AWD-LSTM and MoS

Model	#Params	Dev PPL	Test PPL
Inan et al. (2016) – Tied Variational LSTM + augmented loss	24M	75.7	73.2
Zilly et al. (2016) – Variational RHN	23M	67.9	65.4
Zoph & Le (2016) – NAS Cell	25M	-	64.0
Merity et al. (2017) – AWD-LSTM	24M	60.7	58.8
Pham et al. (2018) – Efficient NAS	24M	60.8	58.6
Liu et al. (2018) – Differentiable NAS	23M	58.3	56.1
Yang et al. (2017) – AWD-LSTM-MoS	22M	58.08	55.97
Melis et al. (2018) – 2-layer skip-LSTM + dropout tuning	24M	57.1	55.3
Ours – Transformer-XL	24M	**56.72**	**54.52**
Merity et al. (2017) – AWD-LSTM + finetuning†	24M	60.0	57.3
Yang et al. (2017) – AWD-LSTM-MoS + finetuning†	22M	**56.54**	**54.44**

Table 5: Comparison with state-of-the-art results on Penn Treebank. † indicates using two-step finetuning.

Both accepted by ICLR 2018

Regularizing and Optimizing LSTM Language Models

Stephen Merity [1] **Nitish Shirish Keskar** [1] **Richard Socher** [1]

AWD-LSTM = ASGD (Averaged SGD) + Weight Dropped LSTM

Weight-dropped LSTM

- Regular LSTM:

$$
\begin{aligned}
i_t &= \sigma(W^i x_t + U^i h_{t-1}) \\
f_t &= \sigma(W^f x_t + U^f h_{t-1}) \\
o_t &= \sigma(W^o x_t + U^o h_{t-1}) \\
\tilde{c}_t &= \tanh(W^c x_t + U^c h_{t-1}) \\
c_t &= i_t \odot \tilde{c}_t + f_t \odot + \tilde{c}_{t-1} \\
h_t &= o_t \odot \tanh(c_t)
\end{aligned}
$$

- **Weight-dropped**: Perform DropConnect on the hidden-to-hidden weight matrices $[U^i, U^f, U^o, U^c]$

- The same individual dropped weights remain dropped for the entirety of the forward and backward pass

Averaged SGD

- For the specific task of neural language modeling, traditionally SGD without momentum has been found to outperform other algorithms

$$\min_{w} \quad \frac{1}{N}\sum_{i=1}^{N} f_i(w)$$

$$w_{k+1} = w_k - \gamma_k \hat{\nabla} f(w_k)$$

f_i: loss function for i-th data point, w: weights of the network, γ_k is the learning rate at k-th step

- Averaged SGD: Instead of returning the last iterate as the solution, it returns $\frac{1}{(K-T+1)}\sum_{i=T}^{K} w_i$

K: total number of the iteration; T: user-specified averaging trigger

Averaged SGD

- How to determine the value of T?
- Non-monotonically Triggered ASGD (NT- ASGD) is proposed:
  - Triggers the averaging when the validation metric fails to improve for multiple cycles
  - Setting L as iterations in an epoch and $n = 5$ works well

Algorithm 1 Non-monotonically Triggered ASGD (NT-ASGD)

Inputs: Initial point w_0, learning rate γ, logging interval L, non-monotone interval n.

Initialize $k \leftarrow 0$, $t \leftarrow 0$, $T \leftarrow 0$, `logs` $\leftarrow$ `[]`
while stopping criterion not met **do**
  Compute stochastic gradient $\hat{\nabla} f(w_k)$ and take SGD step (1).
  if $\texttt{mod}(k, L) = 0$ and $T = 0$ **then**
    Compute validation perplexity v.
    if $t > n$ and $v > \min_{l \in \{t-n, \cdots, t\}}$ `logs[l]` **then**
      Set $T \leftarrow k$
    end if
    Append v to `logs`
    $t \leftarrow t + 1$
  end if
end while
return $\frac{\sum_{i=T}^{k} w_i}{(k-T+1)}$

