Improving Neural Language Models with Weight Norm Initialization and Regularization

Christian Herold* **Yingbo Gao*** **Hermann Ney**
Human Language Technology and Pattern Recognition Group
Computer Science Department
RWTH Aachen University
D-52056 Aachen, Germany
<surname>@i6.informatik.rwth-aachen.de

Abstract

Embedding and projection matrices are commonly used in neural language models (NLM) as well as in other sequence processing networks that operate on large vocabularies. We examine such matrices in fine-tuned language models and observe that a NLM learns word vectors whose norms are related to the word frequencies. We show that by initializing the weight norms with scaled log word counts, together with other techniques, lower perplexities can be obtained in early epochs of training. We also introduce a weight norm regularization loss term, whose hyperparameters are tuned via a grid search. With this method, we are able to significantly improve perplexities on two word-level language modeling tasks (without dynamic evaluation): from 54.44 to 53.16 on Penn Treebank (PTB) and from 61.45 to 60.13 on WikiText-2 (WT2).

1 Introduction

A language model (LM) measures how likely a certain sequence of words is for a given language. It does so by calculating the probability of occurrence of that sequence, which can be learned from monolingual text data. Many models in machine translation and automatic speech recognition benefit from the use of a LM (Corazza et al., 1995; Peter et al., 2017).

While count-based LMs (Katz, 1987; Kneser and Ney, 1995) provided the best results in the past, substantial improvements were achieved with the introduction of neural networks in the field of language modeling (Bengio et al., 2003). Different types of architectures such as feedforward neural networks (Schwenk, 2007) and recurrent neural networks (Mikolov et al., 2010) have since been used for language modeling. Currently, variants of long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) networks give the best results on popular language modeling tasks (Yang et al., 2018).

In natural language processing, words are typically represented by high-dimensional one-hot vectors. To reduce dimensionality and to be able to learn relationships between words, they are mapped into a lower-dimensional, continuous embedding space. Mathematically, this is done by multiplying the one-hot vector with the embedding matrix. Similarly, to receive a probability distribution over the vocabulary, a mapping from an embedding space is performed by a projection matrix followed by a softmax operation. These two matrices can be tied together in order to reduce the number of parameters and improve the results of NLMs (Inan et al., 2017; Press and Wolf, 2017).

Since the row vectors in the embedding and projection matrices are effectively word vectors in a continuous space, we investigate such weight vectors in well-trained and fine-tuned NLMs. We observe that the learned word vector generally has a greater norm for a frequent word than an infrequent word. We then specifically examine the weight vector norm distribution and design initialization and normalization strategies to improve NLMs.

Our contribution is twofold:

- We identify that word vectors learned by NLMs have a weight norm distribution that resembles logarithm of the word counts. We then correspondingly develop a weight initialization strategy to aid NLM training.
- We design a weight norm regularization loss term that increases the generalization ability of the model. Applying this loss term, we achieve state-of-the-art results on Penn Treebank (PTB) and WikiText-2 (WT2) language modeling tasks.

*Equal contribution. Ordering determined by coin flipping.

Proceedings of the Third Conference on Machine Translation (WMT), Volume 1: Research Papers, pages 93–100

https://doi.org/10.18653/v1/W18-64010

2 Related Work

Melis et al. (2018) investigated different NLM architectures and regularization methods with the use of a black-box hyperparameter tuner. In particular, the LSTM architecture was compared to two more recent recurrent approaches, namely recurrent highway networks (Zilly et al., 2017) and neural architecture search (Zoph and Le, 2017). They found that the standard LSTM architecture outperforms other models, if properly regularized.

Merity et al. (2017a) used various regularization methods such as activation regularization (Merity et al., 2017b) in a LSTM model. They also introduced a variant of the averaged stochastic gradient method, where the averaging trigger is not tuned by the user but relies on a non-monotonic condition instead. With these and further regularization and optimization methods, improved results on PTB and WT2 were achieved.

To further improve this network architecture, Yang et al. (2018) introduced the mixture of softmaxes (MoS) model, claiming that the calculation of the output probabilities with a single softmax layer is a bottleneck. In their approach, several output probabilities are calculated and then combined via a weighted sum. The LSTM-MoS architecture provides state-of-the-art results on PTB and WT2 at the time of writing and is used as the baseline model for comparisons in this work.

