A Appendices

A.1 Hyperparameters

For `Penn Treebank` and `Wikitext-2`, we trained the following baselines: a two layer FF 256-dim 5-gram model and a two layer 256-dim LSTM. The FF model is given by the following:

$$p(w_t \mid \mathbf{w}_{<t}) = W_x \text{ReLU}(\text{Conv}(\mathbf{E}_w(\mathbf{w}_{t-4:t-1}))) \tag{6}$$

where $\mathbf{E}_w$ gives the word embeddings and $W_x \in \mathbb{R}^{|\mathcal{X}| \times h}$ is weight-tied to the embeddings.

For the (5-gram) FF model we use a batch size of 128 and a bptt length of 64, as we found the model needed a larger batch size to achieve decent performance. For the LSTM, we use a batch size of 16 and a BPTT length of 32. For both baseline models we use AdamW (Loshchilov and Hutter, 2017) with a learning rate of 1e-3 and a dropout rate of 0.3 on the activations in the model. Both models use a hidden dimension of 256 throughout. These same hyperparameters were applied on both `Penn Treebank` and `Wikitext-2`.

For the HMMs we use a batch size of 16 and a BPTT length of 32. We use state dropout with $\lambda = 0.5$. We use AdamW (Loshchilov and Hutter, 2017) with a learning rate of 1e-2 for `Penn Treebank`, and a learning rate of 1e-3 for `Wikitext-2`.

All weights are initialized with the Kaiming uniform initialization. The FF model was trained for 100 epochs, while all other models were trained for 50. Validation likelihood was checked 4 times per epoch, and learning rates were decayed by a factor of 4 if the validation performance did not improve after 8 checks.

Hyperparameter search was performed manually, using the best validation perplexity achieved in a run. Bounds:

1. Learning rate $\in \{0.0001, 0.001, 0.01, 0.1\}$
2. Dropout $\lambda \in \{0, 0.25, 0.5, 0.75\}$
3. Hidden dimension $h \in \{128, 256, 512\}$
4. Batch size $\in \{16, 32, 64, 128\}$

Experiments were run on RTX 2080 GPUs.

On `PTB` the FF model takes 3s per epoch, the LSTM 23s, and the VL-HMM 2^{15} 433s. The inference for VL-HMM has not heavily optimized, and uses a kernel produced by TVM (Chen et al., 2018) for computing gradients through marginal inference.

Constraint	$\lvert\mathcal{Z}\rvert$	$\lvert\mathcal{C}_x\rvert$	m	Val PPL
Brown	16384	512	32	137
Brown	16384	256	64	138
Brown	16384	128	128	134
Brown	16384	64	256	136
None	1024	-	-	180
Brown	1024	256	4	182
Brown	1024	128	8	194
Uniform	8192	128	-	150
Brown	8192	128	64	142
Uniform	16384	128	-	146
Brown	16384	128	128	136

Table 3: Emission constraint ablations on `Penn Treebank`.

A.2 HMM Parameterization

Let $\mathbf{E}, \mathbf{D} \in \mathbb{R}^{v \times h}$ be an embedding matrix and a matrix of the same size, where v is the size of the vocab and h the hidden dimension. We use the following residual network as our MLP:

$$\begin{aligned} f_i(\mathbf{E}) &= g_i(\text{ReLU}(\mathbf{E}W_{i1})) \\ g_i(\mathbf{D}) &= \text{LayerNorm}(\text{ReLU}(\mathbf{D}W_{i2}) + \mathbf{D}) \end{aligned} \tag{7}$$

with $i \in \{\text{out}, \text{in}, \text{emit}\}$, $W_{i1}, W_{i2} \in \mathbb{R}^{h \times h}$.

For the factored state embeddings in Sec. 4, we introduce residual networks $f_j, j \in \{o, i, e\}$ to compose block and state embeddings, yielding

$$\begin{aligned} \mathbf{H}_{\text{out}} &= f_{\text{out}}(f_o([\mathbf{E}_m, \mathbf{E}_z])) \\ \mathbf{H}_{\text{in}} &= f_{\text{in}}(f_i([\mathbf{E}_m, \mathbf{E}_z])) \\ \mathbf{H}_{\text{emit}} &= f_{\text{emit}}(f_e([\mathbf{E}_m, \mathbf{E}_z])) \end{aligned} \tag{8}$$

A.3 Emission Constraint Ablation

Tbl. 3 shows the results from emission constraint ablations. For the ablations in this section, we do not use the factored state embedding and instead directly learn embeddings $\mathbf{E}_z \in \mathbb{R}^{|\mathcal{Z}| \times h}$. We examine the effect of factored state embeddings in the next section.

