On Training Bi-directional Neural Network Language Model with Noise Contrastive Estimation

Tianxing He[1]*, Yu Zhang*[2]*, Jasha Droppo*[3]*, Kai Yu*[1]

[1] Shanghai Jiao Tong University [2]MIT CSAIL [3]Microsoft Research

cloudygoose@sjtu.edu.cn, yzhang87@csail.mit.edu, jdroppo@microsoft.com, kai.yu@sjtu.edu.cn

Abstract

Although uni-directional recurrent neural network language model(RNNLM) has been very successful, it's hard to train a bi-directional RNNLM properly due to the generative nature of language model. In this work, we propose to train bi-directional RNNLM with noise contrastive estimation(NCE), since the properties of NCE training will help the model to achieve sentence-level normalization. Experiments are conducted on two hand-crafted tasks on the PTB data set: a rescore task and a sanity test. Although(regretfully), the model trained by NCE did not out-perform the baseline uni-directional NNLM, it is shown that NCE-trained bi-directional NNLM behaves well in the sanity test and outperformed the one trained by conventional maximum likelihood training on the rescore task.

Index Terms: Language model, recurrent neural network, noise contrastive estimation

1. Introduction

Recent years have witnessed exciting performance improvements in the field of language modeling, largely due to introduction of a series of neural network language models(NNLM). Although the conventional back-off n-gram language model has been widely used in the automatic speech recognition (ASR) or machine translation(MT) community for its simplicity and effectiveness, it has suffered from the *curse-of-dimensionality* problem caused by huge number of possible word combinations in real-world text. Recently, neural network based language models have attracted great interest due to its effective encoding of word context history [1, 2, 3, 4].In neural network based language models, the word context is projected into a continuous space and the projection, represented by the transformation matrices in the NN, are learned during training. The projected continuous word vectors are also referred to as *word embeddings*. With the continuous context representation, feed-forward neural network language models (FNNLM)[1, 2, 3, 4, 5], have achieved both better perplexity(PPL) and better word error rate (WER) when embedded into a real-world system.

Despite the benefits of effective context representation brought by word embeddings, FNNLM is still a short-span language model and not capable of utilizing long-term word history for the target word prediction. To address this issue, recurrent neural network language model (RNNLM), which introduces a recurrent connection in the hidden layer, is proposed to preserve long-term context. It has achieved significant performance gain on perplexity and word error rate (WER) performance on various data sets [6, 7, 8, 9, 10, 11], out-performing traditional back-off n-gram models and FNNLMs.

However, RNN training generally suffered from the "vanishing gradient" problem[12]:the gradient flow will decay sharply through a non-linear operation. The LSTM[13] structure alleviates this problem by introducing a "memory cell" structure which allows the gradient to travel without being squashed by a non-linear operation. Also, it has a set of gates which enable the model to decide whether to memorize, forget, or output information. By introducing the LSTM strucutre into RNNLM[8, 14], LSTMLM is able to remember longer context information and get more performance gain. Inspired by its success, several variants of LSTM have been proposed, recently the gated recurrent unit(GRU)[15] is gaining increasing popularity becuase it has matching performance with LSTM but has simpler structure. More recently, [16] has proposed to introduce the concept of memory into NNLM. By fetching memories from previous time, the model is able to "explicitly" utilizing long-term dependency without recurrence structure.

While these research efforts have been focusing on better utilization of history information, it would be desirable if the model can utilize context information from both sides. In literature, very few attempts have been made to train a proper bi-directional neural network language model, even though bi-directional NN has already been successfully applied to other fields[17]. This is because the bi-directional model won't be by itself normalized because of the generative nature of language model, which makes the conventional maximum likelihood training framework improper for its training.

In this work, attempts have been made to train a bi-directional neural network language model with noise contrastive estimation, an alternative to maximum likelihood training, which does not have the constrain that the model to be trained needs to be inherently normalized. The rest of the paper is organized as follows: in section 2, the motivation of this work is discussed, in section 3 the formulation of the model are elaborated in detail, implementation is covered in section 4, finally experiment results are shown in section 5 and related works are discussed in section 6.

