RECURRENT NEURAL NETWORK LANGUAGE MODEL TRAINING WITH NOISE CONTRASTIVE ESTIMATION FOR SPEECH RECOGNITION

X. Chen, X. Liu, M. J. F. Gales and P. C. Woodland

University of Cambridge Engineering Dept,
Trumpington St., Cambridge, CB2 1PZ, U.K.
Email: {*xc257,xl207,mjfg,pcw*}*@eng.cam.ac.uk*

ABSTRACT

In recent years recurrent neural network language models (RNNLMs) have been successfully applied to a range of tasks including speech recognition. However, an important issue that limits the quantity of data used, and their possible application areas, is the computational cost in training. A significant part of this cost is associated with the softmax function at the output layer, as this requires a normalization term to be explicitly calculated. This impacts both the training and testing speed, especially when a large output vocabulary is used. To address this problem, noise contrastive estimation (NCE) is explored in RNNLM training. NCE does not require the above normalization during both training and testing. It is insensitive to the output layer size. On a large vocabulary conversational telephone speech recognition task, a doubling in training speed on a GPU and a 56 times speed up in test time evaluation on a CPU were obtained.

***Index Terms*—** language model, recurrent neural network, GPU, noise contrastive estimation, speech recognition

1. INTRODUCTION

Statistical language models (LMs) are crucial components in many speech and language processing systems designed for tasks such as speech recognition, spoken language understanding and machine translation. Recently, recurrent neural network language models (RNNLMs) have been shown consistent performance improvements across a range of these tasks [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]. One important practical issue associated with RNNLMs is the computational cost incurred in model training. This limits the quantity of data and their possible application areas, and therefore has drawn increasing research interest in recent years [2, 11, 12, 5, 13, 10, 14, 15].

A major part of the computation load is incurred at the output layer. One standard approach to handle this problem is to use class-based outputs. This limits the size of the output layer to be computed, thus allowing systems to be trained on CPUs. However, this approach is sensitive to the underlying word to class assignment scheme used at the output layer, and additionally complicates the implementation of bunch mode training parallelization [5]. To address these issues, RNNLMs with a full output layer were used and trained efficiently on GPUs using spliced sentence bunch in previous research [15]. A training speedup of 27 times was obtained over class based RNNLMs trained on CPUs.

One key factor that limits the scalability of RNNLMs is the computation of the normalization term in the output layer. This has a significant impact on both training and testing, especially when a large output vocabulary is used, in particular, in full output based RNNLMs. One technique that can be used to improve the testing speed is introducing the variance of the normalization term into the conventional cross entropy based objective function. This has been applied to training of feedforward NNLMs, class based [13, 10, 14] and full output RNNLMs [16]. By minimizing the variance of the normalization term during training, the normalization term at the output layer can be ignored during testing time thus gaining significant improvements in speed. However, the explicit computation of this normalization term is still required in training and it does not improve training speed.

In order to handle this problem, techniques that alleviate the need for explicit normalization at both training and testing time can be used to significantly improve the efficiency of RNNLMs. In this paper, noise contrastive estimation (NCE) [17] is explored for this purpose. NCE performs a nonlinear logistic regression to discriminate between the observed data and some artificially generated noise data. During NCE based training of NNLMs, only the connections associated with a few words in the output layer need to be considered, instead of computing the normalization over the full output vocabulary. Besides, NCE is able to constrain the variance of normalization term implicitly to be very small during training, which make it feasible to use “unnormalized” probabilities during testing. NCE was previously used to improve the training and evaluation efficiency of log-bilinear language models [18] and feedforward NNLMs [19]. A modified NCE algorithm using negative sampling was also adopted to deriving a distributed representation of words and phrases [20]. In this paper, NCE is used to improve the training and testing speed of RNNLMs for speech recognition.

The rest of this paper is organized as follows. RNNLMs are reviewed in section 2. Noise contrastive estimation is presented in section 3. Its detailed implementation is presented in section 4. Experiment results on a large vocabulary conversational telephone speech transcription task and Google’s One Billion Words corpus are reported in section 5. Section 6 draws conclusions.

