Unsupervised Neural Hidden Markov Models

Ke Tran[2*] **Yonatan Bisk**[1] **Ashish Vaswani**[3*] **Daniel Marcu**[1] **Kevin Knight**[1]
[1]Information Sciences Institute, University of Southern California
[2]Informatics Institute, University of Amsterdam
[3]Google Brain, Mountain View
m.k.tran@uva.nl, ybisk@isi.edu,
avaswani@google.com, marcu@isi.edu, knight@isi.edu

Abstract

In this work, we present the first results for neuralizing an Unsupervised Hidden Markov Model. We evaluate our approach on tag induction. Our approach outperforms existing generative models and is competitive with the state-of-the-art though with a simpler model easily extended to include additional context.

1 Introduction

Probabilistic graphical models are among the most important tools available to the NLP community. In particular, the ability to train generative models using Expectation-Maximization (EM), Variational Inference (VI), and sampling methods like MCMC has enabled the development of unsupervised systems for tag and grammar induction, alignment, topic models and more. These latent variable models discover hidden structure in text which aligns to known linguistic phenomena and whose clusters are easily identifiable.

Recently, much of supervised NLP has found great success by augmenting or replacing context, features, and word representations with embeddings derived from Deep Neural Networks. These models allow for learning highly expressive non-convex functions by simply backpropagating prediction errors. Inspired by Berg-Kirkpatrick et al. (2010), who bridged the gap between supervised and unsupervised training with features, we bring neural networks to unsupervised learning by providing evidence that even in unsupervised settings, simple neural network models trained to maximize the marginal likelihood can outperform more complicated models that use expensive inference.

In this work, we show how a single latent variable sequence model, Hidden Markov Models (HMMs), can be implemented with neural networks by simply optimizing the incomplete data likelihood. The key insight is to perform standard forward-backward inference to compute posteriors of latent variables and then backpropagate the posteriors through the networks to maximize the likelihood of the data.

Using features in unsupervised learning has been a fruitful enterprise (Das and Petrov, 2011; Berg-Kirkpatrick and Klein, 2010; Cohen et al., 2011) and attempts to combine HMMs and Neural Networks date back to 1991 (Bengio et al., 1991). Additionally, similarity metrics derived from word embeddings have also been shown to improve unsupervised word alignment (Songyot and Chiang, 2014).

Interest in the interface of graphical models and neural networks has grown recently as new inference procedures have been proposed (Kingma and Welling, 2014; Johnson et al., 2016). Common to this work and ours is the use of neural networks to produce potentials. The approach presented here is easily applied to other latent variable models where inference is tractable and are typically trained with EM. We believe there are three important strengths:

1. Using a neural network to produce model probabilities allows for seamless integration of additional context not easily represented by conditioning variables in a traditional model.

[*] This research was carried out while all authors were at the Information Sciences Institute.

2. Gradient based training trivially allows for multiple objectives in the same loss function.

3. Rich model representations do not saturate as quickly and can therefore utilize large quantities of unlabeled text.

Our focus in this preliminary work is to present a generative neural approach to HMMs and demonstrate how this framework lends itself to modularity (e.g. the easy inclusion of morphological information via Convolutional Neural Networks §5), and the addition of extra conditioning context (e.g. using an RNN to model the sentence §6). Our approach will be demonstrated and evaluated on the simple task of part-of-speech tag induction. Future work, should investigate the second and third proposed strengths.

2 Framework

Graphical models have been widely used in NLP. Typically potential functions $\psi(\mathbf{z}, \mathbf{x})$ over a set of latent variables, $\mathbf{z}$, and observed variables, $\mathbf{x}$, are defined based on hand-crafted features. Moreover, independence assumptions between variables are often made for the sake of tractability. Here, we propose using neural networks (NNs) to produce the potentials since neural networks are universal approximators. Neural networks can extract useful task-specific abstract representations of data. Additionally, Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) based Recurrent Neural Networks (RNNs), allow for modeling unbounded context with far fewer parameters than naive one-hot feature encodings. The reparameterization of potentials with neural networks (NNs) is seamless:

$$\psi(\mathbf{z}, \mathbf{x}) = f_{\text{NN}}(\mathbf{z}, \mathbf{x} \mid \theta) \tag{1}$$