2. Motivation

Statistical language models assign a probability $P(\mathcal{W})$ to a given sentence $\mathcal{W} =< w_1, w_2, ..., w_n >$, which can be decomposed into a product of word-level probabilities using the rule of conditional probability:

$$P(\mathcal{W}) = \Pi_i P(w_i|w_{1..i-1}) \quad (1)$$

Language models by this formulation predict the probabil-

This work was supported by the Shanghai Sailing Program No. 16YF1405300, the Program for Professor of Special Appointment (Eastern Scholar) at Shanghai Institutions of Higher Learning, the China NSFC projects (No. 61573241 and No. 61603252) and the Interdisciplinary Program (14JCZ03) of Shanghai Jiao Tong University in China.

ity distribution of the next word given its former words(history). Since the prediction only depends on history information, in this work, this kind of model is denoted as uni-directional language model. All types of language model mentioned in section 1 fall into this category, but note that shot-span models like N-gram makes the "Markov Chain" assumption $P(w_i|w_{1..i-1}) \approx P(w_i|w_{i-N..i-1})$ to alleviate the data-sparsity problem.

For uni-directional language models, as long as each word-level probability is properly normalized, normalization is also guaranteed on sentence level:

$$\sum_{\mathcal{W}} P_{LM}(\mathcal{W}) = 1 \tag{2}$$

This is the key reason why the "maximum likelihood estimation" training framework, which requires the model to be inherently probabilistic, has been successfully applied to the parameter estimation(training) of uni-directional language models. And recent years of research effort in the field of neural network language model has been focused on getting a better representation of history context using sophisticated recurrent neural network structures like LSTM[8].

Unfortunately, while recently bi-directional neural network like BI-RNN or BI-LSTM has been successfully applied to many tasks, it is not trivial to apply this powerful model to language modeling, the main challenge is that the bi-directional information will break the sentence-level normalization[1], making the model no longer valid for the MLE training framework(please refer to section 3 for more details).

In this work, noise contrastive estimation(NCE)[18] is used to train a bi-directional neural network based LM, one big advantage of NCE over MLE is that it doesn't require the model to be self-normalized. This enables the utilization of bi-directional information for word-level scoring. Formulations of this work will be elaborated in the next section.

3. Formulation

3.1. Model Formulation

In this work, $P(\mathcal{W})$ consists of the product of word-level scores(similar to uni-directional LM) and a learned normalization scalar c, required by the NCE framework to ensure normalization:

$$\begin{aligned} f'(\mathcal{W}) &= \Pi_i f_i(\mathcal{W}) \\ P^{NCE}(\mathcal{W}) &= f'(\mathcal{W}) exp(c) \end{aligned} \tag{3}$$

where f_i the scoring given by a bi-directional neural network on each word index. And the "NCE" superscript for $P^{NCE}(\mathcal{W})$ is for indicating the normalization is induced by NCE training.

In this work, the same bi-directional neural network structure that has been used in [17, 19] is applied, and is shown in figure 1 and formulated below(we are aware that other variants of BI-RNN exist[20], but they are not fundamentally different

[1]If some bi-directional model like $P(w_i|w_{1..i-1,i+1..N})$ is used as the word-level LM, equation 1, and hence equation 2 won't hold any more.

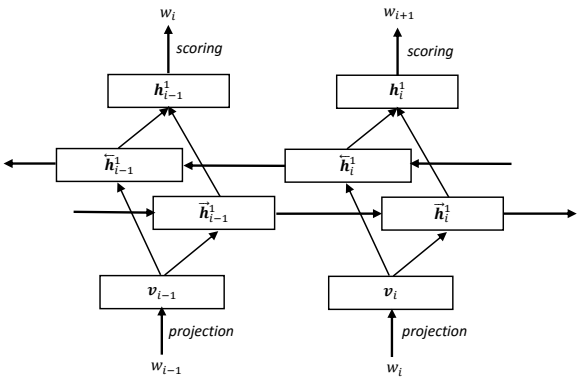

Figure 1: Illustration of the network structure

with regard to this work):

