Scaling Hidden Markov Language Models

April 30, 2021

Language Modeling

> *'OpenAI's new language generator GPT-3 is shockingly good—and completely mindless'*
>
> Will Douglas Heaven (MIT Tech Review)

A common paradigm in modern NLP

- Large, opaque models
- Pretrained via language modeling on large data
- Fine-tuned on small data for downstream task

Language modeling either has useful information for or is a necessary component of downstream tasks

Language Modeling

How now, brown _____

- Given the words seen so far, predict the next word
- Language requires encoding long-range context

Language Models

Language models are primarily

- Transformers
  - Encode context into a continuous vector with stacks of attention-based neural networks
- Recurrent neural networks
  - Encode context into a continuous vector with stacks of nonlinear dynamical systems

Both are difficult to interpret due to model complexity

Hidden Markov Language Models

- Interpretability as a design decision
- Context is encoded as a single integer
- Interpretable, but thought to be very poor language models[1]

[1] Buys, Bisk, and Choi, 'Bridging HMMs and RNNs through Architectural Transformations'.

Research Question

To what extent is the performance of HMMs limited by scale and choices in parameterization?

This work: Scale HMMs on language modeling using techniques drawn from recent advances in neural networks

Background: Hidden Markov Models

Hidden Markov Models (HMMs)

- Classical models for unsupervised per-word tag induction
  - Part-of-speech induction
  - Word alignment for translation
- Admits tractable exact inference
  - Strong conditional independence assumptions
  - Finite set of discrete latent states

HMM State Sizes

Year	Data	Model	States
1989	Phoneme Segmentation	HMM[2]	7
1994	POS	HMM[3]	76
2005	Activity Monitoring	DMC HMM[4]	1k
2006	2D Image Tracking	Convolutional HMM[5]	100k
2009	LM	Split-POS HMM[6]	450
2016	POS	Neural HMM[7]	37
2016	Web Traffic Analysis	FFT HMM[8]	81
2019	Char LM	Cloned HMM[9]	30k

[2] Lee and Hon, 'Speaker-independent phone recognition using hidden Markov models'.

[3] Merialdo, 'Tagging English Text with a Probabilistic Model'.

[4] Siddiqi and Moore, 'Fast Inference and Learning in Large-State-Space HMMs'.

[5] Movellan, Hershey, and Susskind, *Real-Time Video Tracking Using Convolution HMMs*.

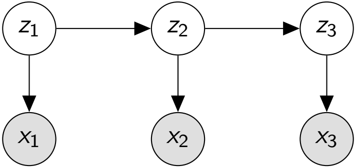
[6] Huang, Eidelman, and Harper, 'Improving A Simple Bigram HMM Part-of-Speech Tagger by Latent Annotation and Self-Training'.

[7] Tran et al., 'Unsupervised Neural Hidden Markov Models'.

[8] Felzenszwalb, Huttenlocher, and Kleinberg, 'Fast Algorithms for Large-State-Space HMMs with Applications to Web Usage Analysis'.

Hidden Markov Models (HMMs)

For times t, model states $z_t \in [Z]$, and tokens $x_t \in [X]$,



This yields the joint distribution

$$p(x, z) = \prod_t p(x_t \mid z_t) p(z_t \mid z_{t-1})$$

with

$$\begin{aligned} &\text{start state} && p(z_1), \\ &\text{transitions} && p(z_t \mid z_{t-1}), \\ &\text{and emissions} && p(x_t \mid z_t) \end{aligned}$$

represented as vectors and matrices

Inference

Given observed $x = (x_1, \ldots, x_T)$ We wish to maximize

$$p(x) = \sum_{z_1} \cdots \sum_{z_T} p(x, z) = \alpha_1^\top \Lambda_2 \Lambda_3 \cdots \Lambda_T \mathbf{1},$$

where we have the

$$\begin{aligned} &\text{start,} & [\alpha_1]_{z_1} &= p(x_1 \mid z_1) p(z_1), \\ &\text{and transition operators,} & [\Lambda_t]_{z_{t-1}, z_t} &= p(x_t \mid z_t) p(z_t \mid z_{t-1}) \end{aligned}$$

