MAJOR-MINOR LSTMS FOR WORD-LEVEL LANGUAGE MODEL

Anonymous authors
Paper under double-blind review

ABSTRACT

As a widely-accepted evaluation criterion, complexity has attracted more and more attention in the design of language models. The parameter count is a proxy for complexity, which is often reported and compared in research papers. In general, more parameters means better model performance, but higher complexity. Therefore, reconciling the contradiction between the complexity and the model performance is necessary. In this paper, we propose a simple method to make use of model parameters more effectively, so that the LSTM-based language models can reach better results without the cost of increasing parameters. The method constructs another small-scale LSTM with a part of parameters originally belonging to the vanilla LSTM in each layer, whose output can assist the next layer in processing the output of the vanilla LSTM. We name these two LSTMs Major-Minor LSTMs. In experiments, we demonstrate the language model with Major-Minor LSTMs surpasses the existing state-of-the-art model on Penn Treebank and WikiText-2 with fewer parameters.

1 INTRODUCTION

Language model (LM) is a foundational component of natural language processing (NLP) tasks, which estimates the probability distribution of a sequence of words $(w_0, ..., w_n)$ by modeling the probability of the next word (w_i) given preceding words $(w_0, ..., w_{i-1})$, i.e.

$$P(w_0, ..., w_n) = P(w_0) \prod_{i=1}^{N} P(w_i | w_0, ..., w_{i-1})$$

Language model plays an important role of systems for machine translation Koehn (2009), speech recognition (Yu & Deng, 2014), learning token embeddings (Botha & Blunsom, 2014; Press & Wolf, 2016), natural language generation (Radford et al., 2017; Merity et al., 2017) and text classification (Howard & Ruder, 2018).

For word-level language model, the vanilla multi-layer Long Short Term Memory (LSTM) networks have been demonstrated to achieve state-of-the-art performance (Yang et al., 2017; Merity et al., 2017). As one of the successful variants of Recurrent Neural Network (RNN), LSTM (Hochreiter & Schmidhuber, 1997) can not only process word sequences of any length, but also alleviate the issue of gradient vanishing effectively.

Since the probability distribution to be predicted is complex, current LSTM-based language models often utilize large-scale LSTMs with multiple layers to enhance generalization capability of the language model, which lead to a huge amount of parameters. In fact, we have found through a lot of experiments that when the scale of a certain LSTM layer is large enough, the improvement of overall model performance will reduce as the LSTM output dimension increases, even the change within a certain range will not cause obvious difference of the model result. But for these large-scale LSTM layers, a small increase in their output size requires a large number of parameters to generate. This means that in order to achieve model effect as good as possible, there will inevitably be a part of parameters in the LSTMs with little contribution. Therefore, how to effectively improve the utilization of model parameters is a challenge for language model.

For the sake of clarity, we call the LSTM in each layer of the vanilla LSTM-based language model Major LSTM. In this paper, we propose to reduce the dimension of Major LSTM appropriately, and

set up a Minor LSTM with the saved parameters, under the premise of not impairing the performance of the Major LSTM significantly. The Minor LSTM generates a set of auxiliary features, which has capability to assist the next layer in processing the output of the current Major LSTM. Since the scale of the Minor LSTM is much smaller than the Major LSTM, the parameters building the Minor LSTM do not exceed the saved parameters. The architecture of the layer we proposed is called Major-Minor LSTMs (MMLSTMs), and for convenience the language model consisting of MMLSTMs is called Major-Minor Language Model (MMLM). We will detailed introduce the architecture of MMLSTMs and explain the rationality of the Minor LSTM later.

We integrate some existing methods (e.g. regularization, optimization, Mixture of Softmax) into our MMLM, and evaluate it on standard language modeling benchmarks. The experimental results show that the MMLM surpasses existing state-of-the-art language model on both Penn Treebank and WikiText-2 datasets with fewer parameters.

2 Related Work

Language modeling has gone through significant development from conventional N-gram language models (Kneser & Ney, 1995; Chen & Goodman, 1999) to neural language models (Bengio et al., 2003; Mikolov et al., 2010; Jozefowicz et al., 2016) in these decades. These classical N-gram models suffer from data sparsity, which makes it difficult to represent large contexts and thus, long-range dependencies. The LSTM-based language models effectively alleviate the problem of long-range dependency, so their performance greatly exceed other neural network language models. In addition, these LSTM-based language models also use a large amount of parameters for better generalization.

