On the Softmax Bottleneck of Recurrent Language Models

Dwarak Govind Parthiban,[1] **Yongyi Mao,** [1] **Diana Inkpen** [1]
[1] University of Ottawa
yottabytt@gmail.com, ymao@uottawa.ca, diana.inkpen@uottawa.ca

Abstract

Recent research has pointed to a limitation of word-level neural language models with softmax outputs. This limitation, known as the "softmax bottleneck" refers to the inability of these models to produce high-rank log probability ($\log P$) matrices. Various solutions have been proposed to break this bottleneck, including Mixture of Softmaxes, SigSoftmax, and Linear Monotonic Softmax with Piecewise Linear Increasing Functions. They were reported to offer better performance in terms of perplexity on test data. A natural perception from these results is a strong positive correlation between the rank of the $\log P$ matrix and the model's performance. In this work, we show via an extensive empirical study that such a correlation is fairly weak and that the high-rank of the $\log P$ matrix is neither necessary nor sufficient for better test perplexity. Although our results are empirical, they are established in part via the construction of a rich family of models, which we call Generalized SigSoftmax. They are able to create diverse ranks for the $\log P$ matrices. We also present an investigation as to why the proposed solutions achieve better performance.

Metrics	Model		
	SS	LMS-PLIF	MoS
rank (reported)	4640	N/A	9980
rank (ours)	4979	580	9979
perplexity difference (reported)	0.40	1.11	2.88
perplexity difference (ours)	-0.06	0.43	1.28

Table 1: Perplexity difference (subtracted from the baseline AWD-LSTM's perplexity; positive means better than the baseline and negative means worse) for SS, LMS-PLIF and MoS models and their respective ranks of the log probability matrix $\log P$ on the test set of PTB. The reported results in their respective original papers is compared with our reproduced results. The rank of the $\log P$ matrix produced by the baseline model that uses the softmax function is 402, as the embedding dimension d is 400. SS and LMS-PLIF use $d = 400$. But MoS uses $d = 280$.

Introduction

This paper is concerned with language models constructed with recurrent neural networks. Suppose that such a language model involves a vocabulary $\mathcal{V}$ and a dictionary of contexts $\mathcal{C}$. The model in effect computes a probability estimate $P_{\mathcal{V}|\mathcal{C}}(v|c)$ of any token $v \in \mathcal{V}$ conditioned on a context $c \in \mathcal{C}$, using a softmax function in its output layer. When such an estimated conditional distribution $P_{\mathcal{V}|\mathcal{C}}(v|c)$ is represented as a $|\mathcal{C}| \times |\mathcal{V}|$ matrix P, Yang et al. (2018) proved that the rank of the $\log P$ cannot be larger than $d+1$ where d is the word embedding dimension. They also hypothesized that the $\log P$ matrix has to be high-rank as a natural language is generally believed to be highly diverse and context-dependent (Mikolov and Zweig 2012). They called the inability of the softmax function to produce high-rank $\log P$ matrices as the "softmax bottleneck". They introduced the "Mixture of Softmaxes" (MoS) layer as a replacement to the output softmax layer that is generally used in language models, to break the softmax bottleneck. They also introduced a variant to MoS called the Mixture of Contexts (MoC) as a baseline, which does not break the softmax bottleneck but has the same parameterization as in MoS. In MoC, the mixing is done for the context vectors rather than the probabilities as in MoS.

Inspired by the work of Yang et al. (2018), there are several works in the recent literature (Kanai et al. 2018; Takase, Suzuki, and Nagata 2018; Yang et al. 2019; Ganea et al. 2019) that propose other approaches to overcome the limitation imposed by the softmax bottleneck, including, for example, "SigSoftmax" (SS) (Kanai et al. 2018) and "Linear Monotonic Softmax with Piecewise Linear Increasing Functions" (LMS-PLIF). Specifically, the SS function is used as a replacement for the softmax function, and the LMS-PLIF is a learnable piecewise linear increasing function that is applied on the logits before they are sent to the softmax function. The successes of SS, LMS-PLIF and MoS seem to have implied that there is a strong correlation between the performance of a language model and the rank of its $\log P$ matrix and that the performance improvements demonstrated in these models are largely attributed to their high ranks.

As all the approaches in existing works were patched on the baseline AWD-LSTM model (Merity, Keskar, and Socher 2018), frequently we will denote a model's name by just mentioning the approach. Hence, we will also mention the baseline AWD-LSTM model as the Softmax model.

The starting point of this work is some puzzling observations we obtained concerning SS, LMS-PLIF, and MoS. In our experiments, we reproduced the results of the baseline AWD-LSTM model and its SS, LMS-PLIF, and MoS counterparts. The models were trained on the Penn Treebank (PTB) dataset. Rank values for $\log P$ matrices computed from all learned models were obtained in addition to the testing perplexities. These results together with the corresponding results reported in the respective papers are shown in Table 1. In this table, we see that a higher rank does not seem to correlate with improved performance in terms of test perplexity. For example, the $\log P$ matrix produced by SS has much higher rank than LMS-PLIF, but it performs visibly worse than LMS-PLIF; the purpose of introducing LMS-PLIF is to produce a high-rank $\log P$ matrix but the rank is not as high as the those from SS and MoS and yet LMS-PLIF performs better than the baseline and SS.

To better understand the relationship between the rank of the $\log P$ matrix and the model's performance and to explain the puzzling observations, this work takes a systematic route of investigation, which we here outline.

