Instructions for EMNLP 2020 Proceedings

Anonymous EMNLP submission

Abstract

todo

1 Introduction

todo

2 Related work

Practitioners have used neural models to improve language modeling performance by utilizing architectures that capture long-range dependencies, moving from feedforward architectures (Bengio et al., 2003) to recurrent neural networks (Mikolov et al., 2010; Zaremba et al., 2014; Merity et al., 2017) as well as transformer-based architectures (Radford et al., 2019).

A recent line of work has investigated the inclusion of latent variables in language models, opting to either introduce a latent variable at the sequence (Bowman et al., 2015) or element level (Chung et al., 2015), with the aim of learning interpretable latent spaces. However, the inclusion of latent variables complicates training and the performance of latent variable models has not been able to match that of fully observed models.

To make matters worse, exact inference itself is intractable in many latent variable sequence models, especially those with large state spaces. We therefore limit our focus to models that permit exact inference, and in particular models with discrete latent variables.

Recent work has demonstrated improvements in tractable discrete latent variable models, such as hidden Markov models (HMMs) and probabilistic context free grammars, through neural parameterizations of conditional distributions (Tran et al., 2016; Kim et al., 2019). In the case of context free grammars, Kim et al. (2019) found that incorporating a compound mixture led to performance improvements, which allowed for potentially infinitely many states while sacrificing exact inference.

An alternative method for increasing the size of the state space which also retains tractability of exact inference was presented in experiments by Dedieu et al. (2019), which introduced a sparsity constraint in the emission distribution of an HMM. The constraint they applied allowed each state to emit only a single element, which allowed them to train a 30k state HMM on character-level language modeling. This particular constraint is problematic for language modeling at the word level, as the size of the vocabulary for a given corpus may be greater than 30k, leaving only a single state per word.

In this work, we build off of Dedieu et al. (2019) to examine whether HMMs with tens of thousands of states are effective language models. We demonstrate that TBD.

Other investigations into improving the performance of HMMs involve relaxing independence or modeling assumptions (Buys et al., 2018) to resemble a recurrent neural network, or through model combination with a recurrent neural network (Krakovna and Doshi-Velez, 2016).

3 Background

We focus on language modeling, where we learn a distribution over sequences of tokens at the word level $\mathbf{x} = \{x_0, \dots, x_T\}$, with each token x_t an element of the finite vocabulary $\mathcal{X}$.

3.1 Hidden Markov Models

Hidden Markov Models (HMMs) specify a joint distribution over the observed words $\mathbf{x}$ and discrete latent states $\mathbf{z} = \{z_0, \dots, z_T\}$, each z_t from finite set $\mathcal{Z}$, with the following generative process: For every time step $t \in \{0, \dots, T\}$, choose a state given the previous state $z_t \mid z_{t-1}$ from the transition distribution $p(z_t \mid z_{t-1})$, where the first state z_0 is chosen from the starting distribution $p(z_0)$. Then choose a word given the current state $x_t \mid z_t$ from the emission distribution $p(x_t \mid z_t)$. This yields the joint distribution

$$p(\mathbf{x}, \mathbf{z}) = p(z_0)p(x_0 \mid z_0) \prod_{t=1}^{T} p(x_t \mid z_t)p(z_t \mid z_{t-1}) \tag{1}$$

The distributions are parameterized as follows

$$\begin{aligned} p(z_0) &\propto e^{\phi_{z_0}} \\ p(z_t \mid z_{t-1}) &\propto e^{\phi_{z_t z_{t-1}}} \\ p(x_t \mid z_t) &\propto e^{\phi_{x_t z_t}} \end{aligned} \tag{2}$$

where each $\phi_{z_0}, \phi_{z_t z_{t-1}}, \phi_{x_t z_t} \in \mathbb{R}$ may be parameterized by a scalar or a neural network.