Zaremba et al. (2014) applied dropout to the non-recurrent connections in LSTM language models with medium and large sizes, and achieved the best results with the large one at that time. Although, the large model was only slightly better than the medium model in the final performance, its parameter count is three times that of the latter. Gal & Ghahramani (2016) used the large LSTM language model with a dropout variation to surpass the result of Zaremba et al. (2014). But there was still no effective way to reduce the model complexity. Inan et al. (2016) introduced a novel theoretical framework leadingto tying together the input embedding and the output projection matrices, greatly reducing the number of trainable parameters. Merity et al. (2017) proposed a weight-dropped LSTM, which used DropConnect on hidden-to-hidden weights as a form of recurrent regularization. Further, they introduced a variant of averaged stochastic gradient method named NT-AvSGD, which was a successful improvement in the language model optimization method. Zolna et al. (2017) proposed to train two identical copies of an RNN (that share parameters) with different dropout masks while minimizing the difference between their predictions. Yang et al. (2017) identified the Softmax bottleneck by formulating language modeling as a matrix factorization problem, and proposed a method called Mixture of Softmax to address it.

However, for these language models using multi-layer LSTMs, in order to improve the model performance, they mainly leveraged the regularization method to reduce the over-fitting of the models under the same parameter counts. In contrast, our method is more straightforward. We assist the Major LSTM directly y constructing the Minor LSTM without increasing the cost of parameters. In addition, our model can be combined with existing regularization methods to produce better results.

3 Major-Minor LSTMs (MMLSTMs) for Language Model

3.1 The Vanilla LSTM-Based Language Model

A vanilla LSTM-based language model can be simply divided into three components: the input of word embeddings, the model of LSTMs, and the Softmax layer.

The mathematical formulation of the LSTM is as follow:

$$i_j = \sigma\left(W_i\left[h_{j-1}; v_j\right] + b_j\right) \tag{1}$$
$$f_j = \sigma\left(W_f\left[h_{j-1}; v_j\right] + b_f\right) \tag{2}$$
$$o_j = \sigma\left(W_o\left[h_{j-1}; v_j\right] + b_o\right) \tag{3}$$
$$g_j = \tanh\left(W_r\left[h_{j-1}; v_j\right] + b_r\right) \tag{4}$$
$$c_j = i_j \odot g_j + f_j \odot c_{j-1} \tag{5}$$
$$h_j = o_j \odot \tanh\left(c_j\right) \tag{6}$$

Where $[W_i, W_f, W_o, W_r], [b_i, b_f, b_o, b_r]$ are weight matrices and bias vectors. v_j is the j_{th} input vector, h_j is the current exposed hidden state, c_j is the memory cell state, and $\odot$ is element-wise multiplication. The embedding of word sequence is fed into LSTM layers, and the output of the last LSTM layer is used in predicting the probabilities distribution.

The Softmax layer calculates the probabilities of each word in the word sequence:

$$y_j = \frac{e^{W_s \cdot x_j}}{1^T \cdot e^{W_s \cdot x_j}} \tag{7}$$

Where $[W_s]$ is weight matrix, x_j is the output of the last LSTM layer corresponding to j_{th}word, $1^T = (1, ..., 1)$.

3.2 MAJOR-MINOR LSTMS FOR LANGUAGE MODEL

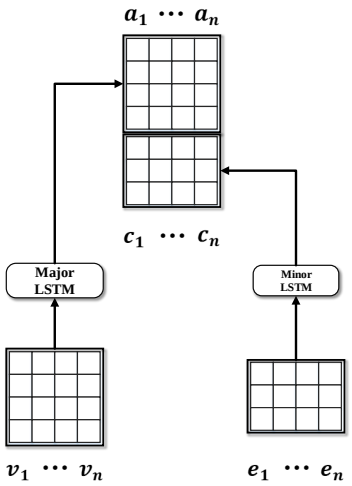


Figure 1: The architecture of MMLSTMs layer. $(v_1, ..., v_n)$ and $(e_1, ..., e_n)$ denote the output of the previous layer and the word embedding of the original word sequence respectively. $(a_1, ..., a_n)$ and $(c_1, ..., c_n)$ represent the output of the Major LSTM and the Minor LSTM.