Other works proposed to tie the embedding and projection matrices. Press and Wolf (2017) investigated the effects of weight tying, analyzed update rules after tying and showed that tied matrices evolve in a similar way as the projection matrix. Inan et al. (2017) were motivated by the fact that with a classification setup over the vocabulary, inter-word information is not utilized to its full potential. They also provided theoretical justification on why it is appropriate to tie the above-mentioned matrices.

Besides using the word embedding matrix, there are other approaches to represent word sequences. Zhang et al. (2015) proposed a new embedding method called fixed-sized ordinally-forgetting encoding (FOFE), which allows them to encode variable-length sentences into fixed-length vectors almost uniquely.

Additionally, Salimans and Kingma (2016) introduced a weight normalization reparametrization trick on weight matrices, which separates the norm and the angle of a vector. This can speed up the convergence of stochastic gradient descent and also allows for explicit scaling of gradients in the amplitude and direction. They also discussed the connections between weight normalization and batch normalization.

On top of one-hot representations of words, Irie et al. (2015) used additional information to represent word sequences. It is shown that the use of long-context bag-of-words as additional feature for language modeling can narrow the gap between feed-forward NLMs and recurrent NLMs.

3 Neural Language Modeling

In NLM the probability of a word sequence $\boldsymbol{x}_1^t = x_1 x_2 ... x_t$ is decomposed as

$$P(\boldsymbol{x}_1^t) = \prod_{j=1}^{t} P(x_j | \boldsymbol{x}_{j-n+1}^{j-1}) \quad (1)$$

so that the $(n-1)$ preceding words $\boldsymbol{x}_{j-n+1}^{j-1}$ are considered for the prediction of the next word x_j. This is typically done by using a recurrent neural network, e.g. a stack of LSTM layers, to encode the input sequence as

$$h_t = \text{LSTM}(E^T[x_{t-n+1}, x_{t-n+2}, ..., x_{t-1}]) \quad (2)$$

where E^T is the transposed embedding matrix, $[x_{t-n+1}, x_{t-n+2}, ..., x_{t-1}]$ are the one-hot encoded preceding words and the LSTM() function returns the last hidden state of the last LSTM layer. The probability distribution over the next word x_t is then calculated as

$$P(x_t = x_k | h_t) = \frac{\exp(W_k h_t)}{\sum_{j=1}^{V} \exp(W_j h_t)} \quad (3)$$

with V being the vocabulary size, $k = 1, 2, ..., V$, and W_k being the k-th row vector in the projection matrix W.

For training the neural network, the cross-entropy error criterion, which is equivalent to the maximum likelihood criterion, is used. For the i-th sequence of words $\boldsymbol{x}_1^{t_i}$, the cross-entropy loss L_i is defined as

$$L_i = -\log P(x_{t_i} = x_{y_i} | h_{t_i}) \quad (4)$$

with y_i being the true label of x_{t_i}. The total loss is then calculated as

$$L = \frac{1}{N} \sum_{i=1}^{N} L_i \quad (5)$$

where N is the total number of sequences. A language model is normally scored by perplexity (ppl). For a given test corpus $\boldsymbol{x}_1^T = x_1 x_2 ... x_T$, the ppl is calculated as

$$ppl = P(\boldsymbol{x}_1^T)^{-\frac{1}{T}} \tag{6}$$

which is a measurement on how likely a given sentence is, according to the prediction of the model.

In the above formulation, we have an embedding matrix E and a projection matrix W. When the two matrices are tied and one-hot vectors are used to represent words, the rows of these matrices are then the word vectors of the corresponding words. Particularly, we focus on the norms of the row vectors and study their relationship with word counts and how to regularize them.

4 Weight Norm Initialization

We first train models on PTB and WT2 as described in (Yang et al., 2018) and plot the norms of learned weight vectors of the embedding matrix in Figure 1.

