Unsupervised Learning on an Approximate Corpus*

Jason Smith and **Jason Eisner**
Center for Language and Speech Processing
Johns Hopkins University
3400 N. Charles St., Baltimore, MD 21218, USA
{jsmith,jason}@cs.jhu.edu

Unsupervised learning techniques can take advantage of large amounts of unannotated text, but the largest text corpus (the Web) is not easy to use in its full form. Instead, we have statistics about this corpus in the form of n-gram counts (Brants and Franz, 2006). While n-gram counts do not directly provide sentences, a distribution over sentences can be estimated from them in the same way that n-gram language models are estimated. We treat this distribution over sentences as an *approximate corpus* and show how unsupervised learning can be performed on such a corpus using variational inference. We compare hidden Markov model (HMM) training on exact and approximate corpora of various sizes, measuring speed and accuracy on unsupervised part-of-speech tagging.

1 Introduction

We consider the problem of training generative models on very large datasets in sublinear time. It is well known how to train an HMM to maximize the likelihood of a corpus of sentences. Here we show how to train faster on a distribution over sentences that compactly approximates the corpus. The distribution is given by an 5-gram backoff language model that has been estimated from statistics of the corpus.

In this paper, we demonstrate our approach on a traditional testbed for new structured-prediction learning algorithms, namely HMMs. We focus on unsupervised learning. This serves to elucidate the structure of our variational training approach, which stitches overlapping n-grams together rather than treating them in isolation. It also confirms that at least in this case, accuracy is not harmed by the key approximations made by our method. In future, we hope to scale up to the Google n-gram corpus (Brants and Franz, 2006) and learn a more detailed, explanatory joint model of tags, syntactic dependencies, and topics. Our intuition here is that web-scale data may be needed to learn the large number of lexically and contextually specific parameters.

*Work was supported in part by NSF grant No. 0347822.

1.1 Formulation

Let $\boldsymbol{w}$ ("words") denote an observation sequence, and let $\boldsymbol{t}$ ("tags") denote a hidden HMM state sequence that may explain $\boldsymbol{w}$. This terminology is taken from the literature on inducing part-of-speech (POS) taggers using a first-order HMM (Merialdo, 1994), which we use as our experimental setting.

Maximum a posteriori (MAP) training of an HMM p_θ seeks parameters θ to maximize

$$N \cdot \sum_{\boldsymbol{w}} c(\boldsymbol{w}) \log \sum_{\boldsymbol{t}} p_\theta(\boldsymbol{w}, \boldsymbol{t}) + \log \Pr_{\text{prior}}(\theta) \quad (1)$$

where c is an empirical distribution that assigns probability $1/N$ to each of the N sentences in a training corpus. Our technical challenge is to **generalize this MAP criterion** to other, structured distributions c that compactly approximate the corpus.

Specifically, we address the case where c is given by any **probabilistic FSA**, such as a **backoff language model**—that is, a variable-order Markov model estimated from corpus statistics. Similar sentences $\boldsymbol{w}$ share subpaths in the FSA and cannot easily be disentangled. The support of c is typically infinite (for a cyclic FSA) or at least exponential. Hence it is no longer practical to compute the tagging distribution $p(\boldsymbol{t} \mid \boldsymbol{w})$ for each sentence $\boldsymbol{w}$ separately, as in traditional MAP-EM or gradient ascent approaches. We will maximize our exact objective, or a cheaper variational approximation to it, in a way that crucially allows us to retain the structure-sharing.

1.2 Motivations

Why train from a distribution rather than a corpus? First, the foundation of statistical NLP is distributions over strings that are specified by weighted automata and grammars. We regard parameter estimation from such a distribution c (rather than from a sample) as a **natural question**. Previous work on modeling c with a distribution from another family was motivated by *approximating a grammar or*

2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 131–141,
Montréal, Canada, June 3-8, 2012.

model rather than *generalizing from a dataset*, and hence removed latent variables while adding parameters (Nederhof, 2000; Mohri and Nederhof, 2001; Liang et al., 2008), whereas we do the reverse.

Second, in practice, one may want to incorporate massive amounts of (possibly out-of-domain) data in order to get **better coverage** of phenomena. Massive datasets usually require a simple model (given a time budget). We propose that it may be possible to use a lot of data *and* a good model by reducing the accuracy of the data representation instead. While training will become more complicated, it can still result in an overall speedup, because a frequent 5-gram collapses into a single parameter of the estimated distribution that only needs to be processed once per training iteration. By pruning low-count n-grams or reducing the maximum n below 5, one can further increase data volume for the fixed time budget at the expense of approximation quality.

Third, one may not have **access** to the original corpus. If one lacks the resources to harvest the web, the Google n-gram corpus was derived from over a trillion words of English web text. Privacy or copyright issues may prevent access, but one may still be able to work with n-gram statistics: Michel et al. (2010) used such statistics from 5 million scanned books. Several systems use n-gram counts (Bergsma et al., 2009; Lin et al., 2009) or other web statistics (Lapata and Keller, 2005) as features within a classifier. A large language model from n-gram counts yields an effective prior over hypotheses in tasks like machine translation (Brants et al., 2007). We similarly construct an n-gram model, but treat it as the primary *training* data whose structure is to be explained by the generative HMM. Thus our criterion does not explain the n-grams in isolation, but rather tries to explain the likely full sentences $\boldsymbol{w}$ that the model reconstructed from overlapping n-grams. This is something like shotgun sequencing, in which likely DNA strings are reconstructed from overlapping short reads (Staden, 1979); however, we train an HMM on the resulting distribution rather than merely trying to find its mode.

Finally, unsupervised HMM training discovers latent structure by approximating an empirical distribution c (the corpus) with a latent-variable distribution p (the trained HMM) that has fewer parameters. We show how to do the same where the distribution c is not a corpus but a finite-state distribution. In general, this finite-state c could represent some sophisticated estimate of the ***population*** **distribution**, using shrinkage, word classes, neural-net predictors, etc. to generalize in some way beyond the training sample *before* fitting p. For the sake of speed and clear comparison, however, our present experiments take c to be a compact approximation to the *sample* distribution, requiring only n-grams.

Spectral learning of HMMs (Hsu et al., 2009) also learns from a collection of n-grams. It has the striking advantage of converging globally to the true HMM parameters (under a certain reparameterization), with enough data and under certain assumptions. However, it does not exploit context beyond a trigram (it will not maximize, even locally, the likelihood of a finite sample of sentences), and cannot exploit priors or structure—e.g., that the emissions are consistent with a tag dictionary or that the transitions encode a higher-order or factorial HMM. Our more general technique extends to other latent-variable models, although it suffers from variational EM's usual local optima and approximation errors.