2. RECURRENT NEURAL NETWORK LMS

In contrast to feedforward NNLMs, recurrent NNLMs [1] represent the full, non-truncated history $h_i =< w_{i-1}, \ldots, w_1 >$ for word w_i using the 1-of-k encoding of previous word w_{i-1} and a contin-

Xie Chen is supported by Toshiba Research Europe Ltd, Cambridge Research Lab. The research leading to these results was also supported by EPSRC grant EP/I031022/1 (Natural Speech Technology) and DARPA under the Broad Operational Language Translation (BOLT) and RATS programs. The paper does not necessarily reflect the position or the policy of US Government and no official endorsement should be inferred. The authors also would like to thanks Ashish Vaswani from USC for suggestions and discussion on training of NNLMs with NCE.

uous vector v_{i-2} for the remaining context. For an empty history, this is initialized, for example, to a vector of all ones. The topology of the recurrent neural network used to compute LM probabilities $P_{\text{RNN}}(w_i|w_{i-1}, v_{i-2})$ consists of three layers. The full history vector, obtained by concatenating w_{i-1} and v_{i-2}, is fed into the input layer. The hidden layer compresses the information from these two inputs and computes a new representation v_{i-1} using a sigmoid activation to achieve non-linearity. This is then passed to the output layer to produce normalized RNNLM probabilities using a softmax activation, as well as recursively fed back into the input layer as the "future" remaining history to compute the LM probability for the following word $P_{\text{RNN}}(w_{i+1}|w_i, v_{i-1})$. As RNNLMs use a vector representation of full histories, they are mostly used for N-best list rescoring. For more efficient lattice rescoring using RNNLMs, appropriate approximation schemes, for example, based on clustering among complete histories [21] can be used.

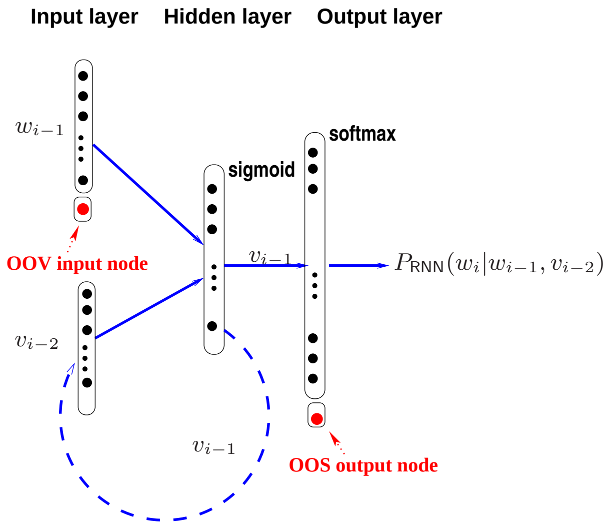

Fig. 1. *An example RNNLM with OOS nodes.*

An RNNLM architecture with an unclustered, full output layer is shown in Figure 1. RNNLMs can be trained using an extended form of the standard back propagation algorithm, back propagation through time (BPTT) [22], where the error is propagated through recurrent connections back for a specific number of time steps, for example, 4 or 5 [2]. This allows RNNLMs to keep information for several time steps in the hidden layer. To reduce the computational cost, a shortlist [23, 24] based output layer vocabulary limited to the most frequent words can be used. To reduce the bias to in-shortlist words during RNNLM training and improve robustness, an additional node is added at the output layer to model the probability mass of out-of-shortlist (OOS) words [25, 26, 21].

Conventional RNNLM training aims to maximise the log-likelihood, or equivalently minimize the cross entropy (CE) measure of the training data sequence containing a total of N_w words. The objective function is given by

$$J^{\text{CE}}(\boldsymbol{\theta}) \;=\; -\frac{1}{N_w}\sum_{i=1}^{N_w} \ln P_{\text{RNN}}(w_i|h_i) \qquad (1)$$

where

$$P_{\text{RNN}}(w_i|h_i) \;=\; \frac{\exp\left(\boldsymbol{\theta}_i^\top \boldsymbol{v}_{i-1}\right)}{\sum_{j=1}^{|V|}\exp\left(\boldsymbol{\theta}_j^\top \boldsymbol{v}_{i-1}\right)} \;=\; \frac{\exp\left(\boldsymbol{\theta}_i^\top \boldsymbol{v}_{i-1}\right)}{Z(h_i)} \qquad (2)$$

is the probability of word w_i given history h_i. $\boldsymbol{\theta}_i$ is the weight vector associated with word i at the output layer. $\boldsymbol{v}_{i-1}$ is the hidden history vector computed at the hidden layer, and $|V|$ is the size of output layer vocabulary. The gradient used in the conventional CE based training for RNNLMs is

$$\begin{aligned}\frac{\partial J^{\text{CE}}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \;=\; & -\frac{1}{N_w}\sum_{i=1}^{N_w}\Bigg(\frac{\partial\left(\boldsymbol{\theta}_i^\top \boldsymbol{v}_{i-1}\right)}{\partial \boldsymbol{\theta}} \\ & -\sum_{j=1}^{|V|} P_{\text{RNN}}(w_j|h_i)\frac{\partial\left(\boldsymbol{\theta}_j^\top \boldsymbol{v}_{i-1}\right)}{\partial \boldsymbol{\theta}}\Bigg). \qquad (3)\end{aligned}$$

The denominator term in equation (2) requires normalization over the full output layer. As discussed in section 1, this operation is computationally highly expensive when computing the RNNLM probabilities during both test time and CE based training when the gradient information of equation (3) is calculated.