The sequence of observed variables are denoted as $\mathbf{x} = \{x_1, \ldots, x_n\}$. In unsupervised learning, we aim to find model parameters θ that maximize the evidence $p(\mathbf{x} \mid \theta)$. We focus on cases when the posterior is tractable and we can use Generalized EM (Dempster et al., 1977) to estimate θ.

$$\begin{align} p(\mathbf{x}) &= \sum_{z} p(\mathbf{x}, \mathbf{z}) \tag{2} \\ &= \mathbb{E}_{q(\mathbf{z})}[\ln p(\mathbf{x}, \mathbf{z} \mid \theta)] + \mathrm{H}[q(\mathbf{z})] \tag{3} \\ &+ \mathrm{KL}\left(q(\mathbf{z}) \parallel p(\mathbf{z} \mid \mathbf{x}, \theta)\right) \tag{4} \end{align}$$

where $q(\mathbf{z})$ is an arbitrary distribution, and H is the entropy function. The E-step of EM estimates the posterior $p(\mathbf{z} \mid \mathbf{x})$ based on the current parameters θ. In the M-step, we choose $q(\mathbf{z})$ to be the posterior $p(\mathbf{z} \mid \mathbf{x})$, setting the KL-divergence to zero. Additionally, the entropy term $\mathrm{H}[q(\mathbf{z})]$ is a constant and can therefore be dropped. This means updating θ only requires maximizing $\mathbb{E}_{p(\mathbf{z} \mid \mathbf{x})}[\ln p(\mathbf{x}, \mathbf{z} \mid \theta)]$. The gradient is therefore defined in terms of the gradient of the joint probability scaled by the posteriors:

$$J(\theta) = \sum_{\mathbf{z}} p(\mathbf{z} \mid \mathbf{x}) \frac{\partial \ln p(\mathbf{x}, \mathbf{z} \mid \theta)}{\partial \theta} \tag{5}$$

In order to perform the gradient update in Eq 5, we need to compute the posterior $p(\mathbf{z} \mid \mathbf{x})$. This can be done efficiently with the Message Passing algorithm. Note that, in cases where the derivative $\frac{\partial}{\partial \theta} \ln p(\mathbf{x}, \mathbf{z} \mid \theta)$ is easy to evaluate, we can perform direct marginal likelihood optimization (Salakhutdinov et al., 2003). We do not address here the question of semi-supervised training, but believe the framework we present lends itself naturally to the incorporation of constraints or labeled data. Next, we demonstrate the application of this framework to HMMs in the service of part-of-speech tag induction.

Text	Pierre	Vinken	will	join	the	board
PTB	`NNP`	`NNP`	`MD`	`VB`	`DT`	`NN`

Table 1: Example Part-of-Speech tagged text.

3 Part-of-Speech Induction

Part-of-speech tags encode morphosyntactic information about a language and are a fundamental tool in downstream NLP applications. In English, the Penn Treebank (Marcus et al., 1994) distinguishes 36 categories and punctuation. Tag induction is the task of taking raw text and both discovering these latent clusters and assigning them to words in situ. Classes can be very specific (e.g. six types of verbs in English) to their syntactic role. Example tags are shown in Table 1. In this example, *board* is labeled as a singular noun while *Pierre Vinken* is a singular proper noun.

Two natural applications of induced tags are as the basis for grammar induction (Spitkovsky et al., 2011; Bisk et al., 2015) or to provide a syntactically informed, though unsupervised, source of word embeddings.

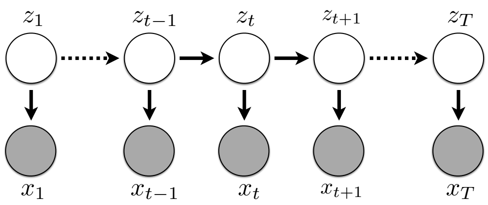

Figure 1: Pictorial representation of a Hidden Markov Model. Latent variable (z_t) transitions depend on the previous value (z_{t-1}), and emit an observed word (x_t) at each time step.