$$\begin{aligned} \mathbf{v}_i &= \mathbf{W}_{xh}\mathbf{x}_i \\ \overrightarrow{\mathbf{h}}^1_i &= g(\overrightarrow{\mathbf{h}}^1_{i-1}, \mathbf{v}_i) \\ \overleftarrow{\mathbf{h}}^1_i &= g(\overleftarrow{\mathbf{h}}^1_{i+1}, \mathbf{v}_i) \\ \mathbf{h}^1_i &= tanh(\mathbf{W}^1_{hf}\overrightarrow{\mathbf{h}}^1_i + \mathbf{W}^1_{hr}\overleftarrow{\mathbf{h}}^1_i + \mathbf{b}^1) \\ \mathbf{u}_i &= exp(\mathbf{W}_{ho}\mathbf{h}^1_i + \mathbf{b}_o) \end{aligned} \tag{4}$$

where $\mathbf{W}_{**}$ and $\mathbf{b}_*$ are the transformation matrices and bias vector parameters in the neural network, and $\mathbf{x}_i$ is the one-hot representation of w_i. Finally, $f_i(\mathcal{W})$ is obtained after a normalizing operation over the vocabulary(denoted as $\mathcal{V}$) on $\mathbf{u}_i$:

$$f_i(\mathcal{W}) = \frac{\mathbf{u}_i(w_i)}{\sum_{w_j \in V} \mathbf{u}_i(w_j)} \tag{5}$$

Note that the word-level normalization is not needed in this work($\mathbf{u}_i(w_i)$ can be used directly as $f_i(\mathcal{W})$), but experiments show that reserving the word-level normalization will give better results.

In this work, gated recurrent unit is used as the recurrent structure $\mathbf{h}_t = g(\mathbf{h}_{t-1}, \mathbf{v}_t)$ because it is faster, causes less memory and has matching performance with the LSTM structure[15]. So our NN model is denoted **BI-GRULM** and the formulation is put below:

$$\begin{aligned} \mathbf{z}_t &= \sigma(\mathbf{W}_{hz}\mathbf{h}_{t-1} + \mathbf{W}_{xz}\mathbf{v}_t + \mathbf{b}_z) \\ \mathbf{r}_t &= \sigma(\mathbf{W}_{hr}\mathbf{h}_{t-1} + \mathbf{W}_{xr}\mathbf{v}_t + \mathbf{b}_r) \\ \tilde{\mathbf{h}}_t &= tanh(\mathbf{W}_{h1}(\mathbf{r}_t * \mathbf{h}_{t-1}) + \mathbf{W}_{h2}\mathbf{v}_t + \mathbf{b}_h) \\ \mathbf{h}_t &= (1 - \mathbf{z}_t) * \mathbf{h}_{t-1} + \mathbf{z}_t * \tilde{\mathbf{h}}_t \end{aligned} \tag{6}$$

where σ is the sigmoid function $\sigma(x) = \frac{1}{1+e^{-x}}$ and $*$ is element-wise multiplication, and note that a different set of parameter is used for forward and backward connections in the bi-directional neural network.

Finally, we also write down the formulations of a one-layer uni-directional GRULM(UNI-GRULM) here since it will be used as baseline model:

$$\begin{aligned} \mathbf{h}^1_i &= g(\mathbf{h}^1_{i-1}, \mathbf{W}_{xh}\mathbf{x}_i) \\ \mathbf{u}_i &= exp(\mathbf{W}_{ho}\mathbf{h}^1_i + \mathbf{b}_o) \end{aligned} \tag{7}$$

Note that other than being uni-directional, the only other difference between these two models is the normalization scalar c. And in this work, the dropout operation is applied on $\mathbf{h}^1_i$ for both models.

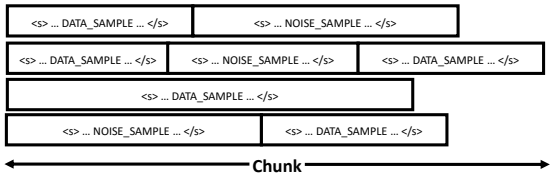

Figure 2: Illustration of the parallel training implementation

3.2. Training of bi-directional NNLM

As stressed in section 2, the MLE framework is not suitable for training bi-directional NNLM, still, in this work MLE training is tried as a baseline experiment. Denoting the the data distribution as $P_{data}(\mathcal{W})$, the MLE objective function is formulated as below:

$$J_{MLE}(\theta) = E_{P_{data}(\mathcal{W})}[logf'_\theta(\mathcal{W})] \tag{8}$$

Note that here the normalization scalar c does not exist in the model.