2 A variational lower bound

Our starting point is the variational EM algorithm (Jordan et al., 1999). Recall that this maximizes a lower bound on the MAP criterion of equation 1, by bounding the log-likelihood subterm as follows:

$$\begin{aligned} &\log \textstyle\sum_{\boldsymbol{t}} p_\theta(\boldsymbol{w}, \boldsymbol{t}) && (2)\\ &= \log \textstyle\sum_{\boldsymbol{t}} q(\boldsymbol{t})(p_\theta(\boldsymbol{w}, \boldsymbol{t})/q(\boldsymbol{t})) \\ &\geq \textstyle\sum_{\boldsymbol{t}} q(\boldsymbol{t}) \log(p_\theta(\boldsymbol{w}, \boldsymbol{t})/q(\boldsymbol{t})) \\ &= \mathbb{E}_{q(\boldsymbol{t})}[\log p_\theta(\boldsymbol{w}, \boldsymbol{t}) - \log q(\boldsymbol{t})] && (3) \end{aligned}$$

This use of Jensen's inequality is valid for any distribution q. As Neal and Hinton (1998) show, the EM algorithm (Dempster et al., 1977) can be regarded as locally maximizing the resulting lower bound by alternating optimization, where q is a free parameter. The E-step optimizes q for fixed θ, and the M-step optimizes θ for fixed q. These computations are tractable for HMMs, since the distribution $q(\boldsymbol{t}) = p_\theta(\boldsymbol{t} \mid \boldsymbol{w})$ that is optimal at the E-step (which makes the inequality tight) can be represented as a lattice (a certain kind of weighted DFA), and this makes the M-step tractable via the forward-backward algorithm. However, there are many extensions such as

factorial HMMs and Bayesian HMMs in which an expectation under $p_\theta(\boldsymbol{t} \mid \boldsymbol{w})$ involves an intractable sum. In this setting, one may use variational EM, in which q is restricted to some parametric family q_ϕ that will permit a tractable M-step. In this case the E-step chooses the optimal values of the *variational parameters* ϕ; the inequality is no longer tight.

There are two equivalent views of how this procedure is applied to a training *corpus*. One view is that the corpus log-likelihood is just as in (2), where $\boldsymbol{w}$ is taken to be the concatenation of all training sentences. The other view is that the corpus log-likelihood is a sum over many terms of the form (2), one for each training sentence $\boldsymbol{w}$, and we bound each summand individually using a different q_ϕ.

However, neither view leads to a practical implementation in our setting. We can neither concatenate all the relevant $\boldsymbol{w}$ nor loop over them, since we want the expectation of (2) under some distribution $c(\boldsymbol{w})$ such that $\{\boldsymbol{w} : c(\boldsymbol{w}) > 0\}$ is very large or infinite. Our move is to make q be a *conditional* distribution $q(\boldsymbol{t} \mid \boldsymbol{w})$ that applies to all $\boldsymbol{w}$ at once. The following holds by applying Jensen's inequality separately to each $\boldsymbol{w}$ in the expectation (this is valid since for each $\boldsymbol{w}$, $q(\boldsymbol{t} \mid \boldsymbol{w})$ is a distribution):

$$\begin{aligned}
&\mathbb{E}_{c(\boldsymbol{w})} \log \textstyle\sum_{\boldsymbol{t}} p_\theta(\boldsymbol{w}, \boldsymbol{t}) && (4)\\
&= \mathbb{E}_{c(\boldsymbol{w})} \log \textstyle\sum_{\boldsymbol{t}} q(\boldsymbol{t} \mid \boldsymbol{w})(p_\theta(\boldsymbol{w}, \boldsymbol{t})/q(\boldsymbol{t} \mid \boldsymbol{w}))\\
&\geq \mathbb{E}_{c(\boldsymbol{w})} \textstyle\sum_{\boldsymbol{t}} q(\boldsymbol{t} \mid \boldsymbol{w}) \log(p_\theta(\boldsymbol{w}, \boldsymbol{t})/q(\boldsymbol{t} \mid \boldsymbol{w}))\\
&= \mathbb{E}_{cq(\boldsymbol{w},\boldsymbol{t})}[\log p_\theta(\boldsymbol{w}, \boldsymbol{t}) - \log q(\boldsymbol{t} \mid \boldsymbol{w})] && (5)
\end{aligned}$$

where we use $cq(\boldsymbol{w}, \boldsymbol{t})$ to denote the joint distribution $c(\boldsymbol{w}) \cdot q(\boldsymbol{t} \mid \boldsymbol{w})$. Thus, just as c is our approximate corpus, cq is our approximate *tagged* corpus. Our variational parameters ϕ will be used to parameterize cq directly. To ensure that cq_ϕ can indeed be expressed as $c(\boldsymbol{w}) \cdot q(\boldsymbol{t} \mid \boldsymbol{w})$, making the above bound valid, it suffices to guarantee that our variational family preserves the marginals:

$$(\forall \boldsymbol{w}) \textstyle\sum_{\boldsymbol{t}} cq_\phi(\boldsymbol{w}, \boldsymbol{t}) = c(\boldsymbol{w})$$

3 Finite-state encodings and algorithms

In the following, we will show how to maximize (5) for particular families of p, c, and cq that can be expressed using finite-state machines (FSMs)—that is, finite-state acceptors (FSAs) and transducers (FSTs). This general presentation of our method enables variations using other FSMs.

A path in an FSA accepts a string. In an FST, each arc is labeled with a "word : tag" pair, so that a path accepts a string *pair* $(\boldsymbol{w}, \boldsymbol{t})$ obtained by respectively concatenating the words and the tags encountered along the path. Our FSMs are weighted in the $(+, \times)$ semiring: the weight of any path is the product $(\times)$ of its arc weights, while the weight assigned to a string or string pair is the total weight $(+)$ of all its accepting paths. An FSM is *unambiguous* if each string or string pair has at most one accepting path.

Figure 1 reviews how to represent an HMM POS tagger as an FST (b), and how composing this with an FSA that accepts a single sentence gives us the familiar HMM tagging lattice as an FST (c). The forward-backward algorithm sums over paths in the lattice via dynamic programming (Rabiner, 1989).

In section 3.1, we replace the straight-line FSA of Figure 1a with an FSA that defines a more general distribution $c(\boldsymbol{w})$ over many sentences. Note that we cannot simply use this as a drop-in replacement in the construction of Figure 1. That would correspond to running EM on a single but uncertain sentence (distributed as $c(\boldsymbol{w})$) rather than a collection of observed sentences. For example, in the case of an ordinary training corpus of N sentences, the new FSA would be a parallel *union* (sum) of N straight-line paths—rather than a serial *concatenation* (product) of those paths as in ordinary EM (see above). Running the forward algorithm on the resulting lattice would compute $\mathbb{E}_{c(\boldsymbol{w})} \sum_{\boldsymbol{t}} p(\boldsymbol{w}, \boldsymbol{t})$, whose log is $\log \mathbb{E}_{c(\boldsymbol{w})} \sum_{\boldsymbol{t}} p(\boldsymbol{w}, \boldsymbol{t})$ rather than our desired $\mathbb{E}_{c(\boldsymbol{w})} \log \sum_{\boldsymbol{t}} p(\boldsymbol{w}, \boldsymbol{t})$. Instead, we use c in section 3.2 to construct a variational family cq_ϕ. We then show in sections 3.3–3.5 how to compute and locally maximize the variational lower bound (5).