3.1 The Hidden Markov Model

A common model for this task, and our primary workhorse, is the Hidden Markov Model trained with the unsupervised message passing algorithm, Baum-Welch (Welch, 2003).

Model HMMs model a sentence by assuming that (a) every word token is generated by a latent class, and (b) the current class at time t is conditioned on the local history $t-1$. Formally, this gives us an emission $p(x_t \mid z_t)$ and transition $p(z_t \mid z_{t-1})$ probability. The graphical model is drawn pictorially in Figure 1, where shaded circles denote observations and empty ones are latent. The probability of a given sequence of observations $\mathbf{x}$ and latent variables $\mathbf{z}$ is given by multiplying transitions and emissions across all time steps (Eq. 6). Finding the optimal sequence of latent classes corresponds to computing an argmax over the values of $\mathbf{z}$.

$$p(\mathbf{x}, \mathbf{z}) = \prod_{t=1}^{n+1} p(z_t \mid z_{t-1}) \prod_{t=1}^{n} p(x_t \mid z_t) \qquad (6)$$

Because our task is unsupervised we do not have a priori access to these distributions, but they can be estimated via Baum-Welch. The algorithm's outline is provided in Algorithm 1.

Training an HMM with EM is highly non-convex and likely to get stuck in local optima (Johnson, 2007). Despite this, sophisticated Bayesian smoothing leads to state-of-the-art performance (Blunsom and Cohn, 2011). Blunsom and Cohn (2011) further extend the HMM by augmenting its emission distributions with character models to capture morphological information and a tri-gram transition matrix which conditions on the previous two states. Recently, Lin et al. (2015) extended several models including the HMM to include pre-trained word embeddings learned by different skip-gram models. Our work will fully neuralize the HMM and learn embeddings during the training of our generative model. There has also been recent work on by Rastogi et al. (2016) on neuralizing Finite-State Transducers.

Algorithm 1 Baum-Welch Algorithm

```
Randomly Initialize distributions ($\theta$)
repeat
    Compute forward messages:        $\forall_{i,t}\ \alpha_i(t)$
    Compute backward messages:       $\forall_{i,t}\ \beta_i(t)$
    Compute posteriors:
        $p(z_t = i \mid \mathbf{x}, \theta) \propto \alpha_i(t)\beta_i(t)$
        $p(z_t = i, z_{t+1} = j \mid \mathbf{x}, \theta)$
            $\propto \alpha_i(t) p(z_{t+1}{=}j | z_t{=}i)$
              $\times \beta_j(t+1) p(x_{t+1} | z_{t+1}{=}j)$
    Update $\theta$
until Converged
```

3.2 Additional Comparisons

While the main focus of our paper is the seamless extension of an unsupervised generative latent variable model with neural networks, for completeness we will also include comparisons to other techniques which do not adhere to the generative assumption. We include Brown clusters (Brown et al., 1992) as a baseline and two clustering techniques as state-of-the-art comparisons: Christodoulopoulos et al. (2011) and Yatbaz et al. (2012).

Of particular interest to us is the work of Brown et al. (1992). Brown clusters group word types through a greedy agglomerative clustering according to their mutual information across the corpus based on bigram probabilities. Brown clusters do not account for a word's membership in multiple syntactic classes, but are a very strong baseline for tag induction. It is possible our approach could be improved by augmenting our objective function to include mutual information computations or a bias towards a harder clustering.

4 Neural HMM

The aforementioned training of an HMM assumes access to two distributions: (1) Emissions with $K \times V$ parameters, and (2) Transitions with $K \times K$ parameters. Here we assume there are K clusters and V

word types in our vocabulary. Our neural HMM (NHMM) will replace these matrices with the output of simple feed-forward neural networks. All conditioning variables will be presented as input to the network and its final softmax layer will provide probabilities. This should replicate the behavior of the standard HMM, but without an explicit representation of the necessary distributions.

4.1 Producing Probabilities

Producing emission and transition probabilities allows for standard inference to take place in the model.