The first question we aim to answer is: *is the rank of the* $\log P$ *matrix truly correlated with performance?*

For that purpose, we construct a family of functions which we call Generalized SigSoftmax (GSS), which, when used to replace the softmax function in the baseline language model, is capable of producing $\log P$ matrices with diverse ranks. Via an empirical study using these models, we show that there is only a fairly weak correlation between the rank of the $\log P$ matrix and the test perplexity. We also perform a qualitative analysis on Softmax, SS, GSS, LMS-PLIF, MoS, and MoC models to see how well the model's ability to produce high-rank $\log P$ matrix correlates with its ability to make better context-dependent predictions. For a selected set of contexts, our results suggest that no strong correlation can be concluded. Going beyond the perplexity metric, we perform additional experiments where we use the word embeddings learned from these models in several downstream word similarity tasks. An analysis of the experimental results suggests that a high-rank language model does not necessarily learn word embeddings with a better quality, at least for these word similarity benchmarks.

These results cast serious doubts on the existence of strong correlation between a model's performance and its ability to produce high-rank $\log P$ matrices. But before we daringly reject this possibility, we move on the investigate another related question: *Are the performance improvements (in terms of perplexity) brought by MoS and LMS-PLIF due to the high ranks in their* $\log P$ *matrices?*

As seen in Table 1 and also confirmed in our many additional experiments in this work, the $\log P$ matrix produced by LMS-PLIF in fact has a fairly low rank. Thus we suggest that the performance advantage of LMS-PLIF is not much related to its rank or its ability to "break the softmax bottleneck" as was claimed by the authors. Through an investigation using a delicate experiment, we suggest that the good performance of LMS-PLIF is more likely due to an implicit regularization effect.

To justify that MoS is able to perform better because of breaking the softmax bottleneck, Yang et al. (2018) showed that there is a positive correlation between the rank of the $\log P$ matrices produced by MoS models (using different number of mixtures K) and the performance in terms of test perplexity. They also showed that, till the rank of the $\log P$ matrix becomes full-rank, increasing K helps in increasing the test performance of the model. Through a series of experiments, we show that it is possible to increase the rank of the $\log P$ matrix without increasing K and without a positive correlation with test performance; we speculate that the point after which increasing K does not help is actually due to the overfitting resulting from an increased capacity; and we also show how the hyperparameters used by MoS could have been an advantage for its better performance.

Overall the extensive experimental study performed in this work suggests that the performance of a well-trained model is not necessarily correlated well with the rank of its $\log P$ matrix. Through this study, we conclude that a high rank of the $\log P$ matrix is neither necessary nor sufficient for a language model to perform well. Other factors, such as regularization and hyperparameter tuning, may interact closely with model's capacity and inductive bias, and play an important role in the model's performance.

Experimental Setup for Fair Comparisons

The experimental setup and results that substantiate the claims in this paper are fully mentioned in Supplementary Material (SM) if they are only partially mentioned here. The SM and code can be accessed at https://github.com/yottabytt/awd-lstm-lmkit .

Code and Datasets Most of our implementation is based on the open source code released by the authors of AWD-LSTM and MoS. Following previous works (Yang et al. 2018; Kanai et al. 2018; Ganea et al. 2019), for our language modeling experiments, we use the Penn Treebank (PTB) (Marcus, Santorini, and Marcinkiewicz 1993) and the WikiText-2 (WT2) (Merity et al. 2017) datasets. The vocabulary sizes $N = |\mathcal{V}|$ of PTB and WT2 are 10,000 and 33,278 respectively. To analyze the quality of learned word embeddings from language models on downstream word similarity tasks, we evaluate (Faruqui and Dyer 2014) the learned embeddings on 13 benchmark datasets namely WS-353 (Finkelstein et al. 2001), WS-353-REL (Agirre et al. 2009), WS-353-REL (Finkelstein et al. 2001), RG-65 (Rubenstein and Goodenough 1965), MC-30 (Miller and Charles 1991), MTurk-287 (Radinsky et al. 2011), MTurk-771 (Halawi et al. 2012), MEN (Bruni et al. 2012), YP-130 (Yang and Powers 2006), VERB-143 (Baker, Reichart, and Korhonen 2014), RW-STANFORD (Luong, Socher, and Manning 2013), SimVerb-3500 (Gerz et al. 2016), and SimLex-999 (Hill, Reichart, and Korhonen 2015).

Hyperparameter configuration To train an AWD-LSTM based model, there is a hyperparameter called the non-monotone interval n that is used to switch the optimization algorithm from SGD to Averaged SGD. This approach is called Non-Monotonically Triggered Averaged SGD (NT-ASGD). The authors of the AWD-LSTM reported that $n = 5$ worked the best across both PTB and WT2 datasets. But Takase, Suzuki, and Nagata (2018) showed that n with values other than 5 worked even better. There are other works (Wang, Gong, and Liu 2019; Ganea et al. 2019; Wang et al. 2020) in which the switch is made explicitly after 200th epoch. We call this approach as Epoch Triggered Averaged SGD (ET-ASGD). We empirically observed that ET-ASGD works better than NT-ASGD with $n = 5$ for all existing works that are based upon the AWD-LSTM model. The reason for a consistent difference between the reported and the reproduced reduction in test perplexities as seen in Table 1 is partly because of our use of ET-ASGD for all the models. Hence, we mostly use ET-ASGD for all our experiments. We use NT-ASGD only for the purpose of investigating the claims made in the literature which are based out of the results obtained using NT-ASGD. We emphasize that for a fair comparison between any two models, both should use either NT-ASGD (same n) or ET-ASGD (same epoch number) during training. For rest of the hyperparameters that are common to all models, we use the same values as that of the baseline AWD-LSTM model, except for MoS and MoC, for which we use the values used by the authors of MoS (Yang et al. 2018), because there is a significant number of hyperparameters whose values are different when compared to those used by the authors of AWD-LSTM. Though it is not fair, changing all of them in one shot could turn out to be disadvantageous to MoS and MoC. But we will be addressing this difference in more detail and its potential impact on model's performance later in this work.