3.1 Modeling a corpus with n-gram counts

n-gram backoff language models have been used for decades in automatic speech recognition and statistical machine translation. We follow the usual FSA construction (Allauzen et al., 2003). The state of a 5-gram FSA model $c(\boldsymbol{w})$ must remember the previous 4-gram. For example, it would include an arc from state defg (the previous 4-gram) to state efgh with label h and weight $c(\mathsf{h} \mid \mathsf{defg})$. Then, with appropriate handling of boundary conditions, a sentence $\boldsymbol{w} = \ldots \mathsf{defghi} \ldots$ is accepted along a single path of weight $c(\boldsymbol{w}) = \cdots c(\mathsf{h} \mid \mathsf{defg}) \cdot c(\mathsf{i} \mid \mathsf{efgh}) \cdots$. Arcs

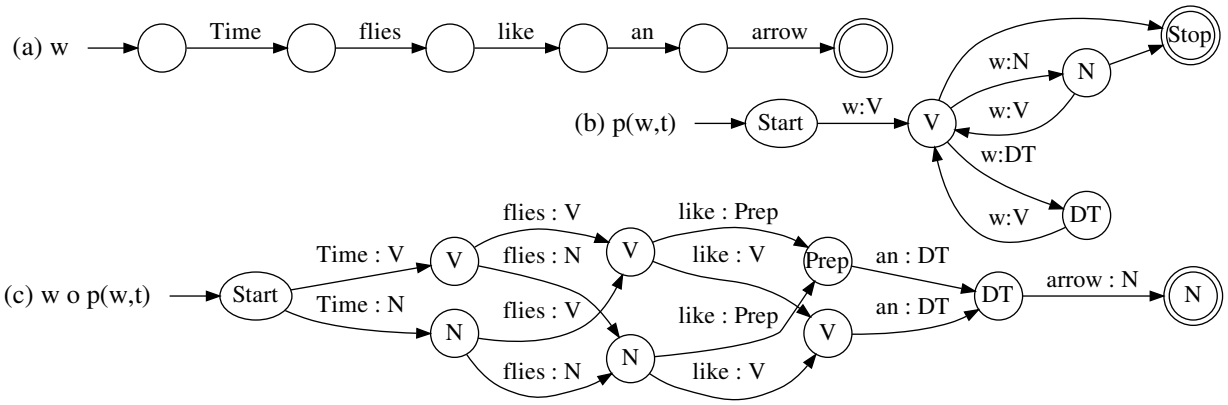

Figure 1: Ordinary HMM tagging with finite-state machines. An arc's label may have up to three components: "word:tag / weight." (Weights are suppressed for space. State labels are not part of the machine but suggest the history recorded by each state.) (a) $\boldsymbol{w}$ is an FSA that generates the sentence "Time flies like an arrow"; all arcs have probability 1. (b) $p(\boldsymbol{w}, \boldsymbol{t})$ is an FST representing an HMM (many arcs are not shown and words are abbreviated as "w"). Each arc $w : t$ is weighted by the product of transition and emission probabilities, $p(t \mid \text{previous } t) \cdot p(w \mid t)$. Composing (a) with (b) yields (c), an FST that encodes the joint probabilities $p(\boldsymbol{w}, \boldsymbol{t})$ of all possible taggings of the sentence $\boldsymbol{w}$.

of weight 0 can be omitted from the FSA.[1]

To estimate a conditional probability like $c(\mathsf{h} \mid \mathsf{defg})$ above, we simply take an unsmoothed ratio of two n-gram counts. This ML estimation means that c will approximate as closely as possible the training sample from which the counts were drawn. That gives a fair comparison with ordinary EM, which trains directly on that sample. (See discussion at the end of section 1.2 for alternatives.)

Yet we decline to construct a full 5-gram model, which would not be as compact as desired. A collection of all web 5-grams would be nearly as large as the web itself (by Zipf's Law). We may not have such a collection. For example, the Google n-gram corpus version 2 contains counts only for 1-grams that appear at least 40 times and 2-, 3-, 4-, and 5-grams that appear at least 10 times (Lin et al., 2009).

Instead, we construct a *backoff* language model. This FSA has one arc for each n-gram *in the collection*. Our algorithm's runtime (per iteration) will be linear in the number of arcs. If the 5-gram defgh is not in our collection, then there can be no h arc leaving defg. When encountering h in state defg, the automaton will instead take a *failure arc* (Allauzen et al., 2003) to the "backoff state" efg. It may be able to consume the h from that state, on an arc with weight $c(\mathsf{h} \mid \mathsf{efg})$; or it may have to back off further to fg. Each state's failure arc is weighted such that the state's outgoing arcs sum to 1. It is labeled with the special symbol Φ, which does not contribute to the word string accepted along a path.

We take care never to allow backoff to the empty state ϵ,[2] since we find that $c(\boldsymbol{w})$ is otherwise too coarse an approximation to English: sampled sentences tend to be disjointed, with some words generated in complete ignorance of their left context.

3.2 The variational distribution $cq(\boldsymbol{w}, \boldsymbol{t})$

The "variational gap" between (4) and (5) is $\mathbb{E}_{c(\boldsymbol{w})}\mathrm{KL}(q(\boldsymbol{t} \mid \boldsymbol{w}) \mid\mid p_\theta(\boldsymbol{t} \mid \boldsymbol{w}))$. That is, the bound is good if q does a good job of approximating p_θ's tagging distribution on a randomly drawn sentence.

Note that $n-1$ is the order of our n-gram Markov

[1] The FSA's initial state is the unigram history #, and its final states (which have no outgoing arcs) are the other states whose n-gram labels end in #. Here # is a boundary symbol that falls between sentences. To compute the weighted transitions, sentence boundaries must be manually or automatically annotated, either on the training corpus as in our present experiments, or directly on the training n-grams if we have only those.

To automatically find boundaries in an n-gram collection, one could apply a local classifier to each n-gram. But in principle, one could exploit more context and get a globally consistent annotation by stitching the n-grams together and applying the methods of this paper—replacing p_θ with an existing CRF sentence boundary detector, replacing c with a document-level (not sentence-level) language model, and optimizing cq_ϕ to be a version of c that is probabilistically annotated with sentence boundaries, which yields our desired distribution over sentences.

[2] To prevent such backoff, it suffices to include all 2-grams with count > 0. But where the full collection of 2-grams is unavailable or too large, one can remove the empty state (and recursively remove all states that transition only to removed states), and then renormalize the model locally or globally.

model $c(\boldsymbol{w})$ (i.e., each word is chosen given the previous $n-1$ words). Let $n_p - 1$ be the order of the HMM $p_\theta(\boldsymbol{w}, \boldsymbol{t})$ that we are training: i.e., each tag is chosen given the previous $n_p - 1$ tags. Our experiments take $n_p = 2$ (a bigram HMM) as in Figure 1.

We will take $q_\phi(\boldsymbol{t} \mid \boldsymbol{w})$ to be a *conditional Markov model* of order $n_q - 1$.[3] It will predict the tag at position i using a multinomial conditioned on the preceding $n_q - 1$ tags and on the word n-gram ending at position i (where n is as large as possible such that this n-gram is in our training collection). ϕ is the collection of all multinomial parameters.

If $n_q = n_p$, then our variational gap can be made 0 as in ordinary non-variational EM (see section 3.5). In our experiments, however, we save memory by choosing $n_q = 1$. Thus, our variational gap is tight to the extent that a word's POS tag under the model p_θ is conditionally independent of previous tags and the rest of the sentence, *given an n-word window*.[4] This is the assumption made by local classification models (Punyakanok et al., 2005; Toutanova and Johnson, 2007). Note that it is milder than the "one tagging per n-gram" hypothesis (Dawborn and Curran, 2009; Lin et al., 2009), which claims that each 5-gram (and therefore each sentence!) is unambiguous as to its full tagging. In contrast, we allow that a tag may be ambiguous even given an n-word window; we merely suppose that there is no *further* disambiguating information accessible to p_θ.[5]

We can encode the resulting $cq(\boldsymbol{w}, \boldsymbol{t})$ as an FST. With $n_q = 1$, the *states* of cq are isomorphic to the states of c. However, an *arc* in c from defg with label h and weight 0.2 is replaced in cq by several arcs—one per tag t—with label h : t and weight $0.2 \cdot q_\phi(t \mid \text{defgh})$.[6] We remark that an encoding of $q(\boldsymbol{t} \mid \boldsymbol{w})$ as an FST would be identical except for dropping the c factor (e.g., 0.2) from each weight. Composing $c \circ q$ would then recover cq.