Emission Architecture Let $\mathbf{v}_k \in \mathbb{R}^D$ be vector embedding of tag z_k, $\mathbf{w}_i \in \mathbb{R}^D$ and b_i vector embedding and bias of word i respectively. The emission probability $p(w_i \mid z_k)$ is given by

$$p(w_i \mid z_k) = \frac{\exp(\mathbf{v}_k^\top \mathbf{w}_i + b_i)}{\sum_{j=1}^{V} \exp(\mathbf{v}_k^\top \mathbf{w}_j + b_j)} \qquad (7)$$

The emission probability can be implemented by a neural network where $\mathbf{w}_i$ is the weight of unit i at the output layer of the network. The tag embeddings $\mathbf{v}_k$ are obtained by a simple feed-forward neural network consisting of a lookup table following by a nonlinear activation (ReLU). When using morphology information (§5), we will first use another network to produce the word embedddings $\mathbf{w}_i$.

Transition Architecture We produce the transition probability directly by using a linear layer of $D \times K^2$. More specifically, let $\mathbf{q} \in \mathbb{R}^D$ be a *query embedding*. The unnormalized transition matrix $\mathbf{T}$ is computed as

$$\mathbf{T} = \mathbf{U}^\top \mathbf{q} + \mathbf{b} \qquad (8)$$

where $\mathbf{U} \in \mathbb{R}^{D \times K^2}$ and $\mathbf{b} \in \mathbb{R}^{K^2}$. We then reshape $\mathbf{T}$ to a $K \times K$ matrix and apply a softmax layer per row to produce valid transition probabilities.

4.2 Training the Neural Network

The probabilities can now be used to perform the aforementioned forward and backward passes over the data to compute posteriors. In this way, we perform the E-step as though we were training a vanilla HMM. Traditionally, these values would simply be re-normalized during the M-step to re-estimate model parameters. Instead, we use them to re-scale our gradients (following the discussion from §2). Combining the HMM factorization of the joint probability $p(\mathbf{x}, \mathbf{z})$ from Eq. 6 with the gradient from Eq. 5, yields the following update rule:

$$\begin{aligned} J(\theta) &= \sum_{\mathbf{z}} p(\mathbf{z} \mid \mathbf{x}) \frac{\partial \ln p(\mathbf{x}, \mathbf{z} \mid \theta)}{\partial \theta} \\ &= \sum_{t} \sum_{z_t} p(z_t \mid \mathbf{x}) \frac{\partial \ln p(x_t \mid z_t, \theta)}{\partial \theta} \\ &\quad + p(z_t, z_{t-1} \mid \mathbf{x}) \frac{\partial \ln p(z_t \mid z_{t-1}, \theta)}{\partial \theta} \end{aligned} \qquad (9)$$

The posteriors $p(z_t \mid \mathbf{x})$ and $p(z_t, z_{t-1} \mid \mathbf{x})$ are obtained by running Baum-Welch as shown in Algorithm 1. Where traditional supervised training can follow a clear gradient signal towards a specific assignment, here we are propagating the model's (un)certainty instead. An additional complication introduced by this paradigm is the question of how many gradient steps to take on a given minibatch. In incremental EM the posteriors are simply accumulated and normalized. Here, we repeatedly recompute gradients on a minibatch until reaching the maximum number of epochs or a convergence threshold is met.

Finally, notice that the factorization of the HMM allows us to evaluate the joint distribution $p(\mathbf{x}, \mathbf{z} \mid \theta)$ easily. We therefore employ Direct Marginal Likelihood (DML) (Salakhutdinov et al., 2003) to optimize the model's parameters. After trying both EM and DML we found EM to be slower to converge and perform slightly weaker. For this reason, the presented results will all be trained with DML.