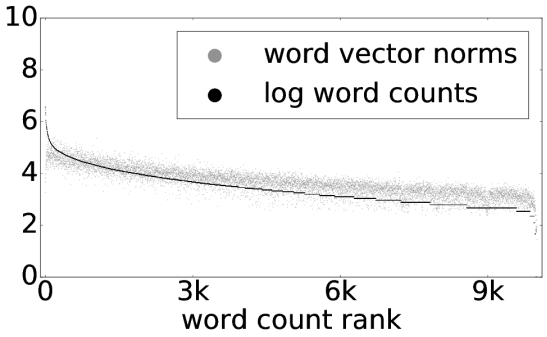
When the words are ranked by their counts and placed on the x-axis from frequent to infrequent, it can be seen that the word vector norms follow a downward trend as well. Log unigram counts are also plotted for comparison. As can be seen, the norm distribution follows a similar trend as the log counts. It is important to note, that the logit for word x_k and context h_t is calculated as $W_k h_t$ (see Equation 3), which can be rewritten as

$$W_k h_t = \|W_k\| \, \|h_t\| \cos(\theta) \tag{7}$$

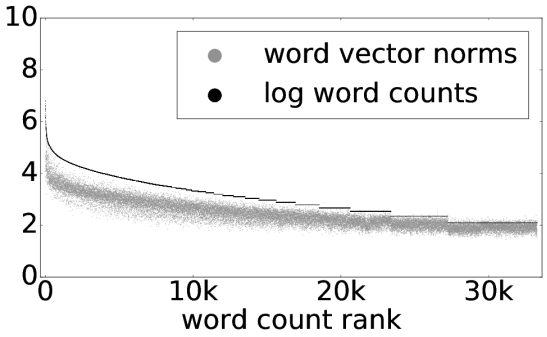
where θ denotes the angle between W_k and h_t. Therefore, one intuition from the aforementioned observation is that, for a frequent word, the network tends to learn a weight vector W_k with a greater norm to maximize likelihood. This motivates our approach to initialize the weight norms with scaled log counts rather than uniformly random values in a specific range.

Because we wish to initialize the weight norms explicitly with scaled logarithm of the word counts, it is helpful to look at a weight vector's magnitude and direction separately. For this purpose, we use a reparameterization technique on the weight vectors as described in (Salimans and Kingma, 2016):

$$W_k = g_k \frac{v_k}{\|v_k\|_2} \tag{8}$$



(a) Penn Treebank



(b) WikiText-2

Figure 1: Word vector norms of fine-tuned MoS models (Yang et al., 2018), trained on (a) Penn Treebank and (b) WikiText-2. Words are ranked by their counts in a descending order and thus frequent words are to the left. Actual logarithm of word counts are plotted in black, and word vector norms are grey. We observe that word vector norms loosely follow the trend of log counts.

where $k = 1, 2, ..., V$, $g_k = \|W_k\|_2$, and v_k is a vector proportional to W_k. Reparameterizing the weight vectors makes it easy to implement the weight norm initialization as

$$g_k = \sigma \log c_k \tag{9}$$

where c_k denotes unigram word count for word k and σ is a scalar applied to the log counts. We sample each component of v_k from a continuous uniform distribution in $[-r, r]$, where r is a hyperparameter, specifying the initialization range. With this, no constraint on the weight vector direction is imposed during initialization.

Additionally, we adopt an adaptive gradient strategy which regularizes the gradients in g_k. As in

$$(\frac{\partial L}{\partial g})' = \begin{cases} [1 - (1-\gamma)\frac{t}{\tau}]\frac{\partial L}{\partial g}, & \text{for } t \leq \tau \\ \gamma \frac{\partial L}{\partial g}, & \text{for } t > \tau \end{cases} \tag{10}$$

when epoch t is no greater than a specified epoch

		Tokens	Vocab Size
Penn Treebank	Train	888k	10k
	Valid	70k	
	Test	79k	
WikiText-2	Train	2.1M	33k
	Valid	214k	
	Test	241k	

Table 1: Statistics of the Penn Treebank and WikiText-2 datasets.