As we discussed before, the size of LSTMs should be large enough so that the language model has sufficient learning capability. However, the promotion of the model performance will become unnoticeable when the parameter count reachs a certain level. For this reason, instead of keeping increasing the parameters of each layer of Major LSTM for a slight performance improvement, it is better to build a Minor LSTM with a low cost to generate an auxiliary feature vector, which assists the next layer in handling the output of the Major LSTM.

The architecture of a Major-Minor LSTMs layer is shown as Figure 1. The inputs of the Major and Minor LSTMs are the output of the previous layer $(v_1, ..., v_n)$ and the original embedding of word sequence $(e_1, ..., e_n)$, respectively. And the size of the Major LSTM output vector $(a_1, ..., a_n)$ is much larger than Minor LSTM $(c_1, ..., c_n)$. For facilitating adjustment, we only set the size of entire Major-Minor LSTMs layer, and adjust the proportions of the Major and Minor LSTMs.

The reason why we do not use the same input of the Major LSTM as the input of the Minor LSTM is three-fold. First, according to our hypothesis, the Major LSTM has enough capability to process the output of the previous layer, so that there is no need to process it with the Minor LSTM. Second, if we consider the layers before the current layer as a whole, the output of the current layer can be written as the function of the original embedding of the word sequence. So the output of the previous layer is only an intermediate result for generating the current layer output. The original embeddings, by contrast, can generate the auxiliary features needed by the current layer more directly. Third, as in Merity et al. (2017), the size of embedding is smaller than that size of hidden state, so that ease the issue of overfitting of the word embeddings. Therefore, compared with the output of the previous LSTM layer, using the original embedding as the input of the Minor LSTM can save more parameters.

3.3 THE RATIONALITY OF THE MINOR LSTM

In MMLM, the output of Major-Minor LSTMs layers will flow into the next MMLSTM layer or the last Softmax layer. But from the Formula (1-7) we can infer that whether the next layer is LSTM or Softmax layer, it can be simplified to the combination of different components in a form of nonlinear affine transformation:

$$y_t = f\left(W_1 \cdot x_t^{Major} + U_1 \cdot s_{t-1} + b_1 + W_2 \cdot x_t^{Minor}\right) \tag{8}$$

where x_t^{Major} $\left(x_t^{Minor}\right)$ denotes the t_{th} output of the Major (Minor) LSTM in the previous layer, s_{t-1} denotes the $(t-1)_{th}$ hidden state in LSTM or 0 in the Softmax layer, $[W_1, W_2, U_1, b_1]$ are parameters of the component. f denotes the nonlinear function of the component, which is sigmoid or tanh function in LSTM and softmax function in Softmax layer. The Formula (8) indicates that we can understand $W_2 \cdot x_t^{Minor}$ as an adaptive bias vector, which adjusts the values of the component y_t for different input word sequence. As a result, the auxiliary feature vector $\left(x_t^{Minor}\right)$ has the capability to assist the next layer in processing the output of the current Major LSTM.

But for general LSTM-based language models (only have Major LSTMs), we can expand the vector x_t^{Major} to $\left(x_t^{Major}, x_t^{Major'}\right)$ by increasing the dimension of the Major LSTM. In this way, the $W_2 \cdot x_t^{Minor}$ in Formula (8) is replaced by $W_2 \cdot x_t^{Major'}$, and the form of the Formula (8) does not have obvious changes. But since the x_t^{Major} and the $x_t^{Major'}$ are generated by the same LSTM, $x_t^{Major'}$ cannot be freely and independently transformed into an auxiliary feature vector for x_t^{Major} like x_t^{Minor} generated by another LSTM. Concretely, if we simplify the Major LSTM to a vanilla RNN, the x_t^{Major} and the $x_t^{Major'}$ can be calculated as:

$$x_t^{Major} = f\left(W \cdot z_t + U \cdot \left(x_t^{Major}, x_t^{Major'}\right) + b\right) \tag{9}$$

$$x_t^{Major'} = f\left(W' \cdot z_t + U' \cdot \left(x_t^{Major}, x_t^{Major'}\right) + b'\right) \tag{10}$$

Where f is the nonlinear function, z_t is the input corresponding to the t_{th} word. $[W, b]$ and $[W', b']$ are parameters corresponding to x_t^{Major} and $x_t^{Major'}$ respectively, and they do not interfere with each other. But because of the appearance of $\left(x_t^{Major}, x_t^{Major'}\right)$, the generation of x_t^{Major} $\left(x_t^{Major'}\right)$ becomes a recursive procedure, which is controlled by both $[W, U, b]$ and $[W', U', b']$. Any change in each parameter will affect every output dimension of the LSTM, so it is difficult to split an independent portion of its output which can be freely transformed as the auxiliary feature vector of the other part.