This construction associates one variational parameter in ϕ with each arc in cq—that is, with each pair (arc in c, tag t), if $n_q = 1$. There would be little point in sharing these parameters across arcs of cq, as that would reduce the expressiveness of the variational distribution without reducing runtime.[7]

Notice that maximizing equation (5) jointly learns not only a compact slow HMM tagger p_θ, but also a large fast tagger q_ϕ that simply memorizes the likely tags in each n-gram context. This is reminiscent of structure compilation (Liang et al., 2008).

3.3 Computing the variational objective

The expectation in equation (5) can now be computed efficiently and elegantly by dynamic programming over the FSMs, for a given θ and ϕ.

We exploit our representation of cq_ϕ as an FSM over the $(+, \times)$ semiring. The path weights represent a probability distribution over the paths. In general, it is efficient to compute the expected **value** of a random FSM path, for any definition of value that decomposes additively over the path's arcs. The approach is to apply the forward algorithm to a version of cq_ϕ where we now regard each arc as weighted by an *ordered pair* of real numbers. The $(+, \times)$ operations for combining weights (section 3) are replaced with the operations of an "expectation semiring" whose elements are such pairs (Eisner, 2002).

Suppose we want to find $\mathbb{E}_{cq_\phi(\boldsymbol{w},\boldsymbol{t})} \log q_\phi(\boldsymbol{t} \mid \boldsymbol{w})$. To reduce this to an expected value problem, we must assign a *value* to each arc of cq_ϕ such that the

[3] A conditional Markov model is a simple case of a maximum-entropy Markov model (McCallum et al., 2000).

[4] At present, the word being tagged is the *last* word in the window. We do have an efficient modification in which the window is *centered* on the word, by using an FST cq that delays the emission of a tag until up to 2 subsequent words have been seen.

[5] With difficulty, one can construct English examples that violate our assumption. (1) "Some monitor lizards from Africa ..." versus "Some monitor lizards from a distance ...": there are words far away from "monitor" that help disambiguate whether "monitor" is a noun or a verb. ("Monitor lizards" are a species, but some people like to monitor lizards.) (2) "Time flies": "flies" is more likely to be a noun if "time" is a verb.

[6] In the case $n_q > 1$, the states of c would need to be split in order to remember $n_q - 1$ tags of history. For example, if c is Figure 1a, splitting its states with $n_q = 2$ would yield a cq with a topology like Figure 1c, but with each arc having an independent variational parameter.

[7] One could increase the number of arcs and hence variational parameters by splitting the states of cq to remember more history. In particular, one could increase the width n_q of the tag window, or one could increase the width of the word window by splitting states of c (without changing the distribution $c(\boldsymbol{w})$).

Conversely, one could reduce the number of variational parameters by further restricting the variational family. For example, requiring $q(\boldsymbol{t} \mid \boldsymbol{w})$ to have entropy 0 (analogous to "hard EM" or "Viterbi EM") would associate a single deterministic tag with each arc of c. This is fast, makes cq as compact as c, and is still milder than "one tagging per n-gram." More generously, one could allow up to 2 tags per arc of c, or use a low-dimensional representation of the arc's distribution over tags.

total value of a path accepting $(\boldsymbol{w}, \boldsymbol{t})$ is $\log q_\phi(\boldsymbol{t} \mid \boldsymbol{w})$. Thus, let the value of each arc in cq_ϕ be the log of its weight in the isomorphic FST $q_\phi(\boldsymbol{t} \mid \boldsymbol{w})$.[8]

We introduce some notation to make this precise. A state of cq_ϕ is a pair of the form $[h_c, h_q]$, where h_c is a state of c (e.g., an $(n-1)$-word history) and h_q is an $(n_q - 1)$-tag history. We saw in the previous section that an arc a leaving this state, and labeled with $w : t$ where w is a word and t is a tag, will have a weight of the form $k_a \stackrel{\text{def}}{=} c(w \mid h_c)\phi_a$ where $\phi_a \stackrel{\text{def}}{=} q_\phi(t \mid h_c w, h_q)$. We now let the value $v_a \stackrel{\text{def}}{=} \log \phi_a$.[9] Then, just as the weight of a path accepting $(\boldsymbol{w}, \boldsymbol{t})$ is $\prod_a k_a = cq_\phi(\boldsymbol{w}, \boldsymbol{t})$, the value of that path is $\sum_a v_a = \log q_\phi(\boldsymbol{t} \mid \boldsymbol{w})$, as desired.

To compute the *expected* value $\bar{r}$ over all paths, we follow a generalized forward-backward recipe (Li and Eisner, 2009, section 4.2). First, run the forward and backward algorithms over cq_ϕ.[10] Now the expected value is a sum over all arcs of cq_ϕ, namely $\bar{r} = \sum_a \alpha_a k_a v_a \beta_a$, where α_a denotes the forward probability of arc a's *source* state and β_a denotes the backward probability of arc a's *target* state.

Now, in fact, the expectation we need to compute is not $\mathbb{E}_{cq_\phi(\boldsymbol{w},\boldsymbol{t})} \log q_\phi(\boldsymbol{t} \mid \boldsymbol{w})$ but rather equation (5). So the value v_a of arc a should not actually be $\log \phi_a$ but rather $\log \theta_a - \log \phi_a$ where $\theta_a \stackrel{\text{def}}{=} p_\theta(t \mid h_p) \cdot p_\theta(w \mid t)$. This is a minor change—except that v_a now depends on h_p, which is the history of $n_p - 1$ previous tags. If $n_p > n_q$, then a's start state does not store such a long history. Thus, the value of a actually depends on how one reaches a! It is properly written as $v_{\boldsymbol{z}a}$, where $\boldsymbol{z}a$ is a path ending with a and $\boldsymbol{z}$ is sufficiently long to determine h_p.[11]

Formally, let $\mathcal{Z}_a$ be a "partitioning" set of paths to a, such that any path in cq_ϕ from an initial state to the start state of a must have *exactly one* $\boldsymbol{z} \in \mathcal{Z}_a$ as a suffix, and each $\boldsymbol{z} \in \mathcal{Z}_a$ is sufficiently long so that $v_{\boldsymbol{z}a}$ is well-defined. We can now find the expected value as $\bar{r} = \sum_a \sum_{\boldsymbol{z} \in \mathcal{Z}_a} \alpha_{\boldsymbol{z}} \left(\prod_{z \in \boldsymbol{z}} k_z\right) k_a v_{\boldsymbol{z}a} \beta_a$.

The above method permits p_θ to score the tag sequences of length n_p that are hypothesized by cq_ϕ. One can regard it as *implicitly* running the generalized forward-backward algorithm over a larger FST that marries the structure of cq_ϕ with the n_p-gram HMM structure,[12] so that each value is again local to a single arc $\overline{\boldsymbol{z}a}$. However, it saves space by working directly on cq_ϕ (which has manageable size because we deliberately kept n_q small), rather than materializing the larger FST (as bad as increasing n_q to n_p).