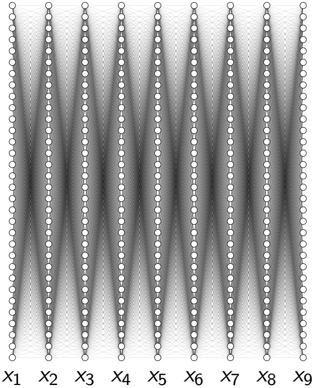
- Matrix representation of forward algorithm:

$$\alpha_t = \alpha_{t-1} \Lambda_t$$

- Requires $O(TZ^2)$ operations in total!

Inference

$$p(x) = \alpha_1^\top \Lambda_2 \cdots \Lambda_T \mathbf{1}$$



- Each node corresponds to a state
- Each edge to an entry in the transition operator matrix

Scaling HMMs

Lessons from Large Neural Language Models

Large models perform better but are . . .

1. Slow to train
2. Prone to overfitting

We must overcome these issues when scaling HMMs

3 Techniques for Training Large HMMs

- Compact neural parameterization
  - ⇧ Generalization
- State dropout
  - ⇧ Speed ⇧ Generalization
- Block-sparse emission constraints
  - ⇧ Speed
- Will cover a fourth in the second part of this talk

Technique 1: Neural Parameterization

- Transition and emission matrices have Z^2 and ZX entries
- More states lead to explosion in parameter count
- Solution: Low dimensional factorization

Neural Parameterization: Softmax Parameterization

$$\boxed{A} \propto \exp\left(\boxed{U} \times \boxed{V^\top}\right)$$

with embeddings $U \in \mathbb{R}^{Z \times D}, V \in \mathbb{R}^{Z \times D}$ or $\mathbb{R}^{X \times D}$

- Can further parameterize U or $V = \mathrm{MLP}(E_u)$
- Similar for emissions

Technique 2: State Dropout

- Dropout is a common technique for regularizing neural networks[10]
  - Reduces a network's reliance on any particular neuron by via random masking
- Extend dropout to the states of an HMM
  - Encourage broad utilization of all states

[10] Srivastava et al., 'Dropout: A Simple Way to Prevent Neural Networks from Overfitting'.

State Dropout

- At each batch, sample dropout mask $\mathbf{b} \in \{0, 1\}^Z$
- Compute distributional parameters by indexing into embeddings U, V

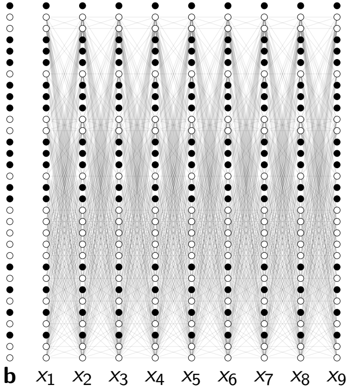
$$\left(\mathbf{b} \circ \boxed{U_{\text{trans}}}\right) \times \left(\mathbf{b} \circ \boxed{V_{\text{trans}}}\right)^\top$$

(a) Unnormalized transition logits

$$\left(\mathbf{b} \circ \boxed{U_{\text{emit}}}\right) \times \boxed{V_{\text{emit}}}^\top$$

(b) Unnormalized emission logits

State Dropout: Inference



- Shaded nodes depict dropped states
- Ignore dropped states during inference

Technique 3: Block-Sparse Emission Constraints

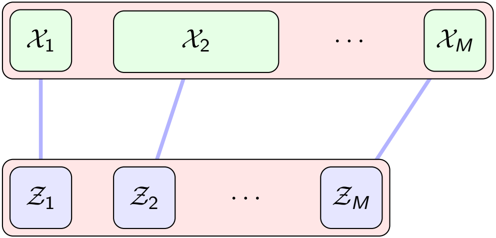
- Reduce cost of marginalization by enforcing structure
- Introduce emission constraints inspired by Cloned HMMs[11]
- Only allow each word to be emit by a subset of states
- Cost of inference is quadratic in the size of the largest subset due to sparsity

[11] Dedieu et al., *Learning higher-order sequential structure with cloned HMMs*.