4.3 HMM and Neural HMM Equivalence

An important result we see in Table 2 is that the Neural HMM (NHMM) performs almost identically to the HMM. At this point, we have replaced the underlying machinery, but the model still has the same information bottlenecks as a standard HMM, which limit the amount and type of information carried between words in the sentence. Additionally, both approaches are optimizing the same objective function, data likelihood, via the computation of posteriors. The equivalency is an important sanity check. The

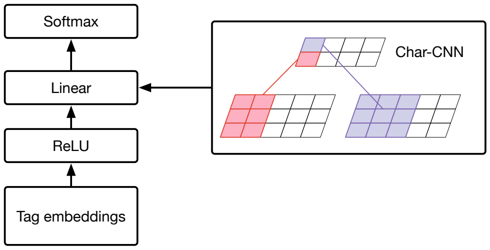

Figure 2: Computational graph of Char-CNN emission network. A character convolutional neural network is used to compute the weight of the linear layer for every minibatch.

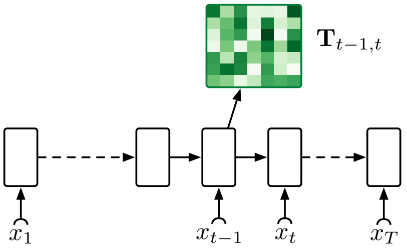

Figure 3: A graphical representation of our LSTM transition network. Transition matrix $\mathbf{T}_{t-1,t}$ from time step $t-1$ to t is computed based on the hidden state of the LSTM at time $t-1$.

following two sections will demonstrate the extensibility of this approach.

5 Convolutions for Morphology

The first benefit of moving to neural networks is the ease with which new information can be provided to the model. The first experiment we will perform is replacing words with embedding vectors derived from a Convolutional Neural Network (CNN) (Kim et al., 2016; Jozefowicz et al., 2016). We use a convolutional kernel with widths from 1 to 7, which covers up to 7 character n-grams (Figure 2). This allows the model to automatically learn lexical representations based on prefix, suffix, and stem information about a word. No additional changes to learning are required for extension.

Adding the convolution does not dramatically slow down our model because the emission distributions can be computed for the whole batch in one operation. We simply pass the whole vocabulary through the convolution in a single operation.

6 Infinite Context with LSTMs

One of the most powerful strengths of neural networks is their ability to create compact representation of data. We will explore this here in the creation of transition matrices. In particular, we chose to augment the transition matrix with all preceding words in the sentence: $p(z_t \mid z_{t-1}, w_0, \dots, w_{t-1})$. Incorporating this amount of context in a traditional HMM is intractable and impossible to estimate, as the number of parameters grows exponentially.

For this reason, we use an stacked LSTM to form a low dimensional representation of the sentence ($C_{0\dots t-1}$) which can be easily fed to our network when producing a transition matrix: $p(z_t \mid z_{t-1}, C_{0\dots t-1})$ in Figure 3. By having the LSTM only consume up to the previous word, we do not break any sequential generative model assumptions.[1] In terms of model architecture, the query embedding $\mathbf{q}$ will be replaced by a hidden state $\mathbf{h}_{t-1}$ of the LSTM at time step $t-1$.

7 Evaluation

Once a model is trained, the one best latent sequence is extracted for every sentence and evaluated on three metrics.

Many-to-One (M-1) Many-to-one computes the most common true part-of-speech tag for each cluster. It then computes tagging accuracy as if the cluster were replaced with that tag. This metric is easily gamed by introducing a large number of clusters.

One-to-One (1-1) One-to-One performs the same computation as Many-to-One but only one cluster is allowed to be assigned to a given tag. This prevents the gaming of M-1.

V-Measure (VM) V-Measure is an F-measure which trades off conditional entropy between the clusters and gold tags. Christodoulopoulos et al. (2010) found VM is to be the most informative and consistent metric, in part because it is agnostic to the number of induced tags.

8 Data and Parameters

To evaluate our approaches, we follow the existing literature and train and test on the full WSJ corpus.

[1] This interpretation does not complicate the computation of forward-backward messages when running Baum-Welch, though it does, by design, break Markovian assumption about knowledge of the past.

There are three components of our models which can be tuned. Something we have to be careful of when train and test are the same data. To avoid cheating, no values were tuned in this work.

Architecture The first parameter is the number of hidden units. We chose 512 because it was the largest power of two we could fit in memory. When we extended our model to include the convolutional emission network, we only used 128 units, due to the intensive computation of Char-CNN over the whole vocabulary per minibatch.