A scalar parameterization would result in $O(|\mathcal{Z}|^2 + |\mathcal{Z}||\mathcal{X}|)$ parameters, since the transition and emission matrices must be explicitly represented. A compact neural parameterization may take as few as $O(H(|\mathcal{Z}| + |\mathcal{X}|))$ parameters, where H is the dimension of the state and word embeddings. Such a parameterization represents the states and words with embeddings, and implicitly parameterizes the transition and emission distributions by modeling those matrices as a function of the embeddings. We use a residual network as in Kim et al. (2019) to parameterize conditional distributions.

4 Model

For language modeling with an HMM, we are interested in the distribution over the observed words obtained by marginalizing over the latent states $p(\mathbf{x}) = \sum_{\mathbf{z}} p(\mathbf{x}, \mathbf{z})$. This sum can be computed in time $O(T|\mathcal{Z}|^2)$ via dynamic programming, which becomes prohibitive if the number of latent states $|\mathcal{Z}|$ is large. However, in order to increase the capacity of HMMs, we must consider large $\mathcal{Z}$. We outline several techniques to manage the computational complexity of marginalization in HMMs.

4.1 Sparse Emission HMMs

We introduce a sparse emission constraint that allows us to efficiently perform conditional inference in large state spaces. Inspired by Cloned HMMs (Dedieu et al., 2019), we constrain each word x to be emit only by $z \in \mathcal{C}_x \subset \mathcal{Z}$:

$$p(x \mid z) \propto 1(z \in \mathcal{C}_x) e^{\phi_{zx}} \tag{3}$$

This allows us to perform marginalization as follows

$$\begin{aligned} &p(\mathbf{x}) \\ &= \sum_{z_0} p(z_0)p(x_0 \mid z_0) \sum_{z_1} p(z_1 \mid z_0)p(x_1 \mid z_1) \\ &\quad \cdots \sum_{z_T} p(z_T \mid z_{T-1})p(x_T \mid z_T) \\ &= \sum_{z_0 \in \mathcal{C}_{x_0}} p(z_0)p(x_0 \mid z_0) \sum_{z_1 \in \mathcal{C}_{x_1}} p(z_1 \mid z_0)p(x_1 \mid z_1) \\ &\quad \cdots \sum_{z_T \in \mathcal{C}_{x_T}} p(z_T \mid z_{T-1})p(x_T \mid z_T) \end{aligned} \tag{4}$$

which takes $O(T \max_x(|\mathcal{C}_x|)^2)$ computation. We choose sets $\mathcal{C}_x$ such that $\forall x, |\mathcal{C}_x| = k$ due to convenience of implementation.

We experiment with two constraint sets, both of which are specified as hyperparameters and remain fixed for a given model. The first constraint set is obtained by sampling subsets of states for each word i.i.d. from a uniform distribution over subsets. The second constraint set uses Brown Clusters (Brown et al., 1992) to assign all words in a given Brown cluster the same subset of states. This constraint set is advantageous, as it admits further optimizations that we discuss below.

4.2 State Dropout

We introduce a variant of dropout called state dropout that operates on the start and transition distributions. The goal of state dropout is to encourage usage of the full state space in HMMs as well as to reduce the complexity of marginalization in a manner identical to the sparse emission constraints.

State dropout samples a mask over states $\mathbf{b}_{\mathcal{C}} \in \{0,1\}^{|\mathcal{C}|}$ for each partition $\mathcal{C} \subset \mathcal{Z}$, such that each sampled $|\mathbf{b}_{\mathcal{C}}| = n$. We only apply this if the $\mathcal{C}_x$ are disjoint partitions, as with the Brown constraint set.