Rank calculation The learned language model is evaluated on the test set to construct the $\log P$ matrices. For all contexts $c \in \mathcal{C}$ in the test set, the log of conditional probability distribution over the tokens in the vocabulary $\log P(\mathcal{V}|c)$ computed by the model are vertically stacked together resulting in a $\log P$ matrix of size $|\mathcal{C}| \times |\mathcal{V}|$. In PTB and WT2 datasets, there are 82,430 and 245,370 contexts in their respective test sets. In the case of the PTB dataset, we use all 82,430 contexts to construct the $\log P$ matrix. However for WT2, due to time and space complexities involved, we consider only the first 33,320 contexts. Following previous works (Yang et al. 2018; Kanai et al. 2018; Ganea et al. 2019), we perform SVD on the $\log P$ matrix to get the singular values and then use Press et al. (2007)'s approach to identify non-zero singular values for calculating the rank of the $\log P$ matrix.

Statistical significance tests Wherever there is a need to claim that one language model is statistically significantly better than the other, we train each of the model 10 times using 10 randomly sampled seeds for the random initialization of the parameters of the model, and then evaluate each instance of the model on the test set resulting in a sample of test perplexities with sample size being 10. Finally, the p-value between the samples of test perplexities from two different models is calculated using an unpaired t-test. Conventionally, a p-value less than 0.05 is required to say that a model A is statistically significantly better than a model B.

GPUs All model training and evaluation were conducted using NVIDIA's V100 GPUs with 32GB of memory. To train a single instance of a model, we use only one GPU and not multiple GPUs.

Is Rank Correlated with Performance?

Generalized SigSoftmax

Kanai et al. (2018) introduced the SigSoftmax (SS) function as an alternative to the softmax function, which can be defined as:

$$\mathrm{SS}(l_z) = \frac{\exp(l_z)\sigma(l_z)}{\sum_{i=1}^{N} \exp(l_i)\sigma(l_i)} \quad (1)$$

where l_z is the z-th component in the logits vector. $\mathrm{SS}(l_z)$ can also be written in terms of softmax as $\mathrm{softmax}(2l_z - \ln(1+\exp(l_z)))$. This dual form of $\mathrm{SS}(l_z)$ served as a starting point for our approach. First, we define a piecewise linear function with two pieces and two parameters c and k as:

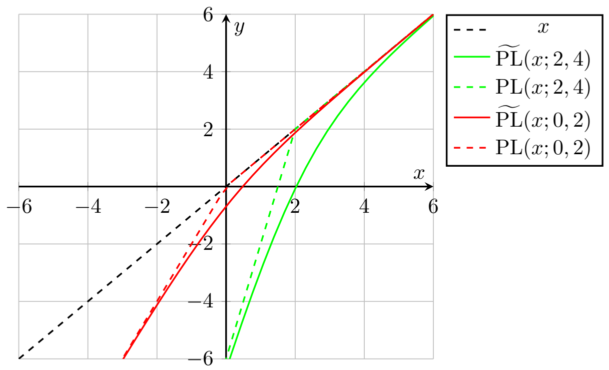

Figure 1: $\widetilde{\mathrm{PL}}(x;c,k)$ vs $\mathrm{PL}(x;c,k)$

$$\mathrm{PL}(x;c,k) = \begin{cases} x & x > c \\ k(x-c)+c & x \le c \end{cases} \quad (2)$$

Note that equation 2 is a generalization to the Parametric ReLU function (He et al. 2015). Now, we introduce a smooth approximation of the function in equation 2 as:

$$\widetilde{\mathrm{PL}}(x;c,k) = k(x-c) + c - (k-1)\,\mathrm{softplus}(x-c) \quad (3)$$

$\widetilde{\mathrm{PL}}$ is a smooth approximation of PL because,

$$\text{as } x \to \infty, \quad \widetilde{\mathrm{PL}}(x;c,k) \to x$$
$$\text{as } x \to -\infty, \quad \widetilde{\mathrm{PL}}(x;c,k) \to k(x-c)+c$$

This can also be seen in Figure 1. Now, we can represent $\mathrm{SS}(l_z)$ in terms of $\widetilde{\mathrm{PL}}$:

$$\mathrm{SS}(l_z) = \mathrm{softmax}(\widetilde{\mathrm{PL}}(l_z;0,2)) \quad (4)$$

Finally, we define Generalized Sigsoftmax (GSS) as follows:

$$\mathrm{GSS}(l_z;c,k) = \mathrm{softmax}(\widetilde{\mathrm{PL}}(l_z;c,k)) \quad (5)$$

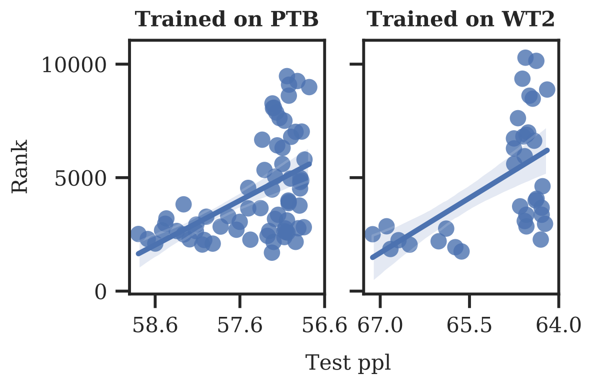

Figure 2: Correlation between the rank of the $\log P$ matrices and perplexity calculated on the test set for several models that use different values for the parameters c and k of GSS. On the PTB dataset, the Pearson correlation coefficient is 0.48. On WT2, it is 0.53.