In this work, noise contrastive estimation[18] is applied to train the bi-directional NNLM. NCE introduces a noise distribution $P_{noise}(\mathcal{W})$ into training and a "to-be-learned" normalization scalar c into the model, and its basic idea is that instead of maximizing the likelihood of the data samples, the model is asked to discriminative samples from the data distribution against samples from the noise distribution:

$$\begin{aligned} J_{NCE}(\theta) =& E_{P_{data}(\mathcal{W})}[logP(D=1|\mathcal{W};\theta)] \\ &+ kE_{P_{noise}(\mathcal{W})}[logP(D=0|\mathcal{W};\theta)] \\ P(D=1|\mathcal{W};\theta) =& \frac{P_\theta^{NCE}(\mathcal{W})}{P_\theta^{NCE}(\mathcal{W}) + kP_{noise}(\mathcal{W})} \\ P(D=0|\mathcal{W};\theta) =& \frac{kP_{noise}(\mathcal{W})}{P_\theta^{NCE}(\mathcal{W}) + kP_{noise}(\mathcal{W})} \end{aligned} \tag{9}$$

assuming a noise ratio of k.

And the gradients are:

$$\begin{aligned} \frac{\partial logP(D=1|\mathcal{W};\theta)}{\partial\theta} &= \frac{kP_{noise}(\mathcal{W})}{P_\theta^{NCE}(\mathcal{W}) + kP_{noise}(\mathcal{W})} \frac{\partial logP_\theta^{NCE}(\mathcal{W})}{\partial\theta} \\ \frac{\partial logP(D=0|\mathcal{W};\theta)}{\partial\theta} &= \frac{-P_\theta^{NCE}(\mathcal{W})}{P_\theta^{NCE}(\mathcal{W}) + kP_{noise}(\mathcal{W})} \frac{\partial logP_\theta^{NCE}(\mathcal{W})}{\partial\theta} \end{aligned} \tag{10}$$

For NCE to get good performance, a noise distribution that is close to the real data distribution is preferred, so in our case it is natural to use a good uni-directional LM as the noise distribution. In this work, N-gram LM is used as the noise distribution since it is efficient to sample from. Details about implementation and training process will be covered in section 4.

4. Training and Implementation details

Mini-batch based stochastic gradient descent(SGD) is used to train bi-directional NNLM in this work. The training process is very similar to [11], but several changes need to be made for the sentence-level bi-directional NNLM training. Since NN training in this work is sentence-level, data(consisted of real data samples and noise model samples) are processed in chunks(see figure 2). Moreover, a batch of data streams is processed together to utilize the computing power of GPU. It is relatively easy to realize this training process of BI-RNN with the help of neural network training tool-kits like CNTK[21]. In this work, the chunk size is set to 90(which is larger than the longest sentence in the ptb data-set) and the batch size is set to 64.

An validation-based learning strategy is used, the learning rate is fixed to a large value at first, and start halving at a rate of 0.6 when no significant improvement on the validation data is observed. And the training is stopped when that happens again. Further, a L2 regularization with coefficient 1e-5 is used.Finally, the SRILM[22] Toolkit is used for N-gram LM training in this work. In our training the N-GRAM noise is generated on-the-fly so noise samples won't be the same between iterations.

original	no it was n't black monday
s-error	no it was n't black **revoke**
d-error	no it was n't monday
i-error	no it **cracks** was n't black monday

Table 2: Examples of decoys in the **ptb-resocre** test set

5. Experiments

5.1. Datasets

In this section, results of experiments designed to test the performance of the proposed bi-directional NNLM trained by NCE. Since the training process is very time-costly when the noise ratio k is large(in our training framework, it will cost at least k times the time for training the baseline UNI-GRULM model), we confined our experiments to the Penn Treebank portion(PTB) of the WSJ corpus, which is publicly available and has been used extensively in LM community. There are 930k tokens, 74k tokens, 82k tokens for training, validation and testing, respectively, and the vocabulary size is 10k.