τ, $(\frac{\partial L}{\partial g_k})'$ — the regularized gradient in g_k, linearly decays to γ ($\gamma \leq 1$) times the unregularized gradient $\frac{\partial L}{\partial g_k}$. Otherwise, we directly use the discounted gradient. In analogy to learning rate decay, this adaptive gradient strategy anneals the word vector norm updates in each step. The intuition for such a strategy is that after a certain amount of epochs, the weight norms should not change so drastically from the initialized scaled log counts.

5 Weight Norm Regularization

Weight regularization (WR) is a well established method to combat overfitting in neural networks, which is especially important on smaller datasets (Krogh and Hertz, 1992). The idea is to push weights in the network to zero, where gradients are not significant. Typically, WR is implemented by adding an extra term to the loss function L_0, which penalizes the norm of all weights in the network. For example, L_2-regularization is implemented as

$$L = L_0 + \frac{\lambda}{2} \sum_{w} (\|w\|_2)^2 \qquad (11)$$

with the sum going over all weights w in the network and λ being the regularization strength. However, this method is not perfect, as it affects every weight in the network equally and may lead to hidden units' weights getting stuck near zero.

In this work we add a constraint specifically on the embedding and projection matrices, whose weights are shared. Since the row vectors in both matrices are word vectors, it seems appropriate to put constraints explicitly on their norms instead of on each individual weight parameter in the matrices.

We propose to add a regularization term to the standard loss function L_0 in the form of

$$L_{wr} = L_0 + \rho \sqrt{\sum_{j=1}^{V} (\|W_j\|_2 - \nu)^2} \qquad (12)$$

where ν, $\rho \geq 0$ are two scalars and W_j is the j-th row vector of the projection matrix W. The L_2-norms of the row vectors are pushed towards ν, while ρ is the regularization strength. This will punish the row vectors for adopting norms other than ν, in the hope of reducing the effect of overfitting on the training data.

The choice of a soft regularization loss term instead of hard-fixing the weight norms in the forward pass is motivated by the weight norm distribution shown in Figure 1. It can be seen that NLMs tend to learn non-equal weight norms for words with different counts. Therefore, hard-fixing weight norms may limit the network's ability to learn.

6 Experiments

6.1 Experiment Setup

The experiments are conducted on two popular language modeling datasets. The number of tokens and size of vocabulary for each dataset are summarized in Table 1.

epoch	Penn Treebank			WikiText-2		
	wni *ppl*	baseline *ppl*	*ppl* reduction (%)	wni *ppl*	baseline *ppl*	*ppl* reduction (%)
1	162.18	180.72	10.26	172.19	192.19	10.41
10	85.92	92.09	6.70	95.90	100.72	4.79
20	73.36	78.94	7.07	85.14	88.21	3.48
30	71.44	73.06	2.22	81.80	82.70	1.09
40	69.27	70.20	1.32	79.28	80.32	1.29

Table 2: Perplexity (*ppl*) improvement using weight norm initialization (wni) in early epochs on Penn Treebank and WikiText-2. *ppl* reduction is around 10% after the first epoch on both tasks, and decays to approximately 1% after 40 epochs. The wni model has slightly higher perplexities than the baseline model from around 50 epochs onward.

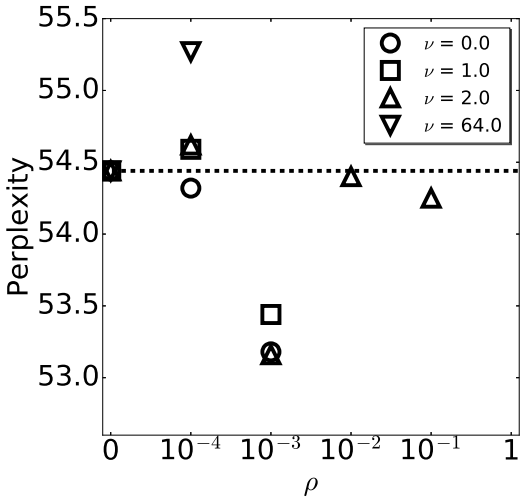
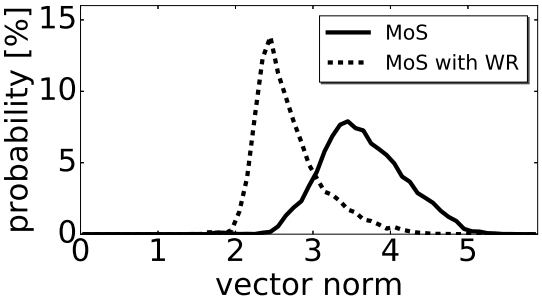
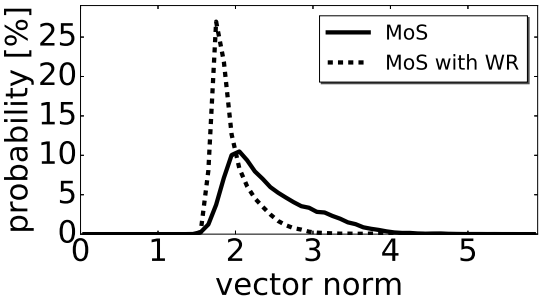