Despite being members of the same family (GSS), softmax and SS had resulted in distinct ranks for their $\log P$ matrix (Table 1), which is our inspiration that other members of the family may allow us to produce diversely ranked $\log P$ matrices, and having such matrices can help us in better understanding the consequences of breaking the softmax bottleneck.

Correlation between the rank of the $\log P$ matrix and perplexity on the test set

To empirically show that the models using $\text{GSS}(l_z; c, k)$ can produce $\log P$ matrices with diverse ranks, we grid search over the cross product of $c = \{0.5\,x \mid x \in [-4, 4]\}$ and $k = \{0.25\,x \mid x \in [5, 12]\}$ on the PTB dataset, and over the cross product of $c = \{0.5\,x \mid x \in [-4, 4]\}$ and $k = \{0.5\,x \mid x \in [3, 6]\}$ on the WT2 dataset. Note that we fix the values for c and k before training. The ranks of the $\log P$ matrices produced by these models along with their test perplexities are shown in Figure 2, in which we can see that GSS is indeed able to produce $\log P$ matrices with diverse ranks. For all our further experiments concerning GSS, the parameters (c, k) are $(-1.5, 2.5)$ and $(-1.5, 3)$ for PTB and WT2 datasets respectively. We picked these values for c and k arbitrarily from the set of (c, k) pairs that result in a high-rank $\log P$ matrix as well as a lower test perplexity.

On both PTB and WT2 datasets, the Pearson correlation coefficients for the relationship between the two variables of interest (1. The rank of the $\log P$ matrix, 2. The test perplexity) is around 0.50. To the best of our knowledge, interpreting the correlation coefficient and classifying them into correlation classes such as weak, moderate, and strong is not standardized and is dependent on the context where it is used. If we exclude the outliers and consider only the test perplexity range of $[56.6, 57.6]$ for the PTB dataset and $[64.0, 65.0]$ for the WT2 dataset, the Pearson correlation coefficient drops to 0.24 and 0.21 on PTB and WT2 datasets respectively. Now, we can safely conclude that the correlation between the two variables of interest is fairly weak.

Quantitative comparison

We compare Softmax, SS, GSS, LMS-PLIF, and MoS models on both PTB and WT2 datasets. Each of these models were trained 11 times using 11 different seeds (10 randomly sampled seeds and the reported seed by respective authors) for the random initialization of model parameters. All models were trained till they converged. We report the mean and one standard deviation for performance metrics. The results are shown in Table 2. On the PTB dataset, we could see that SS and GSS models are not statistically significantly better than the Softmax model, as the p-values are greater than 0.05 despite their differences in the rank of the $\log P$ matrix. On the WT2 dataset, though SS and GSS models are statistically significantly better than the Softmax model, the relative performance improvements in terms of test perplexity are very minimal. On both datasets, LMS-PLIF and MoS models are statistically significantly better than the Softmax model. The relative performance improvements when compared with the Softmax model is small for LMS-PLIF and indeed noteworthy for MoS.

Qualitative analysis

Yang et al. (2018) conducted a qualitative analysis by comparing the quality of predictions made by MoS and MoC models. They cherry-picked six different contexts from the test set of PTB and compared the top-5 predictions made by MoS and MoC models. They showed that MoS was able to get the true next-token among its top-5 predictions for all the six contexts, but MoC did not get the true token among its top-5 predictions for any of the contexts. They claimed that MoS was able to make such accurate context-dependent predictions because of its ability to produce high-rank $\log P$ matrices. We redo the same analysis but we also include Softmax, SS, GSS, LMS-PLIF models in addition to MoS and MoC. Remember that all these models were trained by us using ET-ASGD. The test perplexity of MoC on PTB is 55.81. As they had used NT-ASGD, we also include MoS* and MoC* models that were trained by us using NT-ASGD whose test perplexities are 56.07 and 57.40 respectively, whereas their reported test perplexities are 55.97 and 57.55. In addition to the six contexts cherry-picked by them, we cherry-picked a different set of six contexts that we believe is challenging for the models to make the right context-dependent prediction. Also, we included another six randomly selected contexts. In Figure 3, note that for the contexts cherry-picked by them, there is always a low-rank model that is able to get the true token among its top-5 predictions, in addition to a high-rank model if any. We can observe the same for the contexts cherry-picked by us as well as the randomly chosen contexts. Hence, we believe that a model's qualitative performance has no strong correlation with its ability to produce high-rank $\log P$ matrices.