Since there is no guarantee that the trained model will be properly normalized, the evaluation of perplexity(PPL), which is the most conventional evaluation for LM, can no longer be applied. Instead, we need to resort to some discriminative task in which the LM is asked to tell "good" sentence from "bad" sentences, like its application in decoding or rescoring in systems like speech recognition or machine translation. But still, we want the training corpus and vocabulary size to be small enough, which will enable us to try a large noise ratio k, since sentence-level sampling is considered in this work, it is expected that k needs to be large enough for the training to work.

In light of the above concerns, a rescoring task is created directly on the PTB dataset, denoted as **ptb-rescore**[2]. In this test, random small errors are introduced to each sentence of the original test set, and the LM is then asked to recognize the original sentence from the tampered ones by assigning it the highest score. In this work, three types of error, namely **substitution**, **deletion** and **insertion**, are generated. For each error type, 9 decoys(one decoy only has one error) are generated for each test sentence, constituting three test sets. So a uniform guess will have an accuracy of 10%. Further, a mixed set where each decoy can be of any of the three types of error is also added, denoted as test **sdi**. Some examples are shown in table 2. Note that in this test set all random number(for the position or new word index) are drawn from a uniform distribution, and the **s-test** set is similar to the MSR sentence completion task [23].

5.2. Pseudo-PPL Test

Although perplexity can not be used to evaluate bi-directional NNLM, it is still interesting what PPL the trained model will

[2]This test set and the scripts for reproducing the N-gram baseline are available at `https://bitbucket.org/cloudygoose/ptb_rescore`

Model	noise ratio	Accuracy(%)/Accuracy after **length-norm**(%)			
		test-s	**test-d**	**test-i**	**test-sdi**
4-GRAM	-	75.4/n75.4	3.2/n12.7	**100**/n**98.2**	13.4/n40.8
UNI-GRULM	-	**80.6**/n**80.6**	**3.9**/n**21.8**	99.9/n96.9	**20.2**/n**60.9**
BI-GRULM(MLE)	-	50.0/n50.0	0.31/n21.9	95.3/n31.5	6.8/n27.1
BI-GRULM (NCE)	1	31.9/n31.9	3.9/n12.8	67.4/n53.0	10.9/n17.8
	10	39.9/n39.9	8.8/n19.4	61.8/n48.8	20.5/n26.2
	20	39.2/n39.2	**11.0**/n**21.6**	59.1/n45.3	**21.0**/n26.3
	50	48.4/n48.4	6.8/n19.8	74.2/n54.9	18.1/n29.0
	100	**55.7**/n**55.7**	0.5/n13.4	**98.6**/n**80.4**	10.3/n**34.5**

Table 1: Accuracy result of BI-GRULM models trained by NCE.

Model	Pseudo-PPL		
	test-ptb	**4gram-text**	**uniform-text**
UNI-GRULM	103.7	431.0	91935.7
BI-GRULM(MLE)	1.12	1.16	3.358
BI-GRULM(NCE)	15.5	3846.4	99565.4

Table 3: Pseudo-PPL result of different trained LMs on three test sets. The NCE experiment is with ratio 10.

assign to the test sentences. Besides the original test set for the **PTB** data, two additional text are generated, one is sentences sampled from the 4-GRAM baseline model(denoted as **4gram-text**), the other one is sentences sampled from a uniform distribution(denoted as **uniform-text**). All three sets have around 4,000 sentences. A well-behaved LM is expected to assign lowest PPL to the first set, relatively low PPL to the second, and very high PPL to the last one. The results are shown in table 3.

It is shown that the BI-GRULM(detailed configuration will be discussed in section 5.3) trained with NCE has similiar behavior to the baseline uni-directional model, meaning that NCE is helping the model with sentence-level normalization. On the contrary, BI-GRULM trained with MLE is assigning extremely low PPL to every test set, indicating that the model is not properly normalized. But surprisingly, the relative order of PPL from MLE-trained BI-GRULM is correct.