Figure 2: Model perplexity on the Penn Treebank test set as a function of ρ. The different symbols denote different values of ν. Models not depicted yield higher perplexity values. The doted line marks the baseline result (with $\rho = 0$) as reported by Yang et al. (2018).



(a) Penn Treebank



(b) WikiText-2

Figure 3: Weight norm distributions of the projection matrices' row vectors for the AWD-LSTM-MoS model from Yang et al. (2018) as well as for our regularized version (WR). The models are trained on the (a) Penn Treebank corpus and (b) WikiText-2 corpus with the resulting test perplexities shown in Table 3 and Table 4 respectively.

The smaller one is the PTB corpus with preprocessing from Mikolov et al. (2010), which has a comparatively small vocabulary size of 10k. With a smaller number of sentences, this dataset is a good choice for performing optimization of hyper-parameters. The second corpus WT2, which was introduced by Merity et al. (2016), has over three times the vocabulary size of PTB.

We use the network structure introduced by Yang et al. (2018) with the same hyper-parameter values to ensure comparability. Several regularization techniques are used in this setup, such as dropout and weight decay. Furthermore, the embedding and projection matrices are tied by default. For optimization, we adopt the same strategy as described in (Merity et al., 2017a). That is, a conservative non-monotonic criterion is used to switch from stochastic gradient descent (SGD) to averaged stochastic gradient descent (ASGD) (Polyak and Juditsky, 1992). For more details of the network structure refer to (Yang et al., 2018).

6.2 Weight Norm Initialization

We tune the hyperparameter σ and use a value of $\sigma = 0.5$ to scale the logarithm of word counts. Initialization range r is set to 0.1 for both the reparametrized direction vectors and the baseline word vectors. Empirically, we set $\gamma = 0.1$ and $\tau = 100$ for the adaptive gradient method. Perplexities on both PTB and WT2 in early epochs, as well as the relative perplexity improvement over baseline models are summarized in Table 2.

First, we notice significant improvement after the first epoch of training using weight norm initialization. About 10% of perplexity reduction is achieved on both datasets. This could be beneficial, when one wants to train on large datasets and/or can only train for a limited number of epochs. Second, the perplexity improvements decay down to around 1% after 40 epochs. This is in agreement with our expectation, because apart from reduced gradient in g_k, a weight norm initialized model is not fundamentally different from the baseline model and no major difference should be seen if we train for long enough. It is important to note that with only weight norm initialization, both models eventually converge to perplexities that are slightly worse than the baseline. We also notice that the epochs, after which the optimizer is switched from SGD to ASGD, are different in weight norm initialized models and baseline models.

Model	#Params	Validation	Test
Mikolov and Zweig (2012) - RNN-LDA + KN + cache	9M	-	92.0
Zaremba et al. (2014) - LSTM	20M	86.2	82.7
Gal and 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. (2017) - LSTM + continuous cache pointer†	-	-	72.1
Inan et al. (2017) - Tied Variational LSTM + augmented loss	24M	75.7	73.2
Zilly et al. (2017) - Variational RHN	24M	75.7	73.2
Zoph and Le (2017) - NAS Cell	25M	-	64.0
Melis et al. (2018) - 2-layer skip connection LSTM	24M	60.9	58.3
Merity et al. (2017a) - AWD-LSTM	24M	60.0	57.3
Yang et al. (2018) - AWD-LSTM-MoS	22M	56.54	54.44
Ours - AWD-LSTM-MoS with weight norm regularization	22M	**55.03**	**53.16**

Table 3: Single model perplexity on the Penn Treebank test and validation sets. Baseline results are obtained from (Yang et al., 2018). † indicates the use of dynamic evaluation.