Let $\mathbf{b}$ be the concatenation of the $\mathbf{b}_{\mathcal{C}}$ for all partitions $\mathcal{C}$, such that $b_z = 1$ if $b_{\mathcal{C}z} = 1$, where $z \in \mathcal{C}$. The start and transition distributions with dropout are given by

$$\begin{aligned} p(z_0) &\propto b_{z_0} e^{\phi_{z_0}} \\ p(z_t \mid z_{t-1}) &\propto b_{z_t} b_{z_{t-1}} e^{\phi_{z_t z_{t-1}}} \end{aligned} \tag{5}$$

Marginalizing over $\mathbf{z}$ with state dropout requires $O(Tn^2)$ computation, where n is the number of nonzero elements of $\mathbf{b}_\mathcal{C}, \forall \mathcal{C}$. We utilize state dropout on top of the Brown emission sparsity constraint, subsampling states of size n for each partition during training.

We also consider unstructured dropout with rate p which samples elements of masks $b_{z_0}, b_{z_t z_{t-1}} \overset{iid}{\sim}$ Bern$(1-p)$ for all $z_0, z_t, z_{t-1} \in \mathcal{Z}$, yielding

$$\begin{aligned} p(z_0) &\propto b_{z_0} e^{\phi_{z_0}} \\ p(z_t \mid z_{t-1}) &\propto b_{z_t z_{t-1}} e^{\phi_{z_t z_{t-1}}}. \end{aligned} \tag{6}$$

Unstructured dropout results in sparsity patterns that are more difficult to take advantage of than state dropout, since there is variance in the number of nonzero elements in a mask. (Fix wording)

We find that models with state dropout generalize better than models with unstructured dropout, and additionally have computational benefits as described above.

4.3 Learning

We optimize the evidence of observed sentences $\log p(\mathbf{x}) = \log \sum_\mathbf{z} p(\mathbf{x}, \mathbf{z})$ by first marginalizing over latent states $\mathbf{z}$ then performing gradient ascent. The emission sparsity constraints we introduce as well as state dropout allow us to both perform marginalization and compute the gradient of the evidence efficiently.

5 Experiments

We evaluate the HMM on two language modeling datasets.

5.1 Language Modeling

Datasets The two datasets we used are the `Penn Treebank` (Marcus et al., 1993) and `wikitext2` (Merity et al., 2016). `Penn Treebank` contains 929k tokens in the training corpus, with a vocabulary size of 10k. We use the preprocessing from Mikolov et al. (2011), which lowercases all words and substitutes words outside of the vocabulary with unks. `Wikitext2` contains 2M tokens in the training corpus, with a vocabulary size of 33k. Casing is preserved, and all words outside the vocab are unked. Both datasets contain inter-sentence dependencies, due to the use of documents consisting of more than a single sentence.

Implementation We train two-layer LSTM recurrent neural networks with 256 or 512 units, as well as two-layer feed-forward neural networks with 256 or 512 units. The HMMs we train follow the sparsity constraints outlined in the previous section with a dropout rate of 0.5, and we vary the total number of states as well as states per word. We optimize all models with AdamW (Loshchilov and Hutter, 2017).

We experimented with a couple batching strategies: On `Penn Treebank`, the first strategy discarded the inter-sentence dependencies and shuffled all sentences, and the second treated the corpus as a single flat document without shuffling. On `Wikitext2`, we either shuffled at the document level or treated the corpus as a single document. Prior work on both corpuses treated the corpora as single documents.

See Appendix A for the hyperparameters for all models.

5.2 Results

We report perplexities for `Penn Treebank` in Table 1 and for `wikitext2` in Table 2.

The HMMs in all cases outperform the feedforward models (FF), but underperform the recurrent LSTMs. Since the HMMs require parameters linear in the number of hidden states, we find that the performance of the HMMs scales poorly compared to the other models which only require parameters that scale linearly with the vocbaulary size. Although explicitly representing and summing over the hidden states allows us to explicitly capture uncertainty in the hidden state, it proves to be a limiting factor in terms of performance.

In the remainder of this section we ablate and analyze the HMMs on `Penn Treebank`.

Sparse emission constraint ablation We ablate the emission sparsity constraints in Table 3, and find that the Brown emission constraints outperforms the uniform emission constraints in all model sizes.