The $\mathcal{Z}_a$ trick uses $O(CT^{n_q})$ rather than $O(CT^{n_p})$ space to store the FST, where C is the number of arcs in c (= number of training n-grams) and T is the number of tag types. With or without the trick, runtime is $O(CT^{n_p} + BCT^{n_q})$, where B is the num-

[8] The total value is then the sum of the logs, i.e., the log of the product. This works because q_ϕ is unambiguous, i.e., it computes $q_\phi(\boldsymbol{t} \mid \boldsymbol{w})$ as a product along a single accepting path, rather than summing over multiple paths.

[9] The special case of a failure arc a goes from $[h_c, h_q]$ to $[h'_c, h_q]$, where h'_c is a backed-off version of h_c. It is labeled with $\Phi : \epsilon$, which does not contribute to the word string or tag string accepted along a path. Its weight k_a is the weight $c(\Phi \mid h_c)$ of the corresponding failure arc in c from h_c to h'_c. We define $v_a \stackrel{\text{def}}{=} 0$, so it does not contribute to the total value.

[10] Recall that the forward probability of each state is defined recursively from the forward probabilities of the states that have arcs leading to it. As our FST is cyclic, it is not possible to visit the states in topologically sorted order. We instead solve these simultaneous equations by a relaxation algorithm (Eisner, 2002, section 5): repeatedly sweep through all states, updating their forward probability, until the total forward probability of all final states is close to the correct total of $1 = \sum_{\boldsymbol{w},\boldsymbol{t}} cq_\phi(\boldsymbol{w}, \boldsymbol{t})$ (showing that we have covered all high-prob paths). A corresponding backward relaxation is actually not needed yet (we do need it for $\hat{\beta}$ in section 3.4): backward probabilities are just 1, since cq_ϕ is constructed with locally normalized probabilities.

When we rerun the forward-backward algorithm after a parameter update, we use the previous solution as a starting point for the relaxation algorithm. This greatly speeds convergence.

[11] By concatenating $\boldsymbol{z}$'s start state's h_q with the tags along $\boldsymbol{z}$. Typically $\boldsymbol{z}$ has length $n_p - n_q$ (and $\mathcal{Z}_a$ consists of the paths of that length to a's start state). However, $\boldsymbol{z}$ may be longer if it contains Φ arcs, or shorter if it begins with an initial state.

[12] Constructed by lazy finite-state intersection of cq_ϕ and p_θ (Mohri et al., 2000). These do not have to be n-gram taggers, but must be same-length FSTs (these are closed under intersection) and unambiguous. Define arc *values* in both FSTs such that for any $(\boldsymbol{w}, \boldsymbol{t})$, cq_ϕ and p_θ accept $(\boldsymbol{w}, \boldsymbol{t})$ along unique paths of total values $v = -\log q_\phi(\boldsymbol{t} \mid \boldsymbol{w})$ and $v' = \log p_\theta(\boldsymbol{w}, \boldsymbol{t})$, respectively. We now lift the weights into the expectation semiring (Eisner, 2002) as follows. In cq_ϕ, replace arc a's weight k_a with the semiring weight $\langle k_a, k_a v_a \rangle$. In p_θ, replace arc a''s weight with $\langle 1, v'_{a'} \rangle$. Then if $k = cq_\phi(\boldsymbol{w}, \boldsymbol{t})$, the intersected FST accepts $(\boldsymbol{w}, \boldsymbol{t})$ with weight $\langle k, k(v + v') \rangle$. The expectation of $v + v'$ over all paths is then a sum $\sum_{\overline{\boldsymbol{z}a}} \alpha_{\overline{\boldsymbol{z}a}} r_{\overline{\boldsymbol{z}a}} \beta_{\overline{\boldsymbol{z}a}}$ over arcs $\overline{\boldsymbol{z}a}$ of the intersected FST—we are using $\overline{\boldsymbol{z}a}$ to denote the arc in the intersected FST that corresponds to "a in cq_ϕ when reached via path $\boldsymbol{z}$," and $r_{\overline{\boldsymbol{z}a}}$ to denote the second component of its semiring weight. Here $\alpha_{\overline{\boldsymbol{z}a}}$ and $\beta_{\overline{\boldsymbol{z}a}}$ denote the forward and backward probabilities in the intersected FST, defined from the first components of the semiring weights. We can get them more efficiently from the results of running forward-backward on the smaller cq_ϕ: $\alpha_{\overline{\boldsymbol{z}a}} = \alpha_{\boldsymbol{z}} \prod_{z \in \boldsymbol{z}} k_z$ and $\beta_{\overline{\boldsymbol{z}a}} = \beta_a = 1$.

ber of forward-backward sweeps (footnote 10). The ordinary forward algorithm requires $n_q = n_p$ and takes $O(CT^{n_p})$ time *and* space on a length-C string.

3.4 Computing the gradient as well

To maximize our objective (5), we compute its gradient with respect to θ and ϕ. We follow an efficient recipe from Li and Eisner (2009, section 5, case 3). The runtime and space match those of section 3.3, except that the runtime rises to $O(BCT^{n_p})$.[13]

First suppose that each v_a is local to a single arc. We replace each weight k_a with $\hat{k}_a = \langle k_a, k_a v_a \rangle$ in the so-called expectation semiring, whose sum and product operations can be found in Li and Eisner (2009, Table 1). Using these in the forward-backward algorithm yields quantities $\hat{\alpha}_a$ and $\hat{\beta}_a$ that also fall in the expectation semiring.[14] (Their first components are the old α_a and β_a.) The desired gradient[15] $\langle \nabla\bar{k}, \nabla\bar{r} \rangle$ is $\sum_a \hat{\alpha}_a (\nabla\hat{k}_a)\hat{\beta}_a$,[16] where $\nabla\hat{k}_a = (\nabla k_a, \nabla(k_a v_a)) = (\nabla k_a, (\nabla k_a)v_a + k_a(\nabla v_a))$. Here ∇ gives the vector of partial derivatives with respect to *all* ϕ and θ parameters. Yet each $\nabla\hat{k}_a$ is sparse, with only 3 nonzero components, because $\hat{k}_a$ depends on only one ϕ parameter (ϕ_a) and two θ parameters (via θ_a as defined in section 3.3).

When $n_p > n_q$, we sum not over arcs a of cq_ϕ but over arcs $\overline{\boldsymbol{z}a}$ of the larger FST (footnote 12). Again we can do this *implicitly*, by using the short path $\boldsymbol{z}a$ in cq_ϕ in place of the arc $\overline{\boldsymbol{z}a}$. Each state of cq_ϕ must then store $\hat{\alpha}$ and $\hat{\beta}$ values for *each* of the $T^{n_p - n_q}$ states of the larger FST that it corresponds to. (In the case $n_p - n_q = 1$, as in our experiments, this fortunately does not increase the total asymptotic space, since each state of cq_ϕ already has to store T arcs.)