Model	#Params	Validation	Test
Inan et al. (2017) - Variational LSTM + augmented loss	28M	91.5	87.0
Grave et al. (2017) - LSTM + continuous cache pointer†	-	-	68.9
Melis et al. (2018) - 2-layer skip connection LSTM	24M	69.1	65.9
Merity et al. (2017a) - AWD-LSTM	33M	69.1	66.0
Yang et al. (2018) - AWD-LSTM-MoS	35M	63.88	61.45
Ours - AWD-LSTM-MoS with weight norm regularization	35M	**62.67**	**60.13**

Table 4: Single model perplexity on the WikiText-2 test and validation sets. Baseline results are obtained from (Yang et al., 2018). † indicates the use of dynamic evaluation.

6.3 Weight Norm Regularization

In order to tune the hyperparameters ρ and ν introduced in Section 5, we perform a grid search over the PTB dataset, the results of which are shown in Figure 2. If the norm constraint ν becomes too large, perplexity worsens significantly, as seen in the case of $\nu = 64$. A model with a ν-value of 2 provides the best result in most cases. We hypothesize that a value of ν that is too small results in the logit being close to zero as shown in Equation 7. For the regularization strength ρ, we recognize that $\rho = 10^{-3}$ gives the best result on the PTB test data. Larger or smaller values can hurt the performance of the system, depending also on the value of ν. It should be noted that the optimized value of ρ is significantly larger than the scaling s_{wd} of the weight decay term, which was optimized to be 1.2×10^{-6} by Merity et al. (2017a).

The resulting weight norm distributions of the projection matrices' row vectors are shown in Figure 3a and Figure 3b for models trained on PTB and WT2 respectively. Our efforts of pushing the norms to a value of $\nu = 2.0$ resulted in a noticeably smaller average norm, as well as in a overall more narrow distribution.

With the tuned parameter values $\rho = 10^{-3}$ and $\nu = 2.0$ we improve the previous state-of-the-art result by 1.28 *ppl* on PTB and by 1.32 *ppl* on WT2 (without considering dynamic evaluation (Krause et al., 2018), see Table 3 and Table 4). This is achieved without increasing the number of trainable parameters in the network or slowing down the training process.

7 Conclusion

Word embedding matrix and output projection matrix are important components in LSTM-based LMs. They are also widely used in other NLP models where one-hot vectors of words need to be mapped into lower dimensional space. Given the one-hot nature of word representations, row vectors in such matrices are then the correspond-

ing word vectors. We study specifically the norms of these learned word vectors, the distribution of the norms, and the relationship with word counts. We show that with a simple initialization strategy together with a reparametrization technique, it is possible to get significantly lower perplexity in early epochs during training. By using a weight norm regularization loss term, we are able to obtain significant improvements on standard language modeling tasks — 2.4% *ppl* reduction on PTB and 2.1% on WT2.

We propose three directions to investigate further. First, in this work we use scaled logarithm of word counts to initialize the weight norms. It is a logical next step to use smoothing techniques on the word counts and study the effects of such initializations. Second, we currently apply the same norm constraint on different words. Altering the loss function and regularizing the weight norms to word counts (and smoothed word counts) is worth examining as well. Finally, our focus so far is on weight norms. It is a more exciting and challenging task to study the pairwise inner products, and single out the effects of angular differences.

We also plan to expand our regularization and initialization techniques to the field of neural machine translation. Embedding and projection matrices are also present in neural machine translation networks, which could potentially benefit from our methods as well. It seems natural to use our methods on the transformer architecture introduced by Vaswani et al. (2017), in which the embedding matrices at source and target sides, plus the projection matrix, are three-way tied.