One explanation for the relative benefit of Brown emission constraints over uniform is due to the effect of Brown clusters. The goal of Brown clustering is to place two words in the same cluster if they are used in the same context. In Table 4, we observe that the entropy of the emission distribution for models with uniform emission constraints is lower than models with brown constraints, and

Table 1: Perplexities on the `Penn Treebank` dataset. The FF model is a 256-dim 2-layer feedforward neural network with a window of 4 previous tokens with 0.3 dropout. The LSTM is a 256-dim 2-layer recurrent neural network with 0.3 dropout. The HMM is a 64k-state HMM with 0.5 state dropout. The unshuffled batching strategy was used for all models in this table.

Model	Num Params	Valid PPL	Test PPL
FF	2.8M	164	-
LSTM	3.6M	97	-
HMM	140.9M	125	-

Table 2: Perplexities on the `wikitext2` dataset. The FF model is a 256-dim 2-layer feedforward neural network with a window of 4 previous tokens with 0.3 dropout. The LSTM is a 256-dim 2-layer recurrent neural network with 0.3 dropout. The HMM is a 32k-state HMM with 0.5 state dropout.

Model	Num params	Valid PPL	Test PPL
FF		209	-
LSTM		125	-
HMM		167	-

Table 3: The perplexities for the different emission sparsity constraints in a 1024 state HMM as well as larger HMMs for which exact inference without sparsity is too expensive. The quantities $|\mathcal{Z}|$ and k are the number of hidden states and the number of states per word respectively. The HMMs with 1024 states do not have any dropout, while the 8k and 16k state HMMs have unstructured dropout at a rate of 0.5.

Constraint	$\|\mathcal{Z}\|$	k	Valid PPL
Uniform	1k	32	-
Brown	1k	32	-
None	1k	1k	-
Uniform	8k	128	150*
Brown	8k	128	142*
Uniform	16k	128	146*
Brown	16k	128	134*

the entropy of the transition distributions is higher. This implies that the burden of modeling ambiguity is pushed onto the transition distribution, since the uniform models are less likely to place words that appear in similar contexts together.

Table 4: The average entropies of the the emission and transition distributions for HMMs with uniform and Brown cluster emission constraints. All models have $k = 128$ states per word and use unstructured dropout with a rate of $p = 0.5$.

Constraint	$\|\mathcal{Z}\|$	H(emit)	H(transition)
Uniform	8k	-	-
Brown	8k	-	-
Uniform	16k	-	-
Brown	16k	-	-

Table 5: State occupancies for the dropout strategies and rates. All models were HMMs with Brown cluster emission constraints, 16k total states, and 128 states per word (and therefore 128 Brown clusters).

Dropout type	p	Valid PPL	Occupancy
Unstructured	0.25	-	-
Unstructured	0.5	-	-
Unstructured	0.75	-	-
State	0.25	-	-
State	0.5	-	-
State	0.75	-	-

Dropout analysis The effect of the different dropout strategies and rates is shown in Table 5. We found that state dropout outperformed unstructured dropout overall. This in addition to the computational benefits afforded by state dropout led us to prefer this strategy.

Unstructured dropout was sensitive to batching strategies. We observed a larger gap between training and validation perplexities with the unshuffled batching strategy, whereas unstructured dropout appeared to be robust to the batching strategy.

Analysis 1: Discuss what happens with just state dropout, without partitioning. (Too much variance in gradient estimator, would require variance reduction. Get numbers or just handwaive? Consider exact inference (in constrained model) this work.)

Number of Brown clusters We next examine the sensitivity of HMMs to the number of Brown clusters in Table 6. We find that models at with 16k or 32k total states are not sensitive to the number of Brown clusters in the range where marginalization is computationally feasible. This contrasts with the observation that the number of Brown clusters influenced performance at 1k total states.