With more cleverness, one can eliminate this extra storage while preserving asymptotic runtime (still using sparse vectors). Find $\langle \nabla\bar{k}, (\nabla\bar{r})^{(1)} \rangle = \sum_a \hat{\alpha}_a \langle \nabla k_a, 0 \rangle \hat{\beta}_a$. Also find $\langle \bar{r}, (\nabla\bar{r})^{(2)} \rangle = \sum_a \sum_{\boldsymbol{z} \in \mathcal{Z}_a} \alpha_{\boldsymbol{z}} \left(\prod_{z \in \boldsymbol{z}} \langle k_z, \nabla k_z \rangle \right) \langle k_a v_{\boldsymbol{z}a}, \nabla(k_a v_{\boldsymbol{z}a}) \rangle \beta_a$. Now our desired gradient $\nabla\bar{r}$ emerges as $(\nabla\bar{r})^{(1)} + (\nabla\bar{r})^{(2)}$. The computation of $(\nabla\bar{r})^{(1)}$ uses *modified* definitions of $\hat{\alpha}_a$ and $\hat{\beta}_a$ that depend only on (respectively) the source and target states of a—not $\overline{\boldsymbol{z}a}$.[17] To compute them, initialize $\hat{\alpha}$ (respectively $\hat{\beta}$) at each state to $\langle 1, 0 \rangle$ or $\langle 0, 0 \rangle$ according to whether the state is initial (respectively final). Now iterate repeatedly (footnote 10) over all arcs a: Add $\hat{\alpha}_a \langle k_a, 0 \rangle + \sum_{\boldsymbol{z} \in \mathcal{Z}_a} \alpha_{\boldsymbol{z}} \left(\prod_{z \in \boldsymbol{z}} k_z \right) \langle 0, k_a v_{\boldsymbol{z}a} \rangle$ to the $\hat{\alpha}$ at a's *target* state. Conversely, add $\langle k_a, 0 \rangle \hat{\beta}_a$ to the $\hat{\beta}$ at a's *source* state, and for each $\boldsymbol{z} \in \mathcal{Z}_a$, add $\left(\prod_{z \in \boldsymbol{z}} k_z \right) \langle 0, k_a v_{\boldsymbol{z}a} \rangle \beta_a$ to the $\hat{\beta}$ at $\boldsymbol{z}$'s source state.

3.5 Locally optimizing the objective

Recall that cq_ϕ associates with each $[h_c, h_q, w]$ a block of ϕ parameters that must be ≥ 0 and sum to 1. Our optimization method must enforce these constraints. A standard approach is to use a projected gradient method, where after each gradient step on ϕ, the parameters are projected back onto the probability simplex. We implemented another standard approach: reexpress each block of parameters $\{\phi_a : a \in \mathcal{A}\}$ as $\phi_a \stackrel{\text{def}}{=} \exp \eta_a / \sum_{b \in \mathcal{A}} \exp \eta_b$, as is possible iff the ϕ_a parameters satisfy the constraints. We then follow the gradient of $\bar{r}$ with respect to the new η parameters, given by $\partial\bar{r}/\partial\eta_a = \phi_a(\partial\bar{r}/\partial\phi_a - E_{\mathcal{A}})$ where $E_{\mathcal{A}} = \sum_b \phi_b (\partial\bar{r}/\partial\phi_b)$.

Another common approach is block coordinate ascent on θ and ϕ—this is "variational EM." **M-step:** Given ϕ, we can easily find optimal estimates of the emission and transition probabilities θ. They are respectively proportional to the posterior expected counts of arcs a and paths $\boldsymbol{z}a$ under cq_ϕ, namely $N \cdot \alpha_a k_a \beta_a$ and $N \cdot \alpha_{\boldsymbol{z}} \left(\prod_{z \in \boldsymbol{z}} k_z \right) k_a \beta_a$. **E-step:** Given θ, we cannot easily find the optimal ϕ (even if $n_q = n_p$).[18] This was the rea-

[13] An alternative would be to apply back-propagation (reverse-mode automatic differentiation) to section 3.3's computation of the objective. This would achieve the same runtime as in section 3.3, but would need as much space as time.

[14] This also computes our objective $\bar{r}$: summing the $\hat{\alpha}$'s of the final states of cq_ϕ gives $\langle \bar{k}, \bar{r} \rangle$ where $\bar{k} = 1$ is the total probability of all paths. This alternative computation of the expectation $\bar{r}$, using the forward algorithm (instead of forward-backward) but over the expectation semiring, was given by Eisner (2002).

[15] We are interested in $\nabla\bar{r}$. $\nabla\bar{k}$ is just a byproduct. We remark that $\nabla\bar{k} \neq 0$, even though $\bar{k} = 1$ for any valid parameter vector ϕ (footnote 14), as increasing ϕ *invalidly* can increase $\bar{k}$.

[16] By a product of pairs we always mean $\langle k, r \rangle \langle s, t \rangle \stackrel{\text{def}}{=} \langle ks, kt + rs \rangle$, just as in the expectation semiring, even though the pair $\nabla\hat{k}_a$ is not in that semiring (its components are vectors rather than scalars). See (Li and Eisner, 2009, section 4.3). We also define scalar-by-pair products as $k\langle s, t \rangle \stackrel{\text{def}}{=} \langle ks, kt \rangle$.

[17] First components α_a and β_a remain as in cq_ϕ. $\hat{\alpha}_a$ sums paths to a. $\langle \nabla k_a, 0 \rangle \hat{\beta}_a$ can't quite sum over paths starting with a (their early weights depend on $\boldsymbol{z}$), but $(\nabla\bar{r})^{(2)}$ corrects this.

[18] Recall that cq_ϕ must have locally normalized probabilities (to ensure that its marginal is c). If $n_q = n_p$, the optimal ϕ is as follows: we can reduce the variational gap to 0 by setting

son for gradient ascent. However, for any *single* sum-to-1 block of parameters $\{\phi_a : a \in \mathcal{A}\}$, it is easy to find the optimal values if the others are held fixed. We maximize $L_\mathcal{A} \stackrel{\text{def}}{=} \bar{r} + \lambda_\mathcal{A} \sum_{a \in \mathcal{A}} \phi_a$, where $\lambda_\mathcal{A}$ is a Lagrange multiplier chosen so that the sum is 1. The partial derivative $\partial \bar{r} / \partial \phi_a$ can be found using methods of section 3.4, restricting the sums to $\boldsymbol{z}a$ for the given a. For example, following paragraphs 2–3 of section 3.4, let $\langle \alpha_a, r_a \rangle \stackrel{\text{def}}{=} \sum_{\boldsymbol{z} \in \mathcal{Z}_a} \langle \alpha_{\overline{\boldsymbol{z}a}}, r_{\overline{\boldsymbol{z}a}} \rangle$ where $\langle \alpha_{\overline{\boldsymbol{z}a}}, r_{\overline{\boldsymbol{z}a}} \rangle \stackrel{\text{def}}{=} \hat{\alpha}_{\overline{\boldsymbol{z}a}} \hat{\beta}_{\overline{\boldsymbol{z}a}}$.[19] Setting $\partial L_\mathcal{A} / \partial \phi_a = 0$ implies that ϕ_a is proportional to $\exp((r_a + \sum_{\boldsymbol{z} \in \mathcal{Z}_a} \alpha_{\overline{\boldsymbol{z}a}} \log \theta_{\overline{\boldsymbol{z}a}}) / \alpha_a)$.[20]

Rather than doing block coordinate ascent by updating one ϕ block at a time (and then recomputing r_a values for all blocks, which is slow), one can take an approximate step by updating all blocks in parallel. We find that replacing the E-step with a single parallel step still tends to improve the objective, and that this approximate variational EM is faster than gradient ascent with comparable results.[21]

4 Experiments

4.1 Constrained unsupervised HMM learning

We follow the unsupervised POS tagging setup of Merialdo (1994) and many others (Smith and Eisner, 2005; Haghighi and Klein, 2006; Toutanova and Johnson, 2007; Goldwater and Griffiths, 2007; Johnson, 2007). Given a corpus of sentences, one seeks the maximum-likelihood or MAP parameters of a bigram HMM ($n_p = 2$). The observed sentences, for us, are replaced by the faux sentences extrapolated from observed n-grams via the language model c.