Block-Sparse Emission Constraints: Alignment

Start with a joint partitioning of both states and words

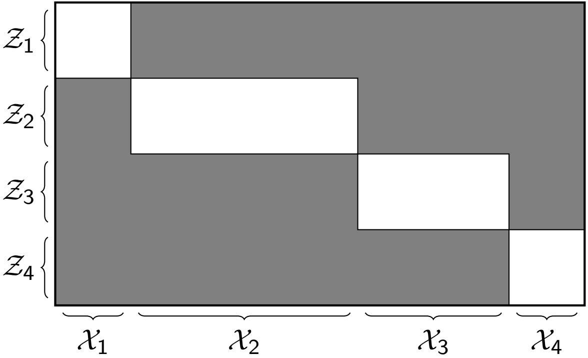
Indices $m \in [M]$ State partitions $\mathcal{Z}_m$ Word partitions $\mathcal{X}_m$



Block-Sparse Emission Constraints

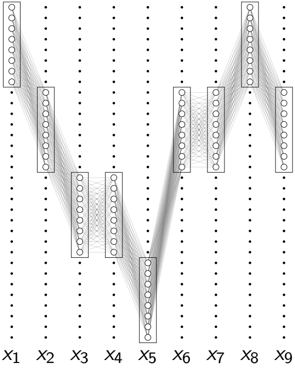
Given the unnormalized emission logits,

- Mask out unaligned state-word entries
- Normalize rows across words in aligned partition

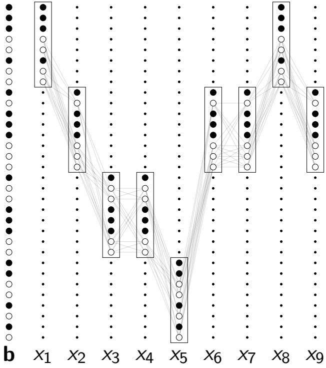


Block-Sparse Emissions: Inference

Given each word x_t, only the states in the correct group can occur



(a) Block-sparse emission



(b) With state dropout

Method Recap

- Compact neural parameterization

  ⬆ Generalization

- State dropout

  ⬆ Speed ⬆ Generalization

- Block-sparse emission constraints

  ⬆ Speed

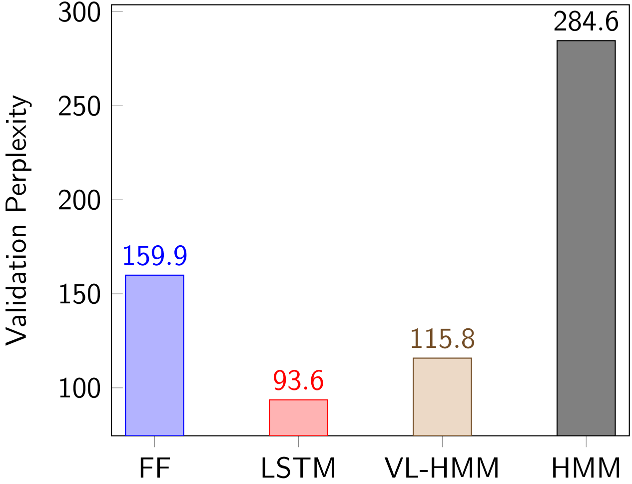
- A fourth after experiments

Experiments

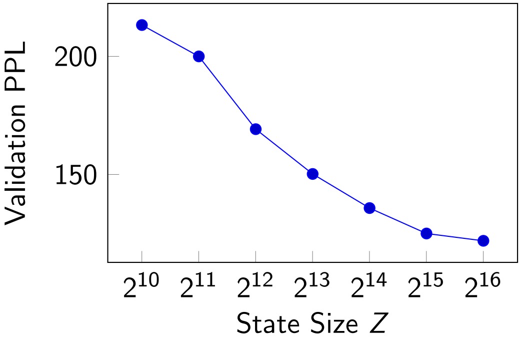
Experiments

- Language modeling on Penn Treebank
- Evaluate perplexity
  - Function of likelihood
  - Lower is better
- Baselines
  - Feedforward 5-gram model
  - 2-layer LSTM
  - A 900 state HMM (Buys et al 2018)
- Model
  - 2^{15} (32k) state very large HMM (VL-HMM)
  - $M = 128$ groups (256 states per type), obtained via Brown Clustering
  - Dropout rate of 0.5 during training