Table 6: The perplexities for HMMs with 16k states and different numbers of Brown clusters for constraining the emission distribution of the HMMs. $|\mathcal{Z}|$ is the total number of hidden states. All models have 0.5 state dropout.

$\|\mathcal{Z}\|$	Num clusters	Valid PPL
16k	32	136
16k	64	137
16k	128	133
16k	256	137

Table 7: The perplexities for HMMs with 128 Brown clusters for constraining the emission distribution of the HMMs. $|\mathcal{Z}|$ is the total number of hidden states. All models have 0.5 state dropout.

$\|\mathcal{Z}\|$	Num clusters	Valid PPL
16k	128	133
32k	256	126
65k	512	124

Weight tying ablation Analysis 1: Discuss performance using different tying methods (found this to not affect performance) and computational savings in terms of number of parameters. Discuss relative parameter inefficiency compared to LSTM / FF.

State size ablation We find that as we increase the number of total states while keeping the Brown clusters fixed, performance improves up until we have 65k states.

Analysis 1: How does state usage change as clusters remain constant but the number of states (and states per word) increases?

Parameterization ablation Analysis 1: Is there a performance difference between neural / scalar parameterization, and is it consistent across state sizes?

Discussion 1: Memory savings when working with state dropout to not instantiate the full matrices during training, which makes working with a neural parameterization beneficial.

5.3 Error analysis

6 Discussion

Acknowledgments

TBD

References

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3(null):1137–1155.

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Józefowicz, and Samy Bengio. 2015. Generating sentences from a continuous space. *CoRR*, abs/1511.06349.

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. *Comput. Linguist.*, 18(4):467–479.

Jan Buys, Yonatan Bisk, and Yejin Choi. 2018. Bridging hmms and rnns through architectural transformations.

Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C. Courville, and Yoshua Bengio. 2015. A recurrent latent variable model for sequential data. *CoRR*, abs/1506.02216.

Antoine Dedieu, Nishad Gothoskar, Scott Swingle, Wolfgang Lehrach, Miguel Lázaro-Gredilla, and Dileep George. 2019. Learning higher-order sequential structure with cloned hmms.

Yoon Kim, Chris Dyer, and Alexander M. Rush. 2019. Compound probabilistic context-free grammars for grammar induction. *CoRR*, abs/1906.10225.

Viktoriya Krakovna and Finale Doshi-Velez. 2016. Increasing the interpretability of recurrent neural networks using hidden markov models.

Ilya Loshchilov and Frank Hutter. 2017. Fixing weight decay regularization in adam. *CoRR*, abs/1711.05101.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2017. Regularizing and optimizing LSTM language models. *CoRR*, abs/1708.02182.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *CoRR*, abs/1609.07843.

Tomas Mikolov, Anoop Deoras, Stefan Kombrink, Lukás Burget, and Jan Cernocký. 2011. Empirical evaluation and combination of advanced language modeling techniques. pages 605–608.

Tomas Mikolov, Martin Karafiát, Lukás Burget, Jan Cernocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. pages 1045–1048.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Ke M. Tran, Yonatan Bisk, Ashish Vaswani, Daniel Marcu, and Kevin Knight. 2016. Unsupervised neural hidden markov models. *CoRR*, abs/1609.09007.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *CoRR*, abs/1409.2329.

A Hyperparameters

LSTM

- 2 layers

B Parameter estimation

We maximize the evidence of the observed sentences $\log p(\mathbf{x}) = \log \sum_{\mathbf{z}} p(\mathbf{x}, \mathbf{z})$ via gradient ascent.

B.1 Gradient of the evidence

Let $\psi_0(z_0, z_1) = \log p(x_0, z_0)$ and $\psi_t(z_t, z_{t+1}) = \log p(x_{t+1}, z_{t+1} \mid z_t)$ for $t \in \{1, \ldots, T-1\}$. Additionally, let $\oplus$ and $\otimes$ be addition and multiplication in the log semiring. After conditioning on the observed $\mathbf{x}$, we can express the evidence as the following:

$$A_x = \log p(\mathbf{x}) = \bigoplus_{\mathbf{z}} \bigotimes_{t=0}^{T-1} \psi_t(z_t, z_{t+1}) \tag{7}$$

where $A_{\mathbf{x}}$ is the clamped log partition function (after observing $\mathbf{x}_{0:T}$).