The states of the HMM correspond to POS tags as in Figure 1. All transitions are allowed, but not all emissions. If a word is listed in a provided "dictionary" with its possible tags, then other tags are given 0 probability of emitting that word. The EM algorithm uses the corpus to learn transition and emission probabilities that explain the data under this constraint. The constraint ensures that the learned states have something to do with true POS tags.

Merialdo (1994) spawned a long line of work on this task. Ideas have included Bayesian learning methods (MacKay, 1997; Goldwater and Griffiths, 2007; Johnson, 2007), better initial parameters (Goldberg et al., 2008), and learning how to constrain the possible parts of speech for a word (Ravi and Knight, 2008), as well as non-HMM sequence models (Smith and Eisner, 2005; Haghighi and Klein, 2006; Toutanova and Johnson, 2007).

Most of this work has used the Penn Treebank (Marcus et al., 1993) as a dataset. While this million-word *Wall Street Journal* (WSJ) corpus is one of the largest that is *manually* annotated with parts of speech, unsupervised learning methods could take advantage of vast amounts of unannotated text. In practice, runtime concerns have sometimes led researchers to use small subsets of the Penn Treebank (Goldwater and Griffiths, 2007; Smith and Eisner, 2005; Haghighi and Klein, 2006). Our goal is to point the way to using even larger datasets.

The reason for all this past research is that (Merialdo, 1994) was a *negative* result: while EM is guaranteed to improve the model's likelihood, it degrades the match between the latent states and true parts of speech (if the starting point is a good one obtained with some supervision). Thus, for the task of POS induction, there must be something wrong with the HMM model, the likelihood objective, or the search procedure. It is clear that the model is far too weak: there are many latent variables in natural language, so the HMM may be picking up on something other than POS tags. Ultimately, fixing this will require richer models with many more parameters. But learning these (lexically specific) parameters will require large training datasets—hence our present methodological exploration on whether it is possible to scale up the original setting.

$q_\phi(t \mid h_c w, h_q)$ to the probability that $\boldsymbol{t}$ begins with t if we randomly draw a suffix $\boldsymbol{w} \sim c(\cdot \mid h_c w)$ and randomly tag $w\boldsymbol{w}$ with $\boldsymbol{t} \sim p_\theta(\cdot \mid w\boldsymbol{w}, h_q)$. This is equivalent to using p_θ with the backward algorithm to conditionally tag *each* possible suffix.

[19] The first component of $\hat{\alpha}_{\overline{\boldsymbol{z}a}} \hat{\beta}_{\overline{\boldsymbol{z}a}}$ is $\alpha_{\overline{\boldsymbol{z}a}} \beta_{\overline{\boldsymbol{z}a}} = \alpha_{\overline{\boldsymbol{z}a}} \cdot 1$.

[20] If a is an arc of cq_ϕ then $\partial \bar{r} / \partial \phi_a$ is the second component of $\sum_{\boldsymbol{z} \in \mathcal{Z}_a} \hat{\alpha}_{\overline{\boldsymbol{z}a}} (\partial \hat{k}_{\overline{\boldsymbol{z}a}} / \partial \phi_a) \hat{\beta}_{\overline{\boldsymbol{z}a}}$. Then $\partial L_\mathcal{A} / \partial \phi_a$ works out to $\sum_{\boldsymbol{z} \in \mathcal{Z}_a} c_a (r_{\overline{\boldsymbol{z}a}} + \alpha_{\overline{\boldsymbol{z}a}} (\log \theta_{\overline{\boldsymbol{z}a}} - \log \phi_a - 1)) + \lambda_\mathcal{A}$. Set to 0 and solve for ϕ_a, noting that $c_a, \alpha_a, \lambda_\mathcal{A}$ are constant over $a \in \mathcal{A}$.

[21] In retrospect, an even faster strategy might be to do a *series* of block ϕ and $\hat{\beta}$ updates, updating $\hat{\beta}$ at a state (footnote 10) immediately after updating ϕ on the arcs leading from that state, which allows a better block update at predecessor states. On an *acyclic* machine, a single *backward* pass of this sort will reduce the variational gap to 0 if $n_q = n_p$ (footnote 18). This is because, thanks to the up-to-date $\hat{\beta}$, each block of arcs gets new ϕ weights in proportion to relative suffix path probabilities *under the new* θ. After this backward pass, a single *forward* pass can update the α values and collect expected counts for the M-step that will update θ. Standard EM is a special case of this strategy.

4.2 Setup

We investigate how much performance degrades when we approximate the corpus *and* train approximately with $n_q = 1$. We examine two measures: likelihood on a held-out corpus and accuracy in POS tagging. We train on corpora of three different sizes:
• WSJ-big (910k words → 441k n-grams @ cutoff 3),
• Giga-20 (20M words → 2.9M n-grams @ cutoff 10),
• Giga-200 (200M wds → 14.4M n-grams @ cutoff 20).
These were drawn from the Penn Treebank (sections 2–23) and the English Gigaword corpus (Parker et al., 2009). For held-out evaluation, we use WSJ-small (Penn Treebank section 0) or WSJ-big.

We estimate backoff language models for these corpora based on collections of n-grams with $n \leq 5$. In this work, we select the n-grams by simple count cutoffs as shown above,[22] taking care to keep all 2-grams as mentioned in footnote 2.

Similar to Merialdo (1994), we use a tag dictionary which limits the possible tags of a word to those it was observed with in the WSJ, provided that the word was observed at least 5 times in the WSJ. We used the reduced tagset of Smith and Eisner (2005), which collapses the original 45 fine-grained part-of-speech tags into just 17 coarser tags.

4.3 Results

In all experiments, our method achieves similar accuracy though slightly worse likelihood. Although this method is meant to be a fast approximation of EM, standard EM is faster on the smallest dataset (WSJ-big). This is because this corpus is not much bigger than the 5-gram language model built from it (at our current pruning level), and so the overhead of the more complex n-gram EM method is a net disadvantage. However, when moving to larger corpora, the iterations of n-gram EM become as fast as standard EM and then faster. We expect this trend to continue as one moves to much larger datasets, as the compression ratio of the pruned language model relative to the original corpus will only improve. The Google n-gram corpus is based on 50× more data than our largest but could be handled in RAM.

[22] Entropy-based pruning (Stolcke, 2000) may be a better selection method when one is in a position to choose. However, count cutoffs were already used in the creation of the Google n-gram corpus, and more complex methods of pruning may not be practical for very large datasets.