Analysis on word embeddings

Among the trainable parameters of all these models, word embeddings are the ones that can be evaluated using different well-known techniques. Hence, the learned embeddings

Model	#Param	Time (/epoch)	Train ppl	Validation ppl	Test ppl	**p-value** (compared with Softmax)	**Rank** (of $\log P$ matrix)
			Penn Treebank dataset				
Softmax	24.22M	**$\sim$55s**	33.91±0.25	59.55±0.12	57.08±0.09	N/A	402±0
SS	24.22M	$\sim$57s	32.87±0.19	59.69±0.15	57.05±0.15	3.92×10^{-1}	5,113±85
GSS	24.22M	$\sim$58s	33.68±0.32	59.51±0.13	57.00±0.14	2.62×10^{-1}	8,904±57
LMS-PLIF	24.32M	$\sim$59s	36.67±0.30	59.08±0.09	56.80±0.11	1.91×10^{-5}	496±83
MoS	21.50M	$\sim$100s	**31.84±0.23**	**57.14±0.26**	**54.91±0.26**	2.02×10^{-15}	**9,981±2**
			WikiText-2 dataset				
Softmax	33.55M	**$\sim$72s**	39.32±0.14	67.57±0.14	64.63±0.08	N/A	402±0
SS	33.55M	$\sim$83s	39.29±0.17	67.28±0.16	64.35±0.17	3.46×10^{-4}	6,634±46
GSS	33.55M	$\sim$87s	39.42±0.41	67.35±0.22	64.50±0.13	2.74×10^{-2}	10,122±72
LMS-PLIF	33.65M	$\sim$93s	41.13±0.15	67.15±0.21	64.16±0.17	2.98×10^{-7}	524±74
MoS	34.90M	$\sim$600s	**36.17±0.30**	**64.54±0.36**	**61.96±0.40**	2.04×10^{-13}	**15,734±188**

Table 2: Performance comparison between different models after each model was trained 11 times using 11 different seeds for random initialization of the parameters in the model. Values mentioned as $x \pm y$ denote the mean $\pm$ one standard deviation. For unpaired t-tests, the sample size of test perplexities is 10 and not 11, as we consider only the model instances whose parameters were randomly initialized using the 10 randomly sampled seeds. More details in SM Table 1.

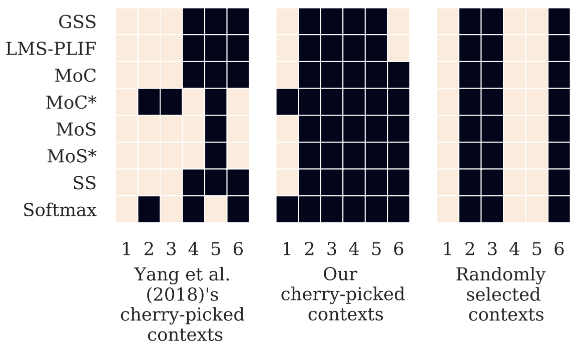

Figure 3: *Non-black* colored cells $c_{i,j}$ denote that the model i got the true next-token among its top-5 predictions for the given context j. Otherwise, *black* colored. MoS* and MoC* were trained using NT-ASGD. Rest of the models use ET-ASGD. More details in SM Table 2.

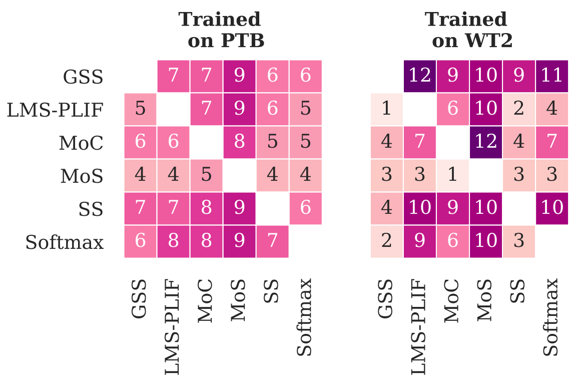

Figure 4: Each cell c_{ij} denotes the number of benchmarks on which the embeddings from the model i has better Spearman's correlation coefficient than those from the model j. On PTB, MoS and MoC use $d = 280$, rest use $d = 400$. On WT2, MoS and MoC use $d = 300$, rest use $d = 400$. More details in SM Table 3.

from Softmax, SS, GSS, LMS-PLIF, MoS, and MoC models are evaluated on 13 word similarity benchmark datasets, to identify if there is any additional benefit in having a high-rank language model, and the results are pairwise compared with each other. The results are shown in Figure 4. One of the key observation is that MoC's embeddings has a better Spearman's correlation coefficient than that of MoS on 8 word similarity benchmarks when the models were trained on PTB. Similarly, on WT2, MoC's embeddings performs better than that of MoS on 12 benchmarks. Remember that MoC does not break the softmax bottleneck. Hence, we state that a model which can produce a high-rank $\log P$ matrix does not necessarily learn better embeddings, at least on these word similarity benchmarks.

Is the Better Performance of LMS-PLIF Due to High Rank?

As seen in Table 2, we know that the rank of the $\log P$ matrix produced by LMS-PLIF on both PTB and WT2 datasets is fairly low, but the model has a better test performance than the Softmax and SS models. Hence, we set out to find the reason for its better test performance. In LMS-PLIF model, the PLIF layer is responsible to learn a piecewise linear increasing function. Essentially, the parameters in the PLIF layer that gets learned are the slopes of several lines (pieces). As LMS-PLIF introduces these additional trainable parameters unlike SS and GSS models, we attempt to find the impact of PLIF layer on the performance of LMS-PLIF model. We freeze the PLIF layer so that the randomly initialized slope values remain constant and do not get changed dur-

Dataset	Train ppl	Validation ppl	Test ppl
PTB	37.16±0.24	59.23±0.11	56.83±0.10
WT2	41.67±0.15	67.22±0.13	64.28±0.17

Table 3: Performance of LMS-PLIF† in which the PLIF layer is frozen.