The second design choice was the number of LSTM layers. We used a three layer LSTM as it worked well for (Tran et al., 2016), and we applied dropout (Srivastava et al., 2014) over the vertical connections of the LSTMs (Pham et al., 2014) with a rate of 0.5.

Finally, the maximum number of inner loop updates applied per batch is set to six. We train all the models for five epochs and perform gradient clipping whenever the gradient norm is greater than five. To determine when to stop applying the gradient during training we simply check when the log probability has converged ($\frac{\text{new}-\text{old}}{\text{old}} < 10^{-4}$) or if the maximum number of inner loops has been reached. All optimization was done using Adam (Kingma and Ba, 2015) with default hyper-parameters.

Initialization In addition to architectural choices we have to initialize all of our parameters. Word embeddings (and character embeddings in the CNN) are drawn from a Gaussian $\mathcal{N}(0, 1)$. The weights of all linear layers in the model are drawn from a uniform distribution with mean zero and a standard deviation of $\sqrt{1/n_{\text{in}}}$, where n_{in} is the input dimension of the linear layer.[2] Additionally, weights for the LSTMs are initialized using $\mathcal{N}(0, 1/2n)$, where n is the number of hidden units, and the bias of the forget gate is set to 1, as suggested by Józefowicz et al. (2015). We present some parameter and modeling ablation analysis in §10.

It is worth emphasizing that parameters are shared at the lower level of our network architectures (see Figure 2 and Figure 3). Sharing parameters not only allows the networks to share statistical strength, but also reduces the computational cost of computing sufficient statistics during training due to the marginalization over latent variables.

	System	M-1	1-1	VM
Base	HMM	62.5	41.4	53.3
	Brown	68.2	49.9	63.0
SOTA	Clark (2003)	71.2		65.6
	Christodoulopoulos (2011)	72.8		66.1
	Blunsom (2011)	**77.5**		**69.8**
	Yatbaz (2012)	80.2		72.1
Our Work	NHMM	59.8	45.7	54.2
	+ Conv	74.1	48.3	66.1
	+ LSTM	65.1	52.4	60.4
	+ Conv & LSTM	**79.1**	**60.7**	**71.7**

Table 2: English Penn Treebank results with 45 induced clusters. We see significant gains from both morphology (+Conv) and extended context (+LSTM). The combination of these approaches results in a very simple system which is competitive with the best generative model in the literature.

In all of our experiments, we use minibatch size of 256 and sentences of 40 words or less due to memory constraints. Evaluation was performed on all sentence lengths. Additionally, we map all the digits to 0, but do not lower-case the data or perform any other preprocessing. All model code is available online for extension and replication at `https://github.com/ketranm/neuralHMM`.

9 Results

Our results are presented in Table 2 along with two baseline systems, and the four top performing and state-of-the-art approaches. As noted earlier, we are happy to see that our NHMM performs almost identically with the standard HMM. Second, we find that our approach, while simple and fast, is competitive with Blunsom (2011). Their Hierarchical Pitman-Yor Process for trigram HMMs with character modeling is a very sophisticated Bayesian approach and the most appropriate comparison to our work.

We see that both extended context (+LSTM) and the addition of morphological information (+Conv) provide substantial boosts to performance. Interestingly, the gains are not completely complementary, as we note that the six and twelve point gains of these additions only combine to a total of sixteen points in

[2] This is the default parameter initialization in Torch.

Configuration	M-1	1-1	VM
Uniform initialization	65.5	50.1	61.7
1 LSTM layer, no dropout	69.3	52.7	63.6
1 LSTM layer, dropout	71.0	55.7	66.2
3 LSTM layers, no dropout	72.7	52.2	65.1
Best Model	**79.1**	**60.7**	**71.7**

Table 3: Exploring different configurations of NHMM

VM improvement. This might imply that at least some of the syntactic context being captured by the LSTM is mirrored in the morphology of the language. This hypothesis is something future work should investigate with morphologically rich languages.