In state-of-the-art ASR systems, NNLMs are often linearly interpolated with n-gram LMs to obtain both a good context coverage and strong generalisation [1, 5, 23, 24, 25, 26]. The interpolated LM probability is given by

$$P(w_i|h_i) \;=\; \lambda P_{\text{NG}}(w_i|h_i) + (1-\lambda)P_{\text{RNN}}(w_i|h_i) \qquad (4)$$

where λ is the weight of the n-gram LM $P_{\text{NG}}(\cdot)$, and kept fixed as 0.5 in all experiments of this paper. In the above interpolation, the probability mass of OOS words assigned by the RNNLM component is re-distributed with equal probabilities among all OOS words.

3. EFFICIENT RNNLM TRAINING AND DECODING USING NOISE CONTRASTIVE ESTIMATION

As discussed in section 1, the explicit computation of the the output layer normalization term significantly impacts both the training and testing speed of RNNLMs. A general solution to this problem is to use techniques that can remove the need to compute such normalization term in both training and testing. One such technique investigated in this paper is based on noise contrastive estimation (NCE) [17]. NCE provides an alternative solution to estimate normalized statistical models when the exact computation of the required normalization term is either computationally impossible or highly expensive to perform, for example, in feedforward and recurrent NNLMs, when a large output layer vocabulary is used. The central idea of NCE is to perform a nonlinear logistic regression to discriminate between the observed data and some artificially generated noise data. The variance of normalization term is minimized implicitly during training. Hence, it allows normalized statistical models, for example, NNLMs, to use "unnormalized" probabilities without explicitly computing the normalization term during both training and decoding. In common with the use of a class based output layer, the NCE algorithm presents a dual purpose solution to improve both the training and evaluation efficiency for RNNLMs.

For NCE based training of RNNLMs, it is assumed that for a given full history context h_i, data samples are generated from a mixture of two distributions: the NCE estimated RNNLM distribution $P_{\text{RNN}}^{\text{NCE}}(\cdot|h_i)$, and some noise distribution $P_n(\cdot|h_i)$ that satisfies a desired sum-to-one constraint. Assuming the noise samples are k times more frequent than true RNNLM data samples, the distribution of data could be described as $\frac{1}{k+1}P_{\text{RNN}}^{\text{NCE}}(\cdot|h_i) + \frac{k}{k+1}P_n(\cdot|h_i)$. The posterior probabilities of some word sample $\tilde{w}$ is generated from the RNNLM, or noise distribution are

$$\begin{aligned} P(C^{\mathsf{RNN}}_{\tilde{w}} = 1|\tilde{w}, h_i) &= \frac{P^{\mathsf{NCE}}_{\mathsf{RNN}}(\tilde{w}|h_i)}{P^{\mathsf{NCE}}_{\mathsf{RNN}}(\tilde{w}|h_i) + kP_n(\tilde{w}|h_i)} \\ P(C^{n}_{\tilde{w}} = 1|\tilde{w}, h_i) &= \frac{kP_n(\tilde{w}|h_i)}{P^{\mathsf{NCE}}_{\mathsf{RNN}}(\tilde{w}|h_i) + kP_n(\tilde{w}|h_i)} \end{aligned} \quad (5)$$

where $C^{\mathsf{RNN}}_{\tilde{w}}$ and $C^{n}_{\tilde{w}}$ are the binary labels indicating which of the two distributions that generated word $\tilde{w}$. The following objective function is minimized during NCE based training,

$$\begin{aligned} J^{\mathsf{NCE}}(\boldsymbol{\theta}) = & -\frac{1}{N_w}\sum_{i=1}^{N_w}\Big(\ln P(C^{\mathsf{RNN}}_{w_i} = 1|w_i, h_i) \\ & + \sum_{j=1}^{k} \ln P(C^{n}_{\breve{w}_{i,j}} = 1|\breve{w}_{i,j}, h_i)\Big) \end{aligned} \quad (6)$$

where a total of k noise samples $\{\breve{w}_{i,j}\}$ are drawn from the noise distribution $P_n(\cdot|h_i)$ for the current training word sample w_i and its history context h_i. The gradient of the above NCE objective function in equation (6) is then computed as