5.3. Evaluation on the ptb-rescore task

In this section accuracy results on the **ptb-rescore** task is presented. Three models are trained to be baseline models: 4-GRAM, UNI-GRULM, and BI-GRULM trained by MLE. Note that unless otherwise mentioned, all GRULMs trianed in the work has 300 neurons on hidden layer and only one layer(in the BI-GRULM case, one layer means one forward layer and one backward layer) is used. This setting is chosen for the reason that adding more neurons or more layers give no significant on the test PPL for the baseline UNI-GRULM model. Through training, a dropout rate of 50% is applied for the UNI-GRULM, but no dropout is applied for the reported experiments for the BI-GRULM because it is found that dropout won't give performance gain in that case.

The baseline results are shown in the upper part of table 1. Overally, the UNI-GRULM model gives the best performance, as expected. An interesting observation is that all model have extremely poor performance on the **test-d** set. This behavior, however, is not so surprising since the LM score of a sentence is afterall a product of word-level probabilities, so decoys with one less word will have big advantage. It is found that this problem can be alleviated by a **length-norm** trick:

$$score_{length-norm}(\mathcal{W}) = \frac{score(\mathcal{W})}{l} = \frac{\sum_i^l logf_i(\mathcal{W})}{l} \quad (11)$$

assuming sentence $\mathcal{W}$ is of length l(including the sentence-end token). Note that this trick is equivalent to ranking the sentences using PPL instead of sentence-level log likelihood and it will do harm to the performance on the **test-i** set, although not large.

Results of BI-GRULM trained by NCE are shown in in the lower part of table 1, it is observed that the **length-norm** trick can also help in this case, and the overall performance is improving with larger and larger noise ratio, however, it became unaffordable for us to run experiments with ratio larger than 100. One strange observation is that performance on the **test-d** set degrades with larger noise ratio, and this causes performance on the **test-sdi** to become worse. Also, comparing with the BI-GRULM(MLE) result, BI-GRULMs trained by NCE with a large noise ratio have overally better performance, indicating that NCE has the potential to utilize to power of BI-GRULM structure more properly.

Unfortunately, the proposed model failed to out-perform the best UNI-GRULM baseline model on every test set. Results on the **test-s** set show that improvement can only be obtained by growing the noise ratio exponentially, this matches our concern in section 5.1, the sentence-level sampling space may be too sparse for our sampling to properly cover.

6. Related work

In [19], bi-directional LSTMLM is trained with MLE and tested by LM rescoring in an ASR task. However, no improvement is observed over the uni-directional baseline model. On the other hand, NCE has been used in uni-directional LM training both for FNNLM[24] and RNNLM[25], the main goal was to speed-up the training and evaluation of these two models because under NCE training the final softmax operation on the output layer is no longer necessary. Note that different from these two work, NCE is applied on the sentence level in this work.

7. Conclusion

In this work the possibility of training bi-directional neural network language model with noise contrastive estimation is carefully investigated. Experiments are conducted on a rescore task and a sanity test on the PTB data set.

Unfortunately our proposed framework did not out-perform the baseline uni-directional NNLM, and the key reason maybe that the sentence-level sampling space is too sparse for our sampling to cover. However, it is shown that NCE-trained bi-directional NNLM outperformed the one trained by conventional maximum likelihood training, and it also behaves well in the sanity test. These results show that NCE is indeed helping the model to achieve sentence-level normalization.

8. Acknowledgements

The authors want to thank Abdelrahman Mohamed, Kaisheng Yao, Geoffrey Zweig, Dong Yu, Mike Seltzer, and Da Zheng for valuable discussions.

9. References

[1] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A neural probabilistic language model," *Journal OF Machine Learning Research*, vol. 3, pp. 1137–1155, 2003.

[2] H. Schwenk, "Continuous space language models," *Computer Speech Language*, vol. 21, no. 3, pp. 492–518, 2007.

[3] F. Morin and Y. Bengio, "Hierarchical probabilistic neural network language model," in *AISTATS*, 2005, pp. 246–252.