Results on PTB Validation Data



State Size Ablation



Validation perplexity on PTB by state size ($\lambda = 0.5$ and $M = 128$)

Other Ablations

Model	Param	Train	Val
VL-HMM (2^{14})	7.2M	115	134
- neural param	423M	119	169
- state dropout	7.2M	88	157

Discussion

- Greatly scaled the state size of HMMs
- Performance improved with increasing state size
- Still a large gap between RNNs and HMMs
- Does the emission sparsity constraint improve computation complexity at the price of accuracy?

Speeding up HMMs with Low-Rank Decompositions

Fast Inference with Low-Rank Decompositions

- The previous approach relied a pre-specified emission sparsity constraint
- Can we scale inference with a weaker constraint?
- Exploit structure in the transition matrix to speed up inference

Inference

Start by unpacking inference to reveal the most expensive step

$$p(x) = \alpha_1^\top \Lambda_2 \Lambda_3 \cdots \Lambda_T \mathbf{1}$$

with

$$\begin{aligned} &\text{start,} && [\alpha_1]_{z_1} = p(x_1 \mid z_1) p(z_1), \\ &\text{and transition operators,} && [\Lambda_t]_{z_{t-1}, z_t} = p(x_t \mid z_t) p(z_t \mid z_{t-1}) \end{aligned}$$

Inference

Decompose transition operators into transition matrix A and emission matrix O

$$
\begin{aligned}
p(x) &= \alpha_1^\top \Lambda_2 \cdot \Lambda_T \mathbf{1} \\
&= \alpha_1^\top (A \operatorname{diag}([O]_{\cdot, x_2})) \cdots \Lambda_T \mathbf{1} \\
&= \alpha_1^\top A \operatorname{diag}([O]_{\cdot, x_2}) \cdots A \operatorname{diag}([O]_{\cdot, x_T}) \mathbf{1}
\end{aligned}
$$

where the most expensive steps are the matrix-vector products $\alpha_t^\top A$, which take $O(Z^2)$ computation

Fast Matrix-Vector Products

- Goal is to reduce the naive matvec complexity of $O(Z^2)$
- Various methods
  - Sparsity (nnz entries)
  - Fast Fourier Transform ($Z \log Z$)
  - Low-Rank decomposition (ZR)
- We utilize low-rank decompositions

Low-Rank Factorization

Factor transitions $A \in [0,1]^{Z \times Z}$ into product of $U, V \in \mathbb{R}^{Z \times R}$

$$\boxed{\alpha^\top} \times \boxed{A} = \boxed{\alpha^\top} \times \boxed{U} \times \boxed{V^\top}$$

resulting in two matrix-vector products of cost $O(ZR)$ each

- Constraint: Entries of A must be nonnegative
- Solution: Use a nonnegative matrix factorization (NMF)