Dataset	Test ppl		p-value
	LMS-PLIF	LMS-PLIF†	
PTB	56.81 ± 0.11	56.82 ± 0.09	8.26 $\times 10^{-1}$
WT2	64.15 ± 0.17	64.30 ± 0.17	6.41 $\times 10^{-2}$

Table 4: p-value from an unpaired t-test between samples of test perplexities of LMS-PLIF† and that of LMS-PLIF. The sample size is 10.

ing training. As shown in Table 4, if we compare the LMS-PLIF model against LMS-PLIF† in which the PLIF layer is frozen, we can see that LMS-PLIF is not statistically significantly better than LMS-PLIF† on both the datasets in terms of test perplexity. What we could potentially infer from this observation is that either the learned slopes are not better than the randomly initialized slopes or the slopes are not getting learned well enough. Note that LMS-PLIF† is exactly the same as the Softmax model except for the constant changes made to the logits before passing them to the softmax function. We get another interesting insight when the LMS-PLIF† model (Table 3) is compared against the Softmax model (Table 2). If we look at the perplexities of both these models, we can see that LMS-PLIF† has traded its performance on the training set for a small performance gain on its validation and test sets, which is essentially a form of regularization. Hence, we suggest that the better performance of LMS-PLIF is more likely due to this implicit regularization because of the PLIF layer.

Is the Better Performance of MoS Due to High Rank?

To verify the role of rank in the better performance of MoS, Yang et al. (2018), on the PTB dataset, showed that increasing the number of mixtures K lead to an increase in the rank of the $\log P$ matrix, and observed a positive correlation between the rank of the $\log P$ matrix and test performance. We repeated this experiment on both PTB and WT2 datasets and observed the same (SM Tables 4 and 5). We also conducted an experiment on the WT2 dataset to show that it is possible to increase the rank of the $\log P$ matrix without increasing K. As shown in Figure 5, we fix $K = 15$ and decrease the dropout rates applied to the MoS layer, and observe a negative correlation between the rank of the $\log P$ matrix and test performance. Hence, we think that the rank of the $\log P$ matrix being high is just a by-product and doubt the claim that it plays a major role for the better test performance of MoS.

Then, to understand why MoS is able to have a better test performance when K is increased, we look at the relationship between K and the capacity (in terms of parameters) for MoS models on both PTB and WT2 datasets. As shown in Figure 6, increasing K actually increases the capacity of the model which might have been the reason for K's positive correlation with better test performance. Yang et al. (2018), on the PTB dataset, had showed that using $K = 15$ result in an almost full-rank $\log P$ matrix, and stated that further increasing the value of K leads to overfitting as there is no more room for improvement in the rank of the $\log P$ matrix. We speculate that the overfitting is actually because of further increasing the capacity, which results in a wider generalization gap (Figure 6).

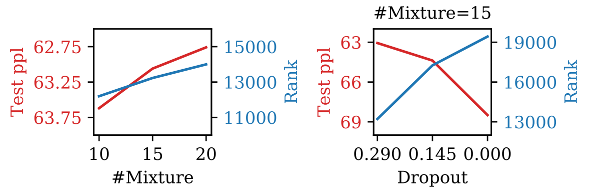

Figure 5: Positive (*left*) and negative (*right*) correlation between the rank of the $\log P$ matrix and test performance (in terms of perplexity) for MoS models trained on WT2.

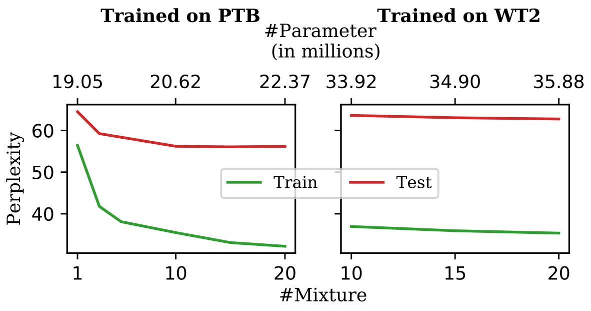

Figure 6: Relationship between the number of mixtures, capacity (in terms of model parameters), and the performance (in terms of perplexity) for trained MoS models.

Yang et al. (2018) used a different set of hyperparameters for MoS on both PTB and WT2 datasets when compared to the baseline Softmax model, which we think as unfair. The differences in hyperparameters were few in the case of PTB when compared to WT2. Also for MoS, it is less time consuming to conduct extensive experiments on PTB over WT2. Hence, we set out to do a fair comparison between MoS and Softmax models on the PTB dataset. To begin with, we want to highlight three hyperparameters that are different in MoS namely the training batch size (bsz), the context vector dimension (d'), and the embedding vector dimension (d). Note that the context vector in AWD-LSTM based models is the output from the top-most LSTM layer. The MoS model uses bsz $= 12$, $d' = 620$, and $d = 280$ whereas the Softmax model uses bsz $= 20$, $d' = 400$, and $d = 400$. Having such a high context vector dimension can be advantageous for the MoS model as it could encode much more context information. Hence, for a fair

Model	**Train ppl**	**Validation ppl**	**Test ppl**
MoS†	42.45±1.03	58.53±0.17	56.39±0.19
Softmax‡	40.87±0.18	58.61±0.10	56.45±0.10

Table 5: Fair comparison between MoS† and Softmax† models on the PTB dataset. MoS† and Softmax† have 17.53M and 17.56M parameters respectively.