Ext-regularization techniques

- **Variable length backpropagation sequences:**
  - Randomly select the sequence length for the forward and backward pass in two steps:
    1. Select the base sequence length to be seq or $\mathrm{seq}/2$ with probability p
    2. Select the sequence length according to $N(\mathrm{seq}, \mathrm{s})$
  - Linearly rescale the learning rate depending on the length of the resulting sequence

Ext-regularization techniques

- **Variational dropout:**
  - DropConnect v.s. Variational dropout:
    - DropConnect is performed on the hidden-to-hidden weight matrices
    - Variational Dropout is performed directly on h_{t-1}
  - Use DropConnect to regularize the hidden-to-hidden transition
  - Use variational dropout for all other dropout operations (e.g. inputs & outputs)
  - Each example within the mini- batch uses a unique dropout mask

Ext-regularization techniques

- **Embedding dropout:**
  - The dropout is broadcast across all the word vector's embedding
  - The remaining embeddings are scaled by $\frac{1}{1-p_e}$

    p_e is the probability of embedding dropout
  - Dropout occurs on the embedding matrix that is used for a full forward and backward pass

Ext-regularization techniques

- **Weight tying:**
  - Share the weights between the embedding and softmax layer
- **Independent embedding size and hidden size:**
  - Reduce the embedding size (and modify the first & last LSTM layers)
- **Activation Regularization and Temporal Activation Regularization:**
  - AR: $\alpha\, L_2(m \odot h_t)$
  - TAR: $\beta\, L_2(h_t - h_{t+1})$

Training Details

- **3-layer LSTM** with hidden size 1150 and embedding size 400
- **Uniformly initialize:** Embedding: $[-0.1, 0.1]$, Others: $[-\frac{1}{\sqrt{H}}, \frac{1}{\sqrt{H}}]$
- **Training:** NT-ASGD for 750 epochs with learning rate decay
- **Fine-tuning:** ASGD, terminates using the same non-monotonic criterion
- **Gradient Clipping:** maximum norm 0.25
- **Initial learning rate:** 30
- **BPTT length:** $N(70, 5)$ with probability 0.95 and $N(35, 5)$ with probability 0.05
- **Dropout** on word vectors, output b/w LSTM layers, final output layer, and embedding: $(0.4, 0.3, 0.4, 0.1)$
- **Weight-dropped:** $0.5/0.65$ for PTB and WT2
- **AR and TAR:** 2 and 1

Ablation Analysis

Model	PTB Validation	PTB Test	WT2 Validation	WT2 Test
AWD-LSTM (tied)	60.0	57.3	68.6	65.8
– fine-tuning	60.7	58.8	69.1	66.0
– NT-ASGD	66.3	63.7	73.3	69.7
– variable sequence lengths	61.3	58.9	69.3	66.2
– embedding dropout	65.1	62.7	71.1	68.1
– weight decay	63.7	61.0	71.9	68.7
– AR/TAR	62.7	60.3	73.2	70.1
– full sized embedding	68.0	65.6	73.7	70.7
– weight-dropping	71.1	68.9	78.4	74.9

Table 4. Model ablations for our best LSTM models reporting results over the validation and test set on Penn Treebank and WikiText-2. Ablations are split into optimization and regularization variants, sorted according to the achieved validation perplexity on WikiText-2.

BREAKING THE SOFTMAX BOTTLENECK: A HIGH-RANK RNN LANGUAGE MODEL

Zhilin Yang*, **Zihang Dai***, **Ruslan Salakhutdinov, William W. Cohen**
School of Computer Science
Carnegie Mellon University
{zhiliny,dzihang,rsalakhu,wcohen}@cs.cmu.edu

Introduction

- **Softmax Bottleneck:**
  - Learning a softmax-based recurrent language model is equivalent to solving a matrix factorization problem
  - Natural language is highly context-dependent, the matrix to be factorized can be high-rank
  - Softmax-based language models with distributed (output) word embeddings do not have enough capacity to model natural language
- **Mixture of Softmax (MoS)** is proposed to solve the problem

LM as Matrix Factorization

- Natural language can have infinite number of contexts, but firstly we consider a finite set of N possible contexts
- Natural language $\mathcal{L} = \{(c_1, P^*(X|c_1)), \cdots, (c_N, P^*(X|c_N))\}$
- Vocabulary: $\{x_1, x_2, \cdots, x_M\}$
- Objective of language model is to learn a distribution: $P_\theta(X|C)$
- Expressiveness of the model: Does there exists a θ such that for all c in L, the equation $P_\theta(X|c) = P^*(X|c)$ holds

