{\theta}_I$ as defined in Condition 1. Indeed, UniformInit uses $\boldsymbol{\theta}_I$ to initialize the state of Viterbi EM.

We note that if $\boldsymbol{\theta}_I$ was known for a specific grammar, then we could have used it as a direct initializer. However, Condition 1 only guarantees its existence, and does not give a practical way to identify it. In general, as mentioned above, $\boldsymbol{\theta} = \mathbf{1}$ can be used to obtain a weighted CFG that satisfies $p(z \mid \boldsymbol{\theta}, x) = 1/|D(\boldsymbol{G}, x)|$. Since we require a uniform posterior distribution, the number of derivations of a fixed length is finite. This means that we can converted the weighted CFG with $\boldsymbol{\theta} = \mathbf{1}$ to a PCFG with the same posterior (Smith and Johnson, 2007), and identify the appropriate $\boldsymbol{\theta}_I$.

8 Related Work

Viterbi training is closely related to the k-means clustering problem, where the objective is to find k centroids for a given set of d-dimensional points such that the sum of distances between the points and the closest centroid is minimized. The analog for Viterbi EM for the k-means problem is the k-means clustering algorithm (Lloyd, 1982), a coordinate ascent algorithm for solving the k-means problem. It works by iterating between an E-like-step, in which each point is assigned the closest centroid, and an M-like-step, in which the centroids are set to be the center of each cluster.

"k" in k-means corresponds, in a sense, to the size of our grammar. k-means has been shown to be NP-hard both when k varies and d is fixed and when d varies and k is fixed (Aloise et al., 2009; Mahajan et al., 2009). An open problem relating to our hardness result would be whether ViterbiTrain (or ConditionalViterbiTrain) is hard even if we do not permit grammars of arbitrarily large size, or at least, constrain the number of rules that do not rewrite to terminals (in our current reduction, the

[4] We note that Grenander (1967) describes a (linear) relationship between the *derivational entropy* and $H(\boldsymbol{\theta}^*, A)$. The derivational entropy is defined as $h(\boldsymbol{\theta}^*, A) = -\sum_{x,z} p(x, z \mid \boldsymbol{\theta}^*) \log p(x, z \mid \boldsymbol{\theta}^*)$, where z ranges over trees that have nonterminal A as the root. It follows immediately from Grenander's result that $\sum_A H(\boldsymbol{\theta}^*, A) \leq \sum_A h(\boldsymbol{\theta}^*, A)$.

size of the grammar grows as the size of the 3-SAT formula grows).

On a related note to §7, Arthur and Vassilvitskii (2007) described a greedy initialization algorithm for initializing the centroids of k-means, called k-means++. They show that their initialization is $O(\log k)$-competitive; i.e., it approximates the optimal clusters assignment by a factor of $O(\log k)$. In §7.1, we showed that uniform-at-random initialization is approximately $O(|\mathcal{N}|L\lambda^2/n)$-competitive (modulo an additive constant) for CNF grammars, where n is the number of sentences, L is the total length of sentences and λ is a measure for distance between the true distribution and the uniform distribution.[5]

Many combinatorial problems in NLP involving phrase-structure trees, alignments, and dependency graphs are hard (Sima'an, 1996; Goodman, 1998; Knight, 1999; Casacuberta and de la Higuera, 2000; Lyngsø and Pederson, 2002; Udupa and Maji, 2006; McDonald and Satta, 2007; DeNero and Klein, 2008, *inter alia*). Of special relevance to this paper is Abe and Warmuth (1992), who showed that the problem of finding maximum likelihood model of probabilistic automata is hard even for a single string and an automaton with two states. Understanding the complexity of NLP problems, we believe, is crucial as we seek effective practical approximations when necessary.

9 Conclusion

We described some properties of Viterbi training for probabilistic context-free grammars. We showed that Viterbi training is NP-hard and, in fact, NP-hard to approximate. We gave motivation for uniform-at-random initialization for derivations in the Viterbi EM algorithm.

Acknowledgments

We acknowledge helpful comments by the anonymous reviewers. This research was supported by NSF grant 0915187.

References

N. Abe and M. Warmuth. 1992. On the computational complexity of approximating distributions by probabilistic automata. *Machine Learning*, 9(2–3):205–260.

S. Abney. 2007. *Semisupervised Learning for Computational Linguistics*. CRC Press.

D. Aloise, A. Deshpande, P. Hansen, and P. Popat. 2009. NP-hardness of Euclidean sum-of-squares clustering. *Machine Learning*, 75(2):245–248.