For MMLSTMs, the auxiliary feature vector from the Minor LSTM can generate an adaptive bias vector to adjust the calculation of current Major LSTM output. By contrast, splitting a part of output in a single LSTM as the auxiliary feature vector will bring a heavier parameter burden to the LSTM.

Hyper-parameter	PTB	WT2
Learning rate	20	20
MMLSTMs layer size	[1080, 1080, 620]	[1200, 1200, 650]
Ratio of Major LSTM	[90%, 90%, 60%]	[90%, 90%, 60%]
Embedding V-dropout	0.5	0.55
Hidden state V-dropout	0.25	0.2
Non-monotone interval	10	10

Table 1: Hyper-parameters used for AWD-MMLSTMs-MoS. V-dropout abbreviates variational dropout (Gal & Ghahramani, 2016). See Merity et al. (2017)for more detailed descriptions.

Hyper-parameter	PTB	WT2
Batch size	150	130
Learning rate(η)	0.0024	0.00198
ε	0.0025	0.0023
λ	0.07	0.0245
bptt	7	7

Table 2: Hyper-parameters used for dynamic evaluation of AWD-MMLSTMs-MoS. See Krause et al. (2017) for more detailed descriptions.

4 EXPERIMENT

4.1 EXPERIMENT DETAILS

We evaluate our MMLM on a preprocessed version of the Penn Treebank (PTB) (Marcus et al., 1993) and the WikiText-2 (WT2) dataset (Merity et al., 2016), and use perplexity as the primary metric. In order to directly verify that our method can further heighten the effect of LSTM-based language model on the basis of some existing techniques, we apply MMLSTMs in AWD-LSTM–MoS architecture proposed by Yang et al. (2017), named AWD-MMLSTMs-MoS. We finetune the hyper-parameters in Yang et al. (2017) based on the validation performance while controlling the model size, and list the hyper-parameters we modified in Table 1. For facilitating the adjustment of the regularization methods on Major and Minor LSTMs, we tie up their embedding and hidden state Variational dropout values. Regarding the dimensional proportion of the Major and Minor LSTMs, we divide each layer into 10 parts, and search various combinations heuristically and manually. It was found that the dimensions of the Major LSTMs on three layers accounted for [90%, 90%, 60%] respectively, and this model can achieve the best results on both datasets. We also apply dynamic evaluation Krause et al. (2017) to further improve our model performance, whose hyper-parameters are listed in Table 2.

4.2 MAIN RESULTS

We present the perplexity results from both our models (AWD-MMLSTMs-MoS) and other competitive models in Table 3 and 4 for PTB and WT2 respectively. We improve the state-of-the-art results on both datasets, 46.81 on PTB and 40.15 on WT2. In conclusion, obtaining our model requires three stages: training, fine-tuning, and dynamic estimation. Our model surpasses AWD-LSTM-MoS in all stages, which fully proves the effectiveness of our method.

4.3 STUDY OF DIMENSIONAL CHANGES ON DIFFERENT LSTM LAYERS

The idea of Major-Minor LSTMs layer is based on our observation that reducing the dimensions of Major LSTMs in each layer properly does not significantly affect the performance of the Major LSTMs. So we build a small-scale Minor LSTM to extract a set of auxiliary features to assist the next layer in processing the output of the current Major LSTM. In this subsection, we study the influence of dimensional changes in different LSTM layers on the vanilla LSTM-based language