Finally, the newer work of Yatbaz et al. (2012) outperforms our approach. It is possible our performance could be improved by following their lead and including knowledge of the future.

10 Parameter Ablation

Our model design decisions and weight initializations were chosen based on best practices set forth in the supervised training literature. We are lucky that these also behaved well in the unsupervised setting. Within unsupervised structure prediction, to our best knowledge, there has been no empirical study on neural network architecture design and weight initialization. We therefore provide an initial overview on the topic for several of our decisions.

Weight Initialization If we run our best model (NHMM+Conv+LSTM) with all the weights initialized from a uniform distribution $\mathcal{U}(-10^{-4}, 10^{-4})$[3] we find a dramatic drop in V-Measure performance (61.7 vs 71.7 in Table 3). This is consistent with the common wisdom that unlike supervised learning (Luong et al., 2015), weight initialization is important to achieve good performance on unsupervised tasks. It is possible that performance could be further enhance via the popular technique of ensembling, would would allow for combining models which converged to different local optima.

LSTM Layers And Dropout We find that dropout is important in training an unsupervised NHMM. Removing dropout causes performance to drop six points. To avoid tuning the dropout rate, future work might investigate the effect of variational dropout (Kingma et al., 2015) in unsupervised learning. We also observed that the number of LSTM layers has an impact on V-Measure. Had we simply used a single layer we would have lost nearly five points. It is possible that more layers, perhaps coupled with more data, would yield even greater gains.

11 Future Work

In addition to parameter tuning and multilingual evaluation, the biggest open questions for our approach are the effects of additional data and augmenting the loss function. Neural networks are notoriously data hungry, indicating that while we achieve competitive results, it is possible our model will scale well when run with large corpora. This would likely require the use of techniques like NCE (Gutmann and Hyvärinen, 2010) which have been shown to be highly effective in related tasks like neural language modeling (Mnih and Teh, 2012; Vaswani et al., 2013). Secondly, despite focusing on ways to augment an HMM, Brown clustering and systems inspired by it perform very well. They aim to maximize mutual information rather than likelihood. It is possible that augmenting or constraining our loss will yield additional performance gains.

Outside of simply maximizing performance on tag induction, a more subtle, but powerful contribution of this work may be its demonstration of the easy and effective nature of using neural networks with Bayesian models traditionally trained by EM. We hope this approach scales well to many other domains and tasks.

Acknowledgments

This work was supported by Contracts W911NF-15-1-0543 and HR0011-15-C-0115 with the US Defense Advanced Research Projects Agency (DARPA) and the Army Research Office (ARO). Additional thanks to Christos Christodoulopoulos.

References

Yoshua Bengio, Renato De Mori, Flammia Giovanni, and Ralf Kompe. 1991. Global optimization of a neural

[3] We choose small standard derivation here for numerical stability when computing forward-backward messages.

network - hidden markov model hybrid. In *Proceedings of the International Joint Conference on Neural Networks*, Seattle, WA.

Taylor Berg-Kirkpatrick and Dan Klein. 2010. Phylogenetic grammar induction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1288–1297, Uppsala, Sweden, July.

Taylor Berg-Kirkpatrick, Alexandre Bouchard-Côté, John DeNero, and Dan Klein. 2010. Painless unsupervised learning with features. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 582–590.

Yonatan Bisk, Christos Christodoulopoulos, and Julia Hockenmaier. 2015. Labeled grammar induction with minimal supervision. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 870–876, Beijing, China, July.

Phil Blunsom and Trevor Cohn. 2011. A Hierarchical Pitman-Yor Process HMM for Unsupervised Part of Speech Induction. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 865–874, Portland, Oregon, USA, June.

Peter F Brown, Peter V deSouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. 1992. Class-Based n-gram Models of Natural Language. *Computational Linguistics*, 18.

Christos Christodoulopoulos, Sharon Goldwater, and Mark Steedman. 2010. Two Decades of Unsupervised POS induction: How far have we come? In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, Cambridge, MA, October.