Acknowledgments

This work has received funding from the European Research Council (ERC) (under the European Union's Horizon 2020 research and innovation programme, grant agreement No 694537, project "SEQCLAS") and the Deutsche Forschungsgemeinschaft (DFG; grant agreement NE 572/8-1, project "CoreTec"). The GPU computing cluster was supported by DFG (Deutsche Forschungsgemeinschaft) under grant INST 222/1168-1 FUGG.

The work reflects only the authors' views and none of the funding agencies is responsible for any use that may be made of the information it contains.

References

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. In *Journal of machine learning research*, volume 3, pages 1137–1155.

A. Corazza, R. De Mori, R. Gretter, R. Kuhn, and G. Satta. 1995. Language models for automatic speech recognition. In *Speech Recognition and Coding*, pages 157–173. Springer.

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

Edouard Grave, Armand Joulin, and Nicolas Usunier. 2017. Improving neural language models with a continuous cache. In *International Conference on Learning Representations*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9:1735–80.

Hakan Inan, Khashayar Khosravi, and Richard Socher. 2017. Tying word vectors and word classifiers: A loss framework for language modeling. In *International Conference on Learning Representations*.

Kazuki Irie, Ralf Schlüter, and Hermann Ney. 2015. Bag-of-words input for long history representation in neural network-based language models for speech recognition. In *Sixteenth Annual Conference of the International Speech Communication Association*.

Slava Katz. 1987. Estimation of probabilities from sparse data for the language model component of a speech recognizer. In *IEEE transactions on acoustics, speech, and signal processing*.

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

Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for m-gram language modeling. In *1995 International Conference on Acoustics, Speech, and Signal Processing*, pages 181–184.

Ben Krause, Emmanuel Kahembwe, Iain Murray, and Steve Renals. 2018. Dynamic evaluation of neural sequence models. In *Proceedings of the 35th International Conference on Machine Learning*, pages 2766–2775.

Anders Krogh and John A. Hertz. 1992. A simple weight decay can improve generalization. In *Advances in neural information processing systems*, pages 950–957.

Gábor Melis, Chris Dyer, and Phil Blunsom. 2018. On the state of the art of evaluation in neural language models. In *International Conference on Learning Representations*.

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

Stephen Merity, Bryan McCann, and Richard Socher. 2017b. Revisiting activation regularization for language rnns. *arXiv preprint arXiv:1708.01009*.

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

Tomas Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *11th Annual Conference of the International Speech Communication Association*.

Tomas Mikolov and Geoffrey Zweig. 2012. Context dependent recurrent neural network language model. In *IEEE Spoken Language Technology Workshop*, pages 234–239.

Jan-Thorsten Peter, Andreas Guta, Tamer Alkhouli, Parnia Bahar, Jan Rosendahl, Nick Rossenbach, Miguel Graça, and Hermann Ney. 2017. The RWTH Aachen University english-german and german-english machine translation system for WMT 2017. In *Proceedings of the Second Conference on Machine Translation*, pages 358–365.

Boris T. Polyak and Anatoli B. Juditsky. 1992. Acceleration of stochastic approximation by averaging. In *SIAM Journal on Control and Optimization*, pages 838–855.

Ofir Press and Lior Wolf. 2017. Using the output embedding to improve language models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 157–163.

Tim Salimans and Diederik P Kingma. 2016. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *Advances in Neural Information Processing Systems*, pages 901–909.

Holger Schwenk. 2007. Continuous space language models. *Computer Speech & Language*, 21(3):492–518.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Zhilin Yang, Zihang Dai, Ruslan Salakhutdinov, and William W. Cohen. 2018. Breaking the softmax bottleneck: A high-rank rnn language model. In *International Conference on Learning Representations*.

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

Shiliang Zhang, Hui Jiang, Mingbin Xu, Junfeng Hou, and Lirong Dai. 2015. The fixed-size ordinally-forgetting encoding method for neural network language models. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, pages 495–500.

Julian Georg Zilly, Rupesh Kumar Srivastava, Jan Koutník, and Jürgen Schmidhuber. 2017. Recurrent highway networks. In *Proceedings of the 34th International Conference on Machine Learning*, pages 4189–4198.

Barret Zoph and Quoc V Le. 2017. Neural architecture search with reinforcement learning. In *International Conference on Learning Representations*.