Model	#Param	Validation	Test
Mikolov & Zweig (2012) – RNN-LDA + KN-5 + cache	9M	-	92.0
Zaremba et al. (2014) - LSTM (large)	66M	82.2	78.4
Gal & Ghahramani (2016) – Variational LSTM (MC)	66M	-	73.4
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	51M	71.1	68.5
Zilly et al. (2016)– Variational RHN	23M	67.9	65.4
Zoph & Le (2016) – NAS Cell	54M	-	62.4
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
Yang et al. (2017) – AWD-LSTM-MoS w/o finetune	21.5M	58.08	55.97
Yang et al. (2017) – AWD-LSTM-MoS	21.5M	56.54	54.44
Ours-AWD-MMLSTMs-MoS w/o finetune	21.3M	56.52	54.51
Ours-AWD-MMLSTMs-MoS	**21.3M**	**55.42**	**53.46**
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
Yang et al. (2017) – AWD-LSTM-MoS + dynamic evaluation*	21.5M	48.33	47.69
Ours-AWD-MMLSTMs-MoS + dynamic evaluation*	**21.3M**	**47.31**	**46.81**

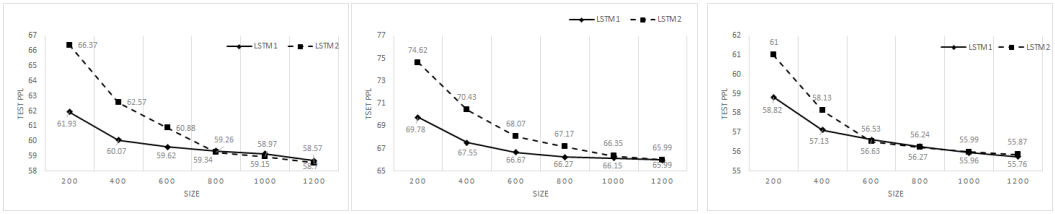
Table 3: Single model perplexity on validation and test sets on Penn Treebank. Baseline results are obtained from Merity et al. (2017) and Yang et al. (2017). * indicates using dynamic evaluation.

Model	#Param	Validation	Test
Inan et al. (2016) - Variational LSTM (tied) + 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 (tied)	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
Yang et al. (2017) – AWD-LSTM-MoS w/o finetune	35M	66.01	63.33
Yang et al. (2017) – AWD-LSTM-MoS	35M	63.88	61.45
Ours-AWD-MMLSTM-MoS w/o finetune	32.3M	64.3	61.77
Ours-AWD-MMLSTM-MoS	**32.3M**	**63.11**	**60.51**
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
Yang et al. (2017) – AWD-LSTM-MoS + dynamical evaluation*	35M	42.41	40.68
Ours-AWD-MMLSTMs-MoS + dynamical evaluation*	**32.3M**	**41.91**	**40.15**

Table 4: Single model perplexity over WikiText-2. Baseline results are obtained from Merity et al. (2017) and Yang et al. (2017). * indicates using dynamic evaluation.

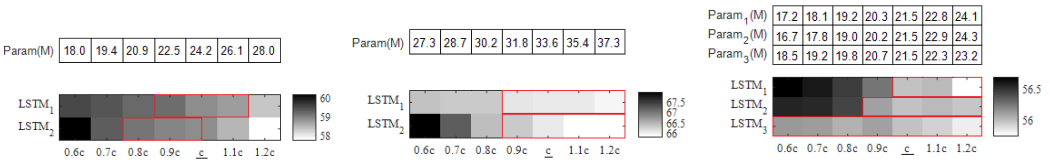
model orderly. We use AWD-LSTM [1] and AWD-LSTM-MoS as experimental objects, and change the LSTM dimension of different layers, then record the perplexities of the language model on PTB and WT2 datasets. Except for the dimensions of the LSTM layer, the other hyper-parameters are the same as in Merity et al. (2017) and Yang et al. (2017). Moreover, we exclude the fine-tuning and dynamic evaluation steps and the experiments of AWD-LSTM-MoS on WT2 for saving computational resources and avoiding distractive factors. The results are reported on both Figure 2 and Figure 3.

[1]The third LSTM layer of AWD-LSTM is connected to the Softmax layer, whose input size is fixed after tying the word vectors and word classifier. So we only experiment over the first two LSTM layers.



(a) AWD-LSTM on PTB (b) AWD-LSTM on WT2 (c) MoS on PTB

Figure 2: The line graphs of models with changed LSTM layers dimensions perplexities on PTB and WT2. The solid line and the dotted line denote the perplexity variation trend of the first LSTM and the second LSTM layer respectively. And the dimensional interval of these three line graphs is 200.