With a VL-NHMM that has $|\mathcal{Z}| = 2^{14}$ states, the model is insensitive to the number of blocks M. However, with fewer states $|\mathcal{Z}| = 2^{10}$ where we are able to able to use fewer blocks to examine whether the block-sparsity of the emission results in a performance loss. With $M = 4$ blocks, the block-sparse HMM matches an unconstrained

HMM with the same number of states. When $M = 8$, the block-sparse model underperforms, implying there may be room for improvement with the larger HMMs that use $M > 8$ blocks.

We additionally compare the blocks induced by Brown clustering with a uniform constraint that samples subsets of states of size n independently and uniformly from $\mathcal{Z}$. This does not admit a partitioning, which makes it difficult to apply state dropout. We therefore zero out half of the logits of the transition matrix randomly before normalization. In the bottom of Tbl. 3, we find that models with uniform constraints are consistently outperformed by models with Brown cluster constraints as measured by validation perplexity. The models with uniform constraints also had poor validation performance despite better training performance, a symptom of overfitting.

These ablations demonstrate that the constraints based on Brown clusters used in this work may not be optimal, motivating future work that learns sparsity structure.

A.4 Factored State Representation Ablation

We examine the effect of factoring state representations into block embeddings and independent state embeddings. Recall that factored state representations were parameterized via

$$\mathbf{H}_{\text{out}}, \mathbf{H}_{\text{in}}, \mathbf{H}_{\text{emit}} = \text{MLP}(\mathbf{E}_m, \mathbf{E}_z)$$

with the block and state embeddings $\mathbf{E}_m, \mathbf{E}_z \in \mathbb{R}^{|\mathcal{Z}| \times h/2}$). We ablate this by comparing to the following independent state representation:

$$\mathbf{H}_{\text{out}}, \mathbf{H}_{\text{in}}, \mathbf{H}_{\text{emit}} = \mathbf{E}'_z$$

with $\mathbf{E}'_{\mathbf{z}} \in \mathbb{R}^{|\mathcal{Z}| \times h}$. The results of the factored state ablation are in Fig. 4,

We find that the performance of independent state embeddings with is similar to a model with factored embeddings, until the number of blocks is ≤ 64.

A.5 Computational Considerations

We reproduce the technique ablation table here in Tbl. 4 for reference. As we remove neural components, the number of parameters increases but the time of the forward pass decreases. This is because generating parameters from a neural network takes strictly more time than having those parameters available, similar to a hierarchical Bayesian model.

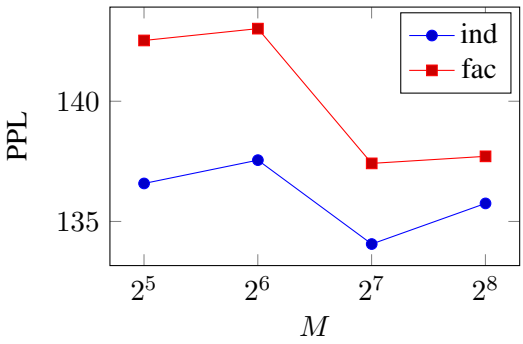

Figure 4: Perplexity on PTB by number of blocks M ($\lambda = 0.5$ and $|\mathcal{Z}| = 2^{14}$).

Model	Size	Train	Val	Time
VL-NHMM (2^{14})	5.6M	122	136	48
- neural param	423M	119	169	14
- dropout	5.6M	89	145	100
- block emb	7.2M	115	134	40

Table 4: Ablations on PTB ($\lambda = 0.5$ and $M = 128$). Time is ms per eval batch (Run on RTX 2080).

When block embeddings are removed and the state representations are directly parameterized, the model is faster due to not needing to recompute state representations. This contrast is even more pronounced when removing neural components altogether, with an almost 3x speedup. However, we note that the drop in generalization is large, indicating the neural parameterization helps with learning. Therefore we use the neural parameterization at training time, then cache the resulting HMM parameters (the transition and emission matrices) for use during inference.