comparison, we make bsz $=$ 12 and $d' = d = 280$ for both MoS and Softmax models which we call as MoS† and Softmax† respectively. The MoS† model would then have a total of 17.53M parameters whereas Softmax† would have only 16.35M parameters. This difference is due to the additional trainable parameters introduced by the MoS layer. For rest of the regularization hyperparameters like dropouts and weight decay, we perform a small-scale hyperparameter finetuning for the Softmax† model to make it perform as good as the MoS† model. However, we were not successful. We assume that it could be because of the less capacity of the Softmax† model. Hence, we made the models under comparison to have a comparable capacity in terms of the number of parameters. To do so, we make $d' = d = 340$ for the Softmax model, which we call Softmax‡ that has a total of 17.56M parameters which is comparable to that of MoS†. We once again perform a hyperparameter finetuning for Softmax‡ to make its performance as good as the MoS†. We were also fair to the MoS† model by finetuning its regularization hyperparameters for an even better performance. The details about hyperparameter finetuning are in SM Section 2. Finally, when MoS† is compared against Softmax‡, we observe that MoS† is not statistically significantly better than Softmax‡, as the p-value is 0.62. We are aware that there could be another set of hyperparameters with which the MoS model could perform significantly better than Softmax model and vice-versa. But we speculate that either the better performance of MoS model in general could be due to its inductive bias or the Softmax model can be made to perform as good as the MoS model when it is made to encode a better inductive bias through techniques like regularization and if the comparison is fair.

Conclusions and Further Discussion

We showed that the high rank of the $\log P$ matrix is neither necessary nor sufficient for the better performance of a recurrent language model. We suggested that an implicit regularization due to the PLIF layer is more likely the reason for LMS-PLIF's better performance. We also suggested that the high-rank $\log P$ matrix produced by MoS could just be a by-product and speculate that the better performance of MoS in general could primarily be due to various other factors that deal with inductive bias and capacity of the model.

Fast singular value decay We also inspected the singular value distributions resulting from the SVD of the $\log P$ matrices. Though there were differences on a log scale (SM Figures 2 and 4), the distributions look more or less the same when the singular values were normalized to [0,1] as seen in Figure 7. Also, the decay rate of all these distributions seems to be very high. Hence, we suspect that the rank values calculated using Press et al. (2007)'s approach could be an overestimate of the effective rank. To support our suspicion, we introduce an alternative metric for calculating the rank which we call ϵ-effective rank. Let $\sigma_1, \sigma_2, .., \sigma_n$ be the singular values resulting from the SVD of a $\log P$ matrix.

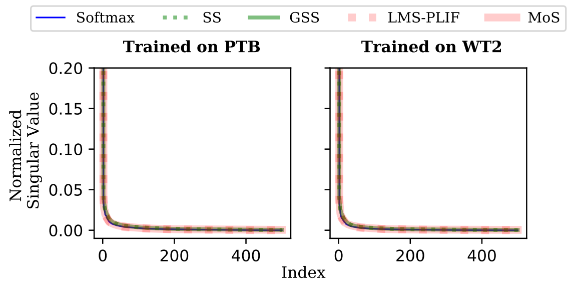

Figure 7: Normalized singular values obtained from the SVD of $\log P$ matrices. For better visibility, the range of y-axis is limited to $[0.0, 0.2]$ and only the first 500 indices are shown in the x-axis.

Model	**Press' rank**	**Effective rank** for various ϵ		
		10^{-3}	10^{-4}	10^{-5}
Penn Treebank dataset				
Softmax	402	52	201	306
SS	4,979	100	297	1,038
GSS	8,989	100	559	3,456
LMS-PLIF	580	50	198	335
MoS	9,983	81	1,521	6,428
WikiText-2 dataset				
Softmax	402	27	141	274
SS	6,590	54	249	1,201
GSS	10,145	60	391	2,988
LMS-PLIF	513	29	150	287
MoS	15,738	49	773	5,982

Table 6: Comparison between Press' rank and ϵ-effective rank

$$\sum_{i=1}^{k} \sigma_i^2 \geq (1-\epsilon) \sum_{i=1}^{n} \sigma_i^2 \quad \forall \epsilon \in [0,1] \qquad (6)$$

The smallest value for k, to which the inequality mentioned in equation 6 holds true, is defined as the ϵ-effective rank. In words, "the smallest number of singular values whose squares sum to equal or more than $1-\epsilon$ fraction of the total sum of squares of singular values" is called the ϵ-effective rank. We compare Press et al. (2007)'s rank and ϵ-effective rank for the $\log P$ matrices produced by different models. As shown in Table 6, it seems that Press et al. (2007)'s rank is most likely an overestimate. We discuss more about Press' rank vs ϵ-effective rank on our SM's Section 5.

Acknowledgements

This research was funded by the Natural Sciences and Engineering Research Council of Canada (NSERC). The experiments were enabled by the support provided by Calcul Quebec (www.calculquebec.ca), Compute Ontario (www.computeontario.ca), West Grid (www.westgrid.ca), and Compute Canada (www.computecanada.ca). We also thank Octavian Ganea and Lingxiao Wang for sharing their code with us.

References

Agirre, E.; Alfonseca, E.; Hall, K.; Kravalova, J.; Pasca, M.; and Soroa, A. 2009. A Study on Similarity and Relatedness Using Distributional and WordNet-based Approaches. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, 19–27. Boulder, Colorado: Association for Computational Linguistics. URL https://www.aclweb.org/anthology/N09-1003.

Baker, S.; Reichart, R.; and Korhonen, A. 2014. An Unsupervised Model for Instance Level Subcategorization Acquisition. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 278–289. Doha, Qatar: Association for Computational Linguistics. doi:10.3115/v1/D14-1034. URL https://www.aclweb.org/anthology/D14-1034.