Christos Christodoulopoulos, Sharon Goldwater, and Mark Steedman. 2011. A Bayesian Mixture Model for Part-of-Speech Induction Using Multiple Features. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, Edinburgh, Scotland, UK., July.

Shay B. Cohen, Dipanjan Das, and Noah A. Smith. 2011. Unsupervised structure prediction with non-parallel multilingual guidance. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 50–61, Edinburgh, Scotland, UK., July.

Dipanjan Das and Slav Petrov. 2011. Unsupervised part-of-speech tagging with bilingual graph-based projections. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 600–609, Portland, Oregon, USA, June.

A Dempster, N Laird, and D Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, January.

Michael Gutmann and Aapo Hyvärinen. 2010. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *International Conference on Artificial Intelligence and Statistics*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780, November.

Matthew J Johnson, David Duvenaud, Alexander B Wiltschko, Sandeep R Datta, and Ryan P Adams. 2016. Composing graphical models with neural networks for structured representations and fast inference. *ArXiv e-prints*, March.

Mark Johnson. 2007. Why doesn't EM find good HMM POS-taggers. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, January.

Rafal Józefowicz, Wojciech Zaremba, and Ilya Sutskever. 2015. An empirical exploration of recurrent network architectures. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pages 2342–2350.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the Limits of Language Modeling. *ArXiv e-prints*, February.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. *AAAI*.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *The International Conference on Learning Representations (ICLR)*.

Diederik P Kingma and Max Welling. 2014. Auto-encoding variational bayes. *The International Conference on Learning Representations (ICLR)*.

Diederik P Kingma, Tim Salimans, and Max Welling. 2015. Variational dropout and the local reparameterization trick. In *Advances in Neural Information Processing Systems 28*, pages 2575–2583. Curran Associates, Inc.

Chu-Cheng Lin, Waleed Ammar, Chris Dyer, and Lori Levin. 2015. Unsupervised pos induction with word embeddings. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1311–1316, Denver, Colorado, May–June.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the*

2015 Conference on Empirical Methods in Natural Language Processing, pages 1412–1421, Lisbon, Portugal, September.

Mitchell P Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The Penn Treebank: Annotating Predicate Argument Structure. In *ARPA Human Language Technology Workshop*.

Andriy Mnih and Yee Whye Teh. 2012. A fast and simple algorithm for training neural probabilistic language models. In *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, pages 1751–1758, New York, NY, USA, July.

Vu Pham, Christopher Bluche, Théodore Kermorvant, and Jérôme Louradour. 2014. Dropout improves recurrent neural networks for handwriting recognition. In *International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 285–290, Sept.

Pushpendre Rastogi, Ryan Cotterell, and Jason Eisner. 2016. Weighting finite-state transductions with neural context. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 623–633, San Diego, California, June.

Ruslan Salakhutdinov, Sam Roweis, and Zoubin Ghahramani. 2003. Optimization with em and expectation-conjugate-gradient. In *Proceedings, Intl. Conf. on Machine Learning (ICML*, pages 672–679.

Theerawat Songyot and David Chiang. 2014. Improving word alignment using word similarity. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1840–1845, Doha, Qatar, October.

Valentin I. Spitkovsky, Hiyan Alshawi, Angel X. Chang, and Daniel Jurafsky. 2011. Unsupervised dependency parsing without gold part-of-speech tags. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1281–1290, Edinburgh, Scotland, UK., July.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *JMLR*, (1):1929–1958, January.

Ke Tran, Arianna Bisazza, and Christof Monz. 2016. Recurrent memory networks for language modeling. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 321–331, San Diego, California, June.

Ashish Vaswani, Yinggong Zhao, Victoria Fossum, and David Chiang. 2013. Decoding with large-scale neural language models improves translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1387–1392, Seattle, Washington, USA, October.

Lloyd R Welch. 2003. Hidden Markov Models and the Baum-Welch Algorithm. *IEEE Information Theory Society Newsletter*, 53(4):1–24, December.

Mehmet Ali Yatbaz, Enis Sert, and Deniz Yuret. 2012. Learning Syntactic Categories Using Paradigmatic Representations of Word Context. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 940–951, Jeju Island, Korea, July.