LM as Matrix Factorization

- Softmax: $$P_\theta(x|c) = \frac{\exp \mathbf{h}_c^\top \mathbf{w}_x}{\sum_{x'} \exp \mathbf{h}_c^\top \mathbf{w}_{x'}}$$ $\mathbf{h}_c^\top \mathbf{w}_x$ is called a *logit*

- We define 3 matrices:

$$\mathbf{H}_\theta = \begin{bmatrix} \mathbf{h}_{c_1}^\top \\ \mathbf{h}_{c_2}^\top \\ \cdots \\ \mathbf{h}_{c_N}^\top \end{bmatrix}; \ \mathbf{W}_\theta = \begin{bmatrix} \mathbf{w}_{x_1}^\top \\ \mathbf{w}_{x_2}^\top \\ \cdots \\ \mathbf{w}_{x_M}^\top \end{bmatrix}; \ \mathbf{A} = \begin{bmatrix} \log P^*(x_1|c_1), & \log P^*(x_2|c_1) & \cdots & \log P^*(x_M|c_1) \\ \log P^*(x_1|c_2), & \log P^*(x_2|c_2) & \cdots & \log P^*(x_M|c_2) \\ \vdots & \vdots & \ddots & \vdots \\ \log P^*(x_1|c_N), & \log P^*(x_2|c_N) & \cdots & \log P^*(x_M|c_N) \end{bmatrix}$$

$\mathbf{H}_\theta \in \mathbb{R}^{N \times d}$, $\mathbf{W}_\theta \in \mathbb{R}^{M \times d}$, $\mathbf{A} \in \mathbb{R}^{N \times M}$

- Rows of $\mathbf{H}_\theta$, $\mathbf{W}_\theta$, and $\mathbf{A}$: context vectors, word embeddings, and log probabilities of the true data distribution

LM as Matrix Factorization

- A set of matrices formed by applying *row-wise shift* to $\boldsymbol{A}$:

$$F(\mathbf{A}) = \{\mathbf{A} + \mathbf{\Lambda}\mathbf{J}_{N,M} | \mathbf{\Lambda} \text{ is diagonal and } \mathbf{\Lambda} \in \mathbb{R}^{N \times N}\}$$

$$\mathbf{J}_{N,M} \text{ is an all-ones matrix with size } N \times M$$

- The set $F(\mathbf{A})$ have 2 properties:

Property 1. *For any matrix* $\mathbf{A}'$, $\mathbf{A}' \in F(\mathbf{A})$ *if and only if* $Softmax(\mathbf{A}') = P^*$. *In other words,* $F(\mathbf{A})$ *defines the set of* ***all*** *possible logits that correspond to the true data distribution.*

Property 2. *For any* $\mathbf{A}_1 \neq \mathbf{A}_2 \in F(\mathbf{A})$, $|rank(\mathbf{A}_1) - rank(\mathbf{A}_2)| \leq 1$. *In other words, all matrices in* $F(\mathbf{A})$ *have similar ranks, with the maximum rank difference being 1.*

LM as Matrix Factorization

Lemma 1. *Given a model parameter θ, $\mathbf{H}_\theta \mathbf{W}_\theta^\top \in F(\mathbf{A})$ if and only if $P_\theta(X|c) = P^*(X|c)$ for all c in $\mathcal{L}$.*

Now the expressiveness question becomes: does there exist a parameter θ and $\mathbf{A}' \in F(\mathbf{A})$ such that

$$\mathbf{H}_\theta \mathbf{W}_\theta^\top = \mathbf{A}'.$$

- The rank of $\mathbf{H}_\theta \mathbf{W}_\theta^\top$ is strictly upper bounded by embedding size n

Proposition 1. *Given that the function family $\mathcal{U}$ is a universal approximator, there exists a parameter θ such that $P_\theta(X|c) = P^*(X|c)$ for all c in $\mathcal{L}$ if and only if $d \geq \min_{\mathbf{A}' \in F(\mathbf{A})} rank(\mathbf{A}')$.*