(a) AWD-LSTM on PTB (b) AWD-LSTM on WT2 (c) MoS on PTB

Figure 3: The perplexity bands of models with changed LSTM layers dimensions on PTB and WT2. The standard dimension is denoted as c, and the dimension range of perplexity bands in (a) (b) and (c) is from 0.6c to 1.2c. In every perplexity band, the red box is used to mark the region within the range of [-0.25, 0.25] compared with the perplexity at dimension c (standard perplexity).

In order to obtain the overall trend of perplexities with dimensional changes in each LSTM layer, we firstly use three line graphs[2] to observe in a large dimension range from 200 to 1200 at intervals of 200. From these line graphs we can clearly see that with the increase of the dimension, the perplexity keeps decreasing, but the rate of decline is getting slower and slower. This means that the contribution of newly added parameters to the improvement of model performance decreases, so it is not worthwhile to strengthen the final effect by setting larger dimensions of LSTM layers. Moreover, we can also find that the dimensional change of the second LSTM layer has more influence than the first LSTM layer on the total model performance.

On the other hand, for intuitively verifying that the dimensional changes in a certain range for each LSTM layer will not cause obvious fluctuation in perplexity, we take the dimensions of each LSTM layer in Merity et al. (2017) and Yang et al. (2017) as standard, then draw the perplexity bands of models with dimensions from 0.6c to 1.2c in each layers at intervals of $0.1c(\approx 100)$. In each perplexity band, we mark the part whose perplexity differs from the standard perplexity within the range of [-0.25, 0.25] with a red box. We find the dimensions of 0.9c in the most layers are included in red boxes, and the red box of the third LSTM layer in AWD-LSTM-MoS includes the dimension of 0.6c. This demonstrates that we can greatly reduce the useless parameters and maintain the model performance by setting the dimension ratio of the Major LSTMs to 90% and 60% in the first two and the last MMLSTMs layers.

4.4 Study of Minor LSTM Input

In Section 3.2 we explained the reason why we use the original embeddings as the input of the Minor LSTMs instead of the output of the previous layer. To compare the influence of the two types inputs on the final results, we conduct experiments on PTB and WT2, while limiting the amount of parameters. We construct two AWD-MMLSTMs-MoS with the two types input of Minor LSTMs, and denote them as MMLM-1 (embedding input) and MMLM-2. The dimension of the first two LSTM layers are set to 1150 for MMLM-2, so that the total parameters of the two models

[2]Since the dimension of the third LSTM of AWD-LSTM-MoS is much smaller than the first two layers, it is not suitable to draw their dimensional changes in the same coordinate system, so we only plot the dimensional changes of the first two LSTM layers in the line graph.

Model	PTB			WT2		
	#Param	Val	Test	#Param	Val	Test
MMLM-1	**21.3M**	**56.52**	**54.51**	**32.3M**	**64.30**	**61.77**
MMLM-2	21.7M	57.95	55.93	32.5M	65.43	62.76

Table 5: The perplexities of MMLM-1 and MMLM-2 over PTB and WT2 datasets.

are roughly equal. The other hyper-parameters of MMLM-1 and MMLM-2 are the same, and we still remove the fine-tuning and dynamic evaluation steps. The results are shown in Table 5. The MMLM-1 surpasses MMLM-2 on both datasets steadily, which demonstrates using the embedding of input word sequence can better generate auxiliary features for Major-LSTM.

Model	PTB				WT2			
	AD	#Param	Val	Test	AD	#Param	Val	Test
MMLM-w/o-Minor-1	1030	21.5M	57.3	55.10	1150	32.5M	65.34	62.73
MMLM-w/o-Minor-2	1030	21.5M	57.14	54.93	1130	32.6M	65.07	62.43
MMLM-w/o-Minor-3	520	21.5M	57.84	55.74	550	32.6M	65.66	63.02
MMLM	-	21.3M	56.52	54.51	-	32.3M	64.3	61.77

Table 6: Ablation study on PTB and WT2 datasets without fine-tuning or dynamical evaluation. AD (Adjusted Dimension) means adjusted dimension of MMLSTM layer whose Minor LSTM is removed, and the columns with the title Val and Test denote the perplexities on the validation sets and test sets respectively.