$$A = \phi(U)\phi(V)^\top,$$

with $\phi : \mathbb{R}^{Z \times R} \to \mathbb{R}_+^{Z \times R}$

Method Recap

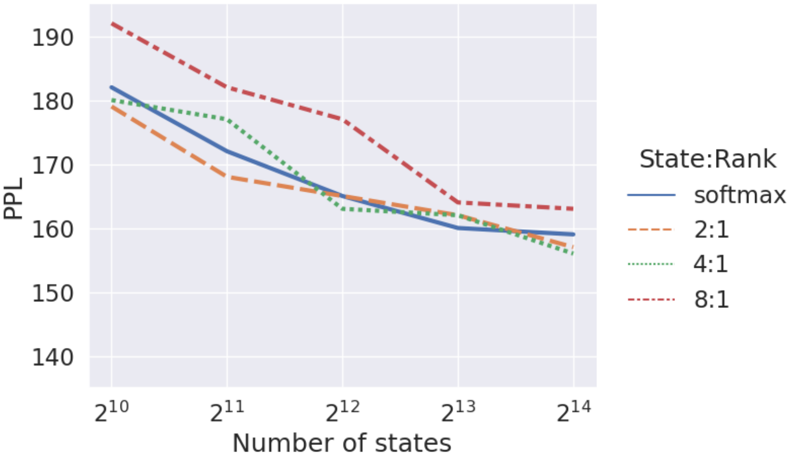
- Target key $O(Z^2)$ matvec step in inference
- Use NMF to reduce cost to $O(ZR)$
- How small can R be relative to Z without sacrificing accuracy?

Experiments

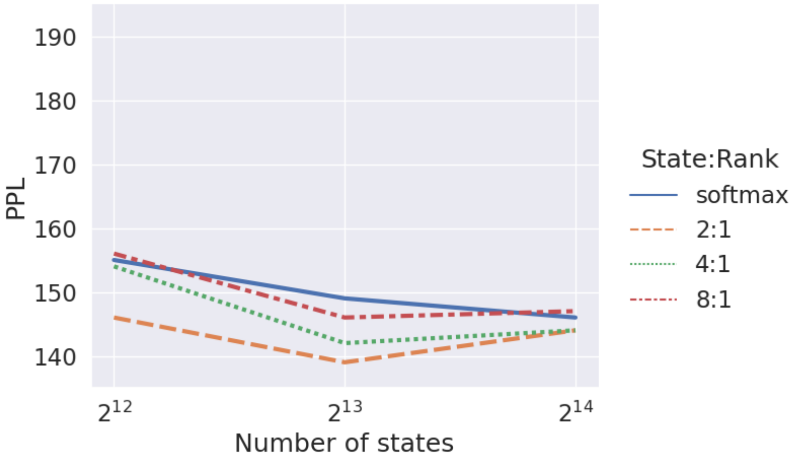
Experiments

- Language modeling on PTB
- Feature map $\phi(U) = \exp(UW)$, with learned $W \in \mathbb{R}^{R \times R}$
- No sparsity constraints

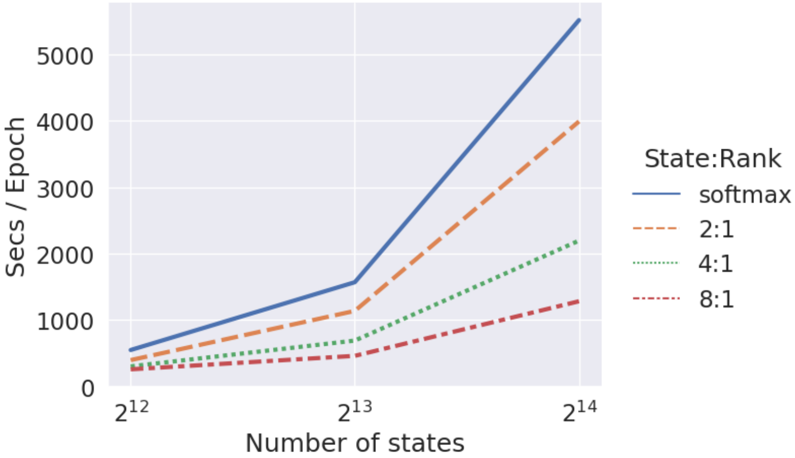
Scaling on PTB (Validation)



Further Scaling on PTB with Dropout (Validation)



Speed Comparison[12]



[12] 16k state VL-HMM takes 2100 s/epoch

Future Work

- Extend to models with intractable exact inference
- Examine the tradeoffs between

Conclusion (TODO)

- Hopeful that HMMs can be competitive language models
- Introduced 4 techniques for tackling speed and overfitting
- Future work will extend to other discrete latent variable models

EOS

Citations

Buys, Jan, Yonatan Bisk, and Yejin Choi. 'Bridging HMMs and RNNs through Architectural Transformations'. In: 2018.