D. Arthur and S. Vassilvitskii. 2007. k-means++: The advantages of careful seeding. In *Proc. of ACM-SIAM symposium on Discrete Algorithms*.

F. Casacuberta and C. de la Higuera. 2000. Computational complexity of problems on probabilistic grammars and transducers. In *Proc. of ICGI*.

E. Charniak. 1997. Statistical parsing with a context-free grammar and word statistics. In *Proc. of AAAI*.

S. B. Cohen and N. A. Smith. 2009. Shared logistic normal distributions for soft parameter tying in unsupervised grammar induction. In *Proc. of HLT-NAACL*.

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

W. H. E. Day. 1983. Computationally difficult parsimony problems in phylogenetic systematics. *Journal of Theoretical Biology*, 103.

J. DeNero and D. Klein. 2008. The complexity of phrase alignment problems. In *Proc. of ACL*.

Y. Freund, H. Seung, E. Shamir, and N. Tishby. 1997. Selective sampling using the query by committee algorithm. *Machine Learning*, 28(2–3):133–168.

S. Goldwater and M. Johnson. 2005. Bias in learning syllable structure. In *Proc. of CoNLL*.

J. Goodman. 1998. *Parsing Inside-Out*. Ph.D. thesis, Harvard University.

U. Grenander. 1967. Syntax-controlled probabilities. Technical report, Brown University, Division of Applied Mathematics.

M. Johnson, T. L. Griffiths, and S. Goldwater. 2006. Adaptor grammars: A framework for specifying compositional nonparameteric Bayesian models. In *Advances in NIPS*.

M. Johnson, T. L. Griffiths, and S. Goldwater. 2007. Bayesian inference for PCFGs via Markov chain Monte Carlo. In *Proc. of NAACL*.

D. Klein and C. Manning. 2001. Natural language grammar induction using a constituent-context model. In *Advances in NIPS*.

K. Knight. 1999. Decoding complexity in word-replacement translation models. *Computational Linguistics*, 25(4):607–615.

S. P. Lloyd. 1982. Least squares quantization in PCM. In *IEEE Transactions on Information Theory*.

R. B. Lyngsø and C. N. S. Pederson. 2002. The consensus string problem and the complexity of comparing hidden Markov models. *Journal of Computing and System Science*, 65(3):545–569.

M. Mahajan, P. Nimbhorkar, and K. Varadarajan. 2009. The planar k-means problem is NP-hard. In *Proc. of International Workshop on Algorithms and Computation*.

[5] Making the assumption that the grammar is in CNF permits us to use L instead of B, since there is a linear relationship between them in that case.

D. McClosky, E. Charniak, and M. Johnson. 2006a. Effective self-training for parsing. In *Proc. of HLT-NAACL*.

D. McClosky, E. Charniak, and M. Johnson. 2006b. Reranking and self-training for parser adaptation. In *Proc. of COLING-ACL*.

R. McDonald and G. Satta. 2007. On the complexity of non-projective data-driven dependency parsing. In *Proc. of IWPT*.

R. M. Neal and G. E. Hinton. 1998. A view of the EM algorithm that justifies incremental, sparse, and other variants. In *Learning and Graphical Models*, pages 355–368. Kluwer Academic Publishers.

K. Sima'an. 1996. Computational complexity of probabilistic disambiguation by means of tree-grammars. In *In Proc. of COLING*.

M. Sipser. 2006. *Introduction to the Theory of Computation, Second Edition*. Thomson Course Technology.

N. A. Smith and M. Johnson. 2007. Weighted and probabilistic context-free grammars are equally expressive. *Computational Linguistics*, 33(4):477–491.

V. I. Spitkovsky, H. Alshawi, D. Jurafsky, and C. D. Manning. 2010. Viterbi training improves unsupervised dependency parsing. In *Proc. of CoNLL*.

R. Udupa and K. Maji. 2006. Computational complexity of statistical machine translation. In *Proc. of EACL*.

M. Wang, N. A. Smith, and T. Mitamura. 2007. What is the Jeopardy model? a quasi-synchronous grammar for question answering. In *Proc. of EMNLP*.

C. Yejin and C. Cardie. 2007. Structured local training and biased potential functions for conditional random fields with application to coreference resolution. In *Proc. of HLT-NAACL*.