Bruni, E.; Boleda, G.; Baroni, M.; and Tran, N.-K. 2012. Distributional semantics in technicolor. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 136–145.

Faruqui, M.; and Dyer, C. 2014. Community Evaluation and Exchange of Word Vectors at wordvectors.org. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, 19–24. Baltimore, Maryland: Association for Computational Linguistics. doi:10.3115/v1/P14-5004. URL https://www.aclweb.org/anthology/P14-5004.

Finkelstein, L.; Gabrilovich, E.; Matias, Y.; Rivlin, E.; Solan, Z.; Wolfman, G.; and Ruppin, E. 2001. Placing search in context: The concept revisited. In *Proceedings of the 10th international conference on World Wide Web*, 406–414.

Ganea, O.; Gelly, S.; Bécigneul, G.; and Severyn, A. 2019. Breaking the Softmax Bottleneck via Learnable Monotonic Pointwise Non-linearities. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97, 2073–2082. URL http://proceedings.mlr.press/v97/ganea19a.html.

Gerz, D.; Vulic, I.; Hill, F.; Reichart, R.; and Korhonen, A. 2016. SimVerb-3500: A Large-Scale Evaluation Set of Verb Similarity. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2173–2182. Austin, Texas: Association for Computational Linguistics. doi:10.18653/v1/D16-1235. URL https://www.aclweb.org/anthology/D16-1235.

Halawi, G.; Dror, G.; Gabrilovich, E.; and Koren, Y. 2012. Large-scale learning of word relatedness with constraints. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, 1406–1414.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, 1026–1034.

Hill, F.; Reichart, R.; and Korhonen, A. 2015. SimLex-999: Evaluating Semantic Models With (Genuine) Similarity Estimation. *Computational Linguistics* 41(4): 665–695. doi:10.1162/COLI_a_00237. URL https://www.aclweb.org/anthology/J15-4004.

Kanai, S.; Fujiwara, Y.; Yamanaka, Y.; and Adachi, S. 2018. Sigsoftmax: Reanalysis of the Softmax Bottleneck.

Luong, M.-T.; Socher, R.; and Manning, C. D. 2013. Better Word Representations with Recursive Neural Networks for Morphology. In *CoNLL*. Sofia, Bulgaria.

Marcus, M. P.; Santorini, B.; and Marcinkiewicz, M. A. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics* 19(2): 313–330. URL https://www.aclweb.org/anthology/J93-2004.

Merity, S.; Keskar, N. S.; and Socher, R. 2018. Regularizing and Optimizing LSTM Language Models. *ArXiv* .

Merity, S.; Xiong, C.; Bradbury, J.; and Socher, R. 2017. Pointer Sentinel Mixture Models. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net. URL https://openreview.net/forum?id=Byj72udxe.

Mikolov, T.; and Zweig, G. 2012. Context dependent recurrent neural network language model. In *2012 IEEE Spoken Language Technology Workshop (SLT)*, 234–239.

Miller, G. A.; and Charles, W. G. 1991. Contextual correlates of semantic similarity. *Language and cognitive processes* 6(1): 1–28.

Press, W. H.; Teukolsky, S. A.; Vetterling, W. T.; and Flannery, B. P. 2007. *Numerical recipes 3rd edition: The art of scientific computing*. Cambridge university press.

Radinsky, K.; Agichtein, E.; Gabrilovich, E.; and Markovitch, S. 2011. A word at a time: computing word relatedness using temporal semantic analysis. In *Proceedings of the 20th international conference on World wide web*, 337–346.

Rubenstein, H.; and Goodenough, J. 1965. Contextual correlates of synonymy. *Commun. ACM* 8: 627–633. doi:10.1145/365628.365657.

Takase, S.; Suzuki, J.; and Nagata, M. 2018. Direct Output Connection for a High-Rank Language Model. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 4599–4609. Brussels, Belgium: Association for Computational Linguistics. doi:10.18653/v1/D18-1489. URL https://www.aclweb.org/anthology/D18-1489.

Wang, D.; Gong, C.; and Liu, Q. 2019. Improving Neural Language Modeling via Adversarial Training. In Chaudhuri, K.; and Salakhutdinov, R., eds., *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, 6555–6565. Long Beach, California, USA: PMLR. URL http://proceedings.mlr.press/v97/wang19f.html.

Wang, L.; Huang, J.; Huang, K.; Hu, Z.; Wang, G.; and Gu, Q. 2020. Improving Neural Language Generation with Spectrum Control. In *International Conference on Learning Representations*. URL https://openreview.net/forum?id=ByxY8CNtvr.

Yang, D.; and Powers, D. M. W. 2006. Verb similarity on the taxonomy of WordNet. In *GWC 2006: Third International WordNet Conference*.

Yang, Z.; Dai, Z.; Salakhutdinov, R.; and Cohen, W. W. 2018. Breaking the Softmax Bottleneck: A High-Rank RNN Language Model. In *International Conference on Learning Representations*. URL https://openreview.net/forum?id=HkwZSG-CZ.

Yang, Z.; Luong, T.; Salakhutdinov, R. R.; and Le, Q. V. 2019. Mixtape: Breaking the Softmax Bottleneck Efficiently. In *Advances in Neural Information Processing Systems 32*, 5775–5783. URL http://papers.nips.cc/paper/9723-mixtape-breaking-the-softmax-bottleneck-efficiently.pdf.