Dedieu, Antoine et al. *Learning higher-order sequential structure with cloned HMMs*. 2019. arXiv: 1905.00507 [stat.ML].

Felzenszwalb, Pedro, Daniel Huttenlocher, and Jon Kleinberg. 'Fast Algorithms for Large-State-Space HMMs with Applications to Web Usage Analysis'. In: *Advances in Neural Information Processing Systems*. Ed. by S. Thrun, L. Saul, and B. Schölkopf. Vol. 16. MIT Press, 2004. URL: https://proceedings.neurips.cc/paper/2003/file/9407c826d8e3c07ad37cb2d13d1cb641-Paper.pdf.

Huang, Zhongqiang, Vladimir Eidelman, and Mary Harper. 'Improving A Simple Bigram HMM Part-of-Speech Tagger by Latent Annotation and Self-Training'. In: (June 2009), pp. 213–216. URL: https://www.aclweb.org/anthology/N09-2054.

Lee, K.-F. and H.-W. Hon. 'Speaker-independent phone recognition using hidden Markov models'. In: *IEEE Transactions*

Generalized Softmax

- Softmax

$$p(z_t \mid z_{t-1}) = \frac{\exp(\mathbf{u}_{z_{t-1}}^\top \mathbf{v}_{z_t})}{\sum_z \exp(\mathbf{u}_{z_{t-1}}^\top \mathbf{v}_z)}$$

- Generalized Softmax

$$p(z_t \mid z_{t-1}) = \frac{K(\mathbf{u}, \mathbf{v})}{\sum_z K(\mathbf{u}, \mathbf{v}_z)} = \frac{\phi(\mathbf{u})^\top \phi(\mathbf{v})}{\sum_z \phi(\mathbf{u})^\top \phi(\mathbf{v}_z)},$$

for positive kernel $K : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}_+$ and feature map $\phi : \mathbb{R}^D \to \mathbb{R}^R$

Generalized Softmax: Inference

- The key $O(Z^2)$ step in the forward algorithm:

$$p(z_t \mid x_{<t}) = \sum_{z_{t-1}} p(z_t \mid z_{t-1}) p(z_{t-1} \mid x_{<t})$$

- In matrix form,

$$\gamma_t = \underbrace{\boldsymbol{\alpha}_{t-1}}_{\mathbb{R}^Z} \underbrace{\Lambda}_{\mathbb{R}^{Z \times Z}},$$

where we have the probability of the

current state,	$[\gamma_t]_{z_t} = p(z_t \mid x_{<t}),$
last state,	$[\boldsymbol{\alpha}_{t-1}]_{z_{t-1}} = p(z_{t-1} \mid x_{<t}),$
transition probability,	$[\Lambda]_{z_{t-1}, z_t} = p(z_t \mid z_{t-1})$

Generalized Softmax: Inference

- Use generalized softmax in transition distribution

$$[\Lambda]_{z_{t-1},z_t} = p(z_t \mid z_{t-1}) \propto \phi(\mathbf{u}_{z_{t-1}})^\top \phi(\mathbf{v}_{z_t})$$

- Allows us to apply associative property of matrix multiplication

$$\begin{aligned}
\boldsymbol{\gamma}_t &= \boldsymbol{\alpha}_{t-1}\Lambda \\
&= \boldsymbol{\alpha}_{t-1}(\mathrm{diag}(d)\phi(U)\phi(V)^\top) \\
&= \underbrace{(\boldsymbol{\alpha}_{t-1} \circ d)}_{\mathbb{R}^Z}\underbrace{\phi(U)}_{\mathbb{R}^{Z\times f}}\underbrace{\phi(V)^\top}_{\mathbb{R}^{f\times Z}},
\end{aligned}$$

  with stacked embeddings $\phi(U), \phi(V) = [\phi(\mathbf{v}_1), \ldots, \phi(\mathbf{v}_Z)]$ and normalizing constants d

- Takes $O(Zf)$ time from left to right!