[4] J. Park, X. Liu, M. J. F. Gales, and P. C. Woodland, "Improved neural network based language modelling andadaptation," in *Proc. InterSpeech*, 2010.

[5] A. Mnih and G. Hinton, "Three new graphical models for statistical language modelling," in *Proc. ICML*, 2007, pp. 641–648.

[6] T. Mikolov, M. Karafiat, L. Burget, J. Cernocky, and S. Khudanpur, "Recurrent neural network based language model," in *Proc. InterSpeech*, 2010.

[7] M. Sundermeyer, I. Oparin, B. Freiberg, R. Schlter, and H. Ney, "Comparison of feedforward and recurrent neural network language models," in *Proc. ICASSP*, 2013.

[8] M. Sundermeyer, R. Schluter, and H. Ney, "Lstm neural networks for language modeling," in *Proc. InterSpeech*, 2012.

[9] Z. Huang, G. Zweig, and B. Dumoulin, "Cache based recurrent neural network language model inference for first pass speech recognition," in *Proc. ICASSP*, 2014.

[10] X. Liu, Y. Wang, X. Chen, M. J. F. Gales, and P. C. Woodland, "Efficient lattice rescoring using recurrent neural network language models," in *Proc. ICASSP*, 2014.

[11] X. Chen, Y. Wang, X. Liu, M. Gales, and P. C. Woodland, "Efficient gpu-based training of recurrent neural network language models using spliced sentence bunch," in *Proc. InterSpeech*, 2014.

[12] S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber, "Gradient flow in recurrent nets: the difficulty of learning long-term dependencies," 2001.

[13] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997. [Online]. Available: http://dx.doi.org/10.1162/neco.1997.9.8.1735

[14] W. Zaremba, I. Sutskever, and O. Vinyals, "Recurrent neural network regularization," *CoRR*, vol. abs/1409.2329, 2014. [Online]. Available: http://arxiv.org/abs/1409.2329

[15] J. Chung, Ç. Gülçehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *CoRR*, vol. abs/1412.3555, 2014. [Online]. Available: http://arxiv.org/abs/1412.3555

[16] S. Zhang, H. Jiang, S. Wei, and L. Dai, "Feedforward sequential memory neural networks without recurrent feedback," *CoRR*, vol. abs/1510.02693, 2015. [Online]. Available: http://arxiv.org/abs/1510.02693

[17] A. Graves and N. Jaitly, "Towards end-to-end speech recognition with recurrent neural networks," in *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, 2014, pp. 1764–1772. [Online]. Available: http://jmlr.org/proceedings/papers/v32/graves14.html

[18] M. U. Gutmann and A. Hyvärinen, "Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics," *J. Mach. Learn. Res.*, vol. 13, no. 1, pp. 307–361, Feb. 2012.

[19] E. Arisoy1, A. Sethy, B. Ramabhadran, and S. Chen, "Bidirectional recurrent neural network language models for automatic speech recognition," in *Proc. ICASSP*, 2015.

[20] M. Berglund, T. Raiko, M. Honkala, L. Kärkkäinen, A. Vetek, and J. Karhunen, "Bidirectional recurrent neural networks as generative models - reconstructing gaps in time series," *CoRR*, vol. abs/1504.01575, 2015.

[21] "An introduction to computational networks and the computational network toolkit," *Microsoft Research Technical Report*, pp. MSR–TR–2014–112, 2014.

[22] A. Stolcke, "Srilm-an extensible language modeling toolkit," in *Proceedings International Conference on Spoken Language Processing*, November 2002, pp. 257–286.

[23] "The microsoft research sentence completion challenge," *Microsoft Research Technical Report*, pp. MSR–TR–2011–129, 2011.

[24] A. Mnih and Y. W. Teh, "A fast and simple algorithm for training neural probabilistic language models," in *Proceedings of the 29th International Conference on Machine Learning*, 2012, pp. 1751–1758.

[25] M. J. F. G. X. Chen, X. Liu and P. C.Woodland, "Recurrent neural network language model training with noise contrastive estimation for speech recognition," in *Proc. ICASSP*, 2015.