4.5 Ablation Study

Through the above experiments, we can reach the conclusion that using MMLSTMs layers can control the amount of parameters effectively and improve the model performance. To further study the contribution of the Minor LSTM in each MMLSTMs layer, we conduct an ablation study on both PTB and WT2. We remove the Minor LSTM from each MMLSTMs layer by layer, and denote the model with removed layer as MMLM-w/o-Minor-i (i = 1, 2, 3). The dimension of the layer removed the Minor LSTM will be adjusted to keep total parameters consistent, and the other hyper-parameters are the same to ensure a single variable contrast experiment. We also exclude the fine-tuning and dynamic evaluation steps, and the results are reported in Table 6.

Comparing with our original three-layer MMLM, we can conclude that the Minor LSTM of the third layer has the greatest effect on the final performance of the model, followed by the first Minor LSTM and finally the second. In addition, this experiment demonstrates once again that MMLSTMs layer can not only reduce the amount of parameters in each layer, but also benefit the model performance.

5 Conclusions

We observed that when the scale of LSTM is large enough, the benefits to the vanilla LSTM-based language model of increasing the dimension of LSTM are very limited, while leading to a lot of parameter waste. In this paper, we studied the phenomenon systematically, then proposed to reduce the dimension of the Major LSTM in the original LSTM layer appropriately, and construct a small-scale Minor LSTM to extract a set of auxiliary features, so that the next layer can process the output of the Major LSTM better. In experiments, we verified the MMLM can control the total parameter counts effectively, and further improve the performance of the language model based on the existing methods such as regularization, optimization and Mixture of Softmax. Moreover, our method improved the current state-of-the-art results on standard benchmarks with less amount of parameters.

REFERENCES

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155, 2003.

Jan Botha and Phil Blunsom. Compositional morphology for word representations and language modelling. In *International Conference on Machine Learning*, pp. 1899–1907, 2014.

Stanley F Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, 13(4):359–394, 1999.

Yarin Gal and Zoubin Ghahramani. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in neural information processing systems*, pp. 1019–1027, 2016.

Edouard Grave, Armand Joulin, and Nicolas Usunier. Improving neural language models with a continuous cache. *arXiv preprint arXiv:1612.04426*, 2016.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

Jeremy Howard and Sebastian Ruder. Fine-tuned language models for text classification. *CoRR*, abs/1801.06146, 2018. URL `http://arxiv.org/abs/1801.06146`.

Hakan Inan, Khashayar Khosravi, and Richard Socher. Tying word vectors and word classifiers: A loss framework for language modeling. *arXiv preprint arXiv:1611.01462*, 2016.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*, 2016.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. Character-aware neural language models. In *AAAI*, pp. 2741–2749, 2016.

Reinhard Kneser and Hermann Ney. Improved backing-off for m-gram language modeling. In *icassp*, volume 1, pp. 181e4, 1995.

Philipp Koehn. *Statistical machine translation*. Cambridge University Press, 2009.

Ben Krause, Emmanuel Kahembwe, Iain Murray, and Steve Renals. Dynamic evaluation of neural sequence models. *arXiv preprint arXiv:1709.07432*, 2017.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330, 1993.

Gábor Melis, Chris Dyer, and Phil Blunsom. On the state of the art of evaluation in neural language models. *arXiv preprint arXiv:1707.05589*, 2017.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. Regularizing and optimizing lstm language models. *arXiv preprint arXiv:1708.02182*, 2017.

Tomas Mikolov and Geoffrey Zweig. Context dependent recurrent neural network language model. *SLT*, 12(234-239):8, 2012.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

Ofir Press and Lior Wolf. Using the output embedding to improve language models. *arXiv preprint arXiv:1608.05859*, 2016.

Alec Radford, Rafal Jozefowicz, and Ilya Sutskever. Learning to generate reviews and discovering sentiment. *arXiv preprint arXiv:1704.01444*, 2017.

Zhilin Yang, Zihang Dai, Ruslan Salakhutdinov, and William W Cohen. Breaking the softmax bottleneck: A high-rank rnn language model. *arXiv preprint arXiv:1711.03953*, 2017.

Dong Yu and Li Deng. *Automatic Speech Recognition: A Deep Learning Approach*. Springer, 2014.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*, 2014.

Julian Georg Zilly, Rupesh Kumar Srivastava, Jan Koutník, and Jürgen Schmidhuber. Recurrent highway networks. *arXiv preprint arXiv:1607.03474*, 2016.

Konrad Zolna, Devansh Arpit, Dendi Suhubdy, and Yoshua Bengio. Fraternal dropout. *arXiv preprint arXiv:1711.00066*, 2017.

Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.