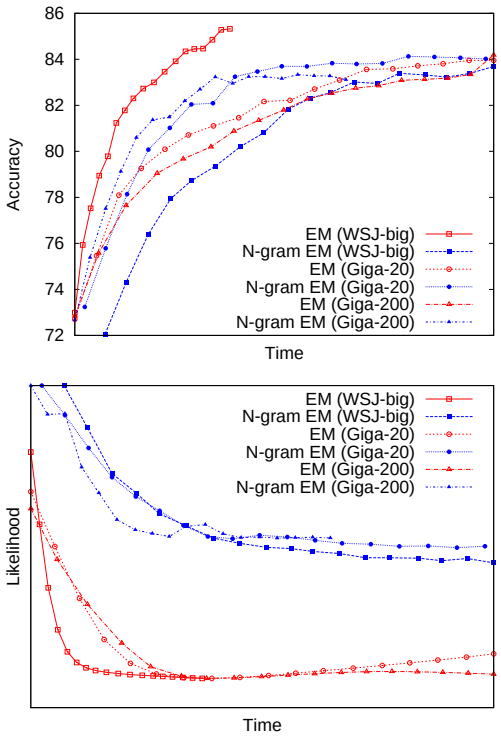

Figure 2: POS-tagging accuracy and log-likelihood after each iteration, measured on WSJ-big when training on the Gigaword datasets, else on WSJ-small. Runtime and log-likelihood are scaled differently for each dataset. Replacing EM with our method changes runtime per iteration from 1.4s → 3.5s, 48s → 47s, and 506s → 321s.

5 Conclusions

We presented a general approach to training generative models on a *distribution* rather than on a training sample. We gave several motivations for this novel problem. We formulated an objective function similar to MAP, and presented a variational lower bound.

Algorithmically, we gave nontrivial general methods for computing and optimizing our variational lower bound for *arbitrary* finite-state data distributions c, generative models p, and variational families q, provided that p and q are unambiguous same-length FSTs. We also gave details for specific useful families for c, p, and q.

As proof of principle, we used a traditional HMM POS tagging task to demonstrate that we can train a model from n-grams almost as accurately as from full sentences, and do so faster to the extent that the n-gram dataset is smaller. More generally, we offer our approach as an intriguing new tool to help semi-supervised learning benefit from very large datasets.

References

Cyril Allauzen, Mehryar Mohri, and Brian Roark. 2003. Generalized algorithms for constructing statistical language models. In *Proc. of ACL*, pages 40–47.

Shane Bergsma, Dekang Lin, and Randy Goebel. 2009. Web-scale n-gram models for lexical disambiguation. In *Proc. of IJCAI*.

Thorsten Brants and Alex Franz. 2006. Web 1T 5-gram version 1. Linguistic Data Consortium, Philadelphia. LDC2006T13.

Thorsten Brants, Ashok C. Popat, Peng Xu, Franz J. Och, and Jeffrey Dean. 2007. Large language models in machine translation. In *Proc. of EMNLP*.

Tim Dawborn and James R. Curran. 2009. CCG parsing with one syntactic structure per n-gram. In *Australasian Language Technology Association Workshop*, pages 71–79.

Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38.

Jason Eisner. 2002. Parameter estimation for probabilistic finite-state transducers. In *Proc. of ACL*, pages 1–8.

Yoav Goldberg, Meni Adler, and Michael Elhadad. 2008. EM can find pretty good HMM POS-taggers (when given a good start). In *Proc. of ACL*, pages 746–754.

Sharon Goldwater and Thomas Griffiths. 2007. A fully Bayesian approach to unsupervised part-of-speech tagging. In *Proc. of ACL*, pages 744–751.

Aria Haghighi and Dan Klein. 2006. Prototype-driven learning for sequence models. In *Proc. of NAACL*, pages 320–327.

Daniel Hsu, Sham M. Kakade, and Tong Zhang. 2009. A spectral algorithm for learning hidden Markov models. In *Proc. of COLT*.

Mark Johnson. 2007. Why doesn't EM find good HMM POS-taggers? In *Proc. of EMNLP-CoNLL*, pages 296–305.

M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul. 1999. An introduction to variational methods for graphical models. In M. I. Jordan, editor, *Learning in Graphical Models*. Kluwer.

Mirella Lapata and Frank Keller. 2005. Web-based models for natural language processing. *ACM Transactions on Speech and Language Processing*.

Zhifei Li and Jason Eisner. 2009. First- and second-order expectation semirings with applications to minimum-risk training on translation forests. In *Proc. of EMNLP*, pages 40–51.

Percy Liang, Hal Daumé III, and Dan Klein. 2008. Structure compilation: Trading structure for features. In *International Conference on Machine Learning (ICML)*, Helsinki, Finland.

D. Lin, K. Church, H. Ji, S. Sekine, D. Yarowsky, S. Bergsma, K. Patil, E. Pitler, R. Lathbury, V. Rao, K. Dalwani, and S. Narsale. 2009. Unsupervised acquisition of lexical knowledge from n-grams. Summer workshop technical report, Center for Language and Speech Processing, Johns Hopkins University.

David J. C. MacKay. 1997. Ensemble learning for hidden Markov models. `http://www.inference.phy.cam.ac.uk/mackay/abstracts/ensemblePaper.html`.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*.

Andrew McCallum, Dayne Freitag, and Fernando Pereira. 2000. Maximum entropy Markov models for information extraction and segmentation. In *Proc. of ICML*, pages 591–598.

B. Merialdo. 1994. Tagging English text with a probabilistic model. *Computational Linguistics*, 20(2):155–171.

J.-B. Michel, Y. K. Shen, A. P. Aiden, A. Veres, M. K. Gray, W. Brockman, The Google Books Team, J. P. Pickett, D. Hoiberg, D. Clancy, P. Norvig, J. Orwant, S. Pinker, M. A. Nowak, and E. L. Aiden. 2010. Quantitative analysis of culture using millions of digitized books. *Science*, 331(6014):176–182.

Mehryar Mohri and Mark-Jan Nederhof. 2001. Regular approximation of context-free grammars through transformation. In Jean-Claude Junqua and Gertjan van Noord, editors, *Robustness in Language and Speech Technology*, chapter 9, pages 153–163. Kluwer Academic Publishers, The Netherlands, February.

Mehryar Mohri, Fernando Pereira, and Michael Riley. 2000. The design principles of a weighted finite-state transducer library. *Theoretical Computer Science*, 231(1):17–32, January.

Radford M. Neal and Geoffrey E. Hinton. 1998. A view of the EM algorithm that justifies incremental, sparse, and other variants. In M.I. Jordan, editor, *Learning in Graphical Models*, pages 355–368. Kluwer.

Mark-Jan Nederhof. 2000. Practical experiments with regular approximation of context-free languages. *Computational Linguistics*, 26(1).

Robert Parker, David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2009. English Gigaword fourth edition. Linguistic Data Consortium, Philadelphia. LDC2009T13.

V. Punyakanok, D. Roth, W. Yih, and D. Zimak. 2005. Learning and inference over constrained output. In *Proc. of IJCAI*, pages 1124–1129.

Lawrence R. Rabiner. 1989. A tutorial on hidden Markov models and selected applications in speech

recognition. *Proc. of the IEEE*, 77(2):257–286, February.

Sujith Ravi and Kevin Knight. 2008. Minimized models for unsupervised part-of-speech tagging. In *Proc. of ACL*, pages 504–512.

Noah A. Smith and Jason Eisner. 2005. Contrastive estimation: Training log-linear models on unlabeled data. In *Proc. of ACL*, pages 354–362.

R. Staden. 1979. A strategy of DNA sequencing employing computer programs. *Nucleic Acids Research*, 6(7):2601–2610, June.

Andreas Stolcke. 2000. Entropy-based pruning of backoff language models. In *DARPA Broadcast News Transcription and Understanding Workshop*, pages 270–274.

Kristina Toutanova and Mark Johnson. 2007. A Bayesian LDA-based model for semi-supervised part-of-speech tagging. In *Proc. of NIPS*, volume 20.