Corollary 1. ***(Softmax Bottleneck)*** *If $d < rank(\mathbf{A}) - 1$, for any function family $\mathcal{U}$ and any model parameter θ, there exists a context c in $\mathcal{L}$ such that $P_\theta(X|c) \neq P^*(X|c)$.*

LM as Matrix Factorization

- Hypothesis: Natural language is high-rank:
  - Natural language is highly context-dependent
  - If A is low-rank, it means humans only need a limited number (10^2) of bases, and all semantic meanings can be created by (potentially) negating and (weighted) averaging these bases
  - Empirically, our high-rank language model outperforms conventional low-rank language models

Mixture of Softmax

- **MoS:** $$P_\theta(x|c) = \sum_{k=1}^{K} \pi_{c,k} \frac{\exp \mathbf{h}_{c,k}^\top \mathbf{w}_x}{\sum_{x'} \exp \mathbf{h}_{c,k}^\top \mathbf{w}_{x'}}; \quad \text{s.t.} \sum_{k=1}^{K} \pi_{c,k} = 1$$

- MoS computes K softmax distributions and uses a weighted average of them as the next-token probability distribution

- Computation:

$$(\mathbf{g}_1, \cdots, \mathbf{g}_T)$$ — Hidden states

$$\pi_{c_t,k} = \frac{\exp \mathbf{w}_{\pi,k}^\top \mathbf{g}_t}{\sum_{k'=1}^{K} \exp \mathbf{w}_{\pi,k'}^\top \mathbf{g}_t}$$ — Prior weights

$$\mathbf{h}_{c_t,k} = \tanh(\mathbf{W}_{h,k} \mathbf{g}_t)$$ — Context vector

$\mathbf{w}_{\pi,k}$ and $\mathbf{W}_{h,k}$ are model parameters

Mixture of Softmax

- Expressiveness:

$$\hat{\mathbf{A}}_{\text{MoS}} = \log \sum_{k=1}^{K} \mathbf{\Pi}_k \exp(\mathbf{H}_{\theta,k} \mathbf{W}_\theta^\top)$$

$\mathbf{\Pi}_k$ is an $(N \times N)$ diagonal matrix with elements being the prior $\pi_{c,k}$

  - With non-linear function (log_sum_exp), $\hat{\mathbf{A}}_{\text{MoS}}$ can be arbitrarily high-rank

- Comparable baseline: Mixture of Contexts:

$$P_\theta(x|c) = \frac{\exp\left(\sum_{k=1}^{K} \pi_{c,k} \mathbf{h}_{c,k}\right)^\top \mathbf{w}_x}{\sum_{x'} \exp\left(\sum_{k=1}^{K} \pi_{c,k} \mathbf{h}_{c,k}\right)^\top \mathbf{w}_{x'}} = \frac{\exp\left(\sum_{k=1}^{K} \pi_{c,k} \mathbf{h}_{c,k}^\top \mathbf{w}_x\right)}{\sum_{x'} \exp\left(\sum_{k=1}^{K} \pi_{c,k} \mathbf{h}_{c,k}^\top \mathbf{w}_{x'}\right)}$$

  - Low-rank model with the same number of parameters

Experiments

Model	#Param	Validation	Test
Mikolov & Zweig (2012) – RNN-LDA + KN-5 + cache	9M‡	-	92.0
Zaremba et al. (2014) – LSTM	20M	86.2	82.7
Gal & Ghahramani (2016) – Variational LSTM (MC)	20M	-	78.6
Kim et al. (2016) – CharCNN	19M	-	78.9
Merity et al. (2016) – Pointer Sentinel-LSTM	21M	72.4	70.9
Grave et al. (2016) – LSTM + continuous cache pointer†	-	-	72.1
Inan et al. (2016) – Tied Variational LSTM + augmented loss	24M	75.7	73.2
Zilly et al. (2016) – Variational RHN	23M	67.9	65.4
Zoph & Le (2016) – NAS Cell	25M	-	64.0
Melis et al. (2017) – 2-layer skip connection LSTM	24M	60.9	58.3
Merity et al. (2017) – AWD-LSTM w/o finetune	24M	60.7	58.8
Merity et al. (2017) – AWD-LSTM	24M	60.0	57.3
Ours – AWD-LSTM-MoS w/o finetune	22M	58.08	55.97
Ours – AWD-LSTM-MoS	22M	**56.54**	**54.44**
Merity et al. (2017) – AWD-LSTM + continuous cache pointer†	24M	53.9	52.8
Krause et al. (2017) – AWD-LSTM + dynamic evaluation†	24M	51.6	51.1
Ours – AWD-LSTM-MoS + dynamic evaluation†	22M	**48.33**	**47.69**