We show that the gradient of the log partition function with respect to the $\psi_t(z_t, z_{t+1})$ is the first moment (a general result for the cumulant generating function of exponential family distributions). Given the value of the derivative $\frac{\partial A_{\mathbf{x}}}{\partial \psi_t(z_t, z_{t+1})}$, we can then apply the chain rule to compute the total derivative with respect to downstream parameters. For example, the gradient with respect to the transition matrix of the HMM is given by

$$\frac{\partial A_{\mathbf{x}}}{\partial \theta_{ij}} = \sum_t \frac{\partial A_{\mathbf{x}}}{\partial \psi_t(i,j)} \frac{\partial \psi_t(i,j)}{\partial \theta_{ij}}$$

Recall that the derivative of logaddexp ($\oplus$ in the log semiring) is

$$\begin{aligned} \frac{\partial}{\partial x} x \oplus y &= \frac{\partial}{\partial x} \log e^x + e^y \\ &= \frac{e^x}{e^x + e^y} \\ &= \exp(x - (x \oplus y)) \end{aligned} \tag{8}$$

while the derivative of addition ($\otimes$ in the log semiring) is

$$\frac{\partial}{\partial x} x \otimes y = 1 \tag{9}$$

The derivative of the clamped log partition function $A_{\mathbf{x}}$ is given by

$$\begin{aligned} &\frac{\partial A_{\mathbf{x}}}{\partial \psi_t(i,j)} \\ &= \frac{\partial}{\partial \psi_t(i,j)} \bigoplus_{\mathbf{z}} \bigotimes_{t'} \psi_{t'}(z_{t'}, z_{t'+1}) \\ &= \frac{\partial}{\partial \psi_t(i,j)} \Bigg(\Bigg(\bigoplus_{\mathbf{z}: z_t = i, z_{t+1} = j} \bigotimes_{t'} \psi_{t'}(z_{t'}, z_{t'+1}) \Bigg) \\ &\qquad \bigoplus \Bigg(\bigoplus_{\mathbf{z}: z_t \neq i, z_{t+1} \neq j} \bigotimes_{t'} \psi_{t'}(z_{t'}, z_{t'+1}) \Bigg) \Bigg) \\ &= \exp \Bigg(\Bigg(\bigoplus_{\mathbf{z}: z_t = i, z_{t+1} = j} \bigotimes_{t'} \psi_{t'}(z_{t'}, z_{t'+1}) \Bigg) \\ &\qquad - \Bigg(\Bigg(\bigoplus_{\mathbf{z}: z_t = i, z_{t+1} = j} \bigotimes_{t'} \psi_{t'}(z_{t'}, z_{t'+1}) \Bigg) \\ &\qquad\quad \bigoplus \Bigg(\bigoplus_{\mathbf{z}: z_t \neq i, z_{t+1} \neq j} \bigotimes_{t'} \psi_{t'}(z_{t'}, z_{t'+1}) \Bigg) \Bigg) \Bigg) \\ &= \exp \Bigg(\Bigg(\bigoplus_{\mathbf{z}: z_t = i, z_{t+1} = j} \bigotimes_{t'} \psi_{t'}(z_{t'}, z_{t'+1}) \Bigg) - A_{\mathbf{x}} \Bigg) \end{aligned}$$

which is the value at i and j of the edge marginal for z_t, z_{t+1} obtained via the forward-backward algorithm. The second equality is a result of the associativity of $\oplus$; the third equality is a result of Eqn. 8; and the fourth equality from Eqn. 9.

C Supplemental Material