Table 1: Single model perplexity on validation and test sets on Penn Treebank. Baseline results are obtained from Merity et al. (2017) and Krause et al. (2017). † indicates using dynamic evaluation.

Experiments

Model	#Param	Validation	Test
Inan et al. (2016) – Variational LSTM + augmented loss	28M	91.5	87.0
Grave et al. (2016) – LSTM + continuous cache pointer†	-	-	68.9
Melis et al. (2017) – 2-layer skip connection LSTM	24M	69.1	65.9
Merity et al. (2017) – AWD-LSTM w/o finetune	33M	69.1	66.0
Merity et al. (2017) – AWD-LSTM	33M	68.6	65.8
Ours – AWD-LSTM-MoS w/o finetune	35M	66.01	63.33
Ours – AWD-LSTM-MoS	35M	**63.88**	**61.45**
Merity et al. (2017) – AWD-LSTM + continuous cache pointer †	33M	53.8	52.0
Krause et al. (2017) – AWD-LSTM + dynamic evaluation†	33M	46.4	44.3
Ours – AWD-LSTM-MoS + dynamical evaluation†	35M	**42.41**	**40.68**

Table 2: Single model perplexity over WikiText-2. Baseline results are obtained from Merity et al. (2017) and Krause et al. (2017). † indicates using dynamic evaluation.

Experiments

- Large dataset:

Model	#Param	Train	Validation	Test
Softmax	119M	41.47	43.86	42.77
MoS	113M	**36.39**	**38.01**	**37.10**

Table 3: Perplexity comparison on 1B word dataset. Train perplexity is the average of the last 4,000 updates.

- Seq2seq:

Model	Perplexity	BLEU-1		BLEU-2		BLEU-3		BLEU-4	
		prec	recall	prec	recall	prec	recall	prec	recall
Seq2Seq-Softmax	34.657	0.249	0.188	0.193	0.151	0.168	0.133	0.141	0.111
Seq2Seq-MoC	33.291	0.259	0.198	0.202	0.159	0.176	0.140	0.148	0.117
Seq2Seq-MoS	**32.727**	**0.272**	**0.206**	**0.213**	**0.166**	**0.185**	**0.146**	**0.157**	**0.123**

Table 4: Evaluation scores on Switchboard.

Ablation Study

Model	PTB Validation	PTB Test	WT2 Validation	WT2 Test
AWD-LSTM-MoS	**58.08**	**55.97**	**66.01**	**63.33**
AWD-LSTM-MoC	59.82	57.55	68.76	65.98
AWD-LSTM (Merity et al. (2017) hyper-parameters)	61.49	58.95	68.73	65.40
AWD-LSTM (MoS hyper-parameters)	78.86	74.86	72.73	69.18

Table 5: Ablation study on Penn Treebank and WikiText-2 without finetuning or dynamical evaluation.

Verify the Rank

- Sample T contexts and stack the T log-probability vectors into $T \times M$ matrix, and then compute the rank:

- Increasing the number of mixture components in MoS increases the rank:

Model	Validation	Test
Softmax	400	400
MoC	280	280
MoS	**9981**	**9981**

Table 6: Rank comparison on PTB. To ensure comparable model sizes, the embedding sizes of Softmax, MoC and MoS are 400, 280, 280 respectively. The vocabulary size, i.e., M, is 10,000 for all models.

#Softmax	Rank	Perplexity
3	6467	58.62
5	8930	57.36
10	9973	56.33
15	9981	55.97
20	9981	56.17

Table 7: Empirical rank and test perplexity on PTB with different number of Softmaxes.