$$\begin{aligned} \frac{\partial J^{\mathsf{NCE}}(\boldsymbol{\theta})}{\partial\boldsymbol{\theta}} = & -\frac{1}{N_w}\sum_{i=1}^{N_w}\Bigg(\frac{kP_n(w_i|h_i)}{P^{\mathsf{NCE}}_{\mathsf{RNN}}(w_i|h_i) + kP_n(w_i|h_i)}\frac{\partial}{\partial\boldsymbol{\theta}}\ln P^{\mathsf{NCE}}_{\mathsf{RNN}}(w_i|h_i) \\ & -\sum_{j=1}^{k}\frac{P^{\mathsf{NCE}}_{\mathsf{RNN}}(\breve{w}_{i,j}|h_i)}{P^{\mathsf{NCE}}_{\mathsf{RNN}}(\breve{w}_{i,j}|h_i) + kP_n(\breve{w}_{i,j}|h_i)}\frac{\partial}{\partial\boldsymbol{\theta}}\ln P^{\mathsf{NCE}}_{\mathsf{RNN}}(\breve{w}_{i,j}|h_i)\Bigg) \end{aligned} \quad (7)$$

where the NCE trained RNNLM distribution is given by

$$P^{\mathsf{NCE}}_{\mathsf{RNN}}(w_i|h_i) = \frac{\exp\left(\boldsymbol{\theta}_i^{\top}\boldsymbol{v}_{i-1}\right)}{Z} \quad (8)$$

and constrained during NCE training to learn a constant, history context independent normalization term Z, in contrast to the standard CE training based RNNLM distribution given in equation (2)[1]. This crucial feature not only allows the resulting RNNLM to learn the desired sum-to-one constraint of standard CE estimated RNNLMs, but also to be efficiently computed during both training and test time without the computation of the output layer normalization term.

4. RNNLM TRAINING WITH NCE

In this paper, NCE training of RNNLMs is implemented on GPU using a spliced sentence bunch mode [15]. A bunch size of 128 was used in all experiments. CUBLAS is used for matrix operations. The NCE objective function shown in equation (7) is optimized on the training set. The cross entropy measure on the validation set is used to control the learning rate.

During NCE training, a number parameters need to be appropriately set. First, a noise distribution is required in NCE training to provide a valid sum-to-one constraint for the NCE estimated RNNLM to learn. As suggested in earlier research presented in [18, 19], a context independent unigram LM distribution is used to draw the noise samples during NCE training in this paper. Second, the setting of k controls the bias towards the characteristics of the noise distribution and balances the trade-off between training efficiency and performance. In this paper, for each data sample w, a total of $k = 10$ noise samples are sampled independently from the noise distribution. It is worth noting that the noise sample could be the predicted word and same noise sample may appear more than once. Finally, NCE training also requires a constant normalization term Z in equation (8) to be set. In previous research on NCE training of log-bilinear LMs [18] and feedforward NNLMs [19], the constant normalization term was set as $\ln Z = 0$. In this paper for RNNLMs an empirically adjusted setting of $\ln Z = 9$, close to the mean of the log scale normalization term computed using a randomly initialized RNNLM. This setting was found to give a good balance between convergence speed and performance and used in all experiments.

The main advantages of RNNLMs training with NCE is summarized below. First, the computation on output layer is reduced dramatically as it only needs to consider k noise samples and target word, instead of the whole output layer. Compared with the CE based training gradient given in equation (3), the computation of NCE gradient in equation (7) gives a total speed up of $\frac{|V|}{k+1}$ times at the output layer. Second, the train speed is insensitive to output layer size. This allows RNNLMs with a large vocabulary to be trained. Finally, the variance of normalization term is constrained to be a small value during NCE training. This can avoid the re-computation of normalization term for different histories, therefore allows the unnormalized RNNLM probabilities to used during decoding.

5. EXPERIMENTS

5.1. Experiments on conversational telephone speech

In this section, RNNLMs are evaluated on the CU-HTK LVCSR system for conversational telephone speech (CTS) used in the 2004 DARPA EARS evaluation. The acoustic models were trained on approximately 2000 hours of Fisher conversational speech released by the LDC. A 59k recognition word list was used in decoding. The system used a multi-pass recognition framework. A detailed description of the baseline system can be found in [27]. The 3 hour **dev04** data, which includes 72 Fisher conversations, was used as a test set. The baseline 4-gram LM was trained using a total of 545 million words from 2 text sources: the LDC Fisher acoustic transcriptions, **Fisher**, of 20 million words (weight 0.75), and the University Washington conversational web data [28], **UWWeb**, of 525 million words (weight 0.25). This baseline LM gave a perplexity (PPL) score of 51.8 and word error rate (WER) of 16.68% on **dev04**.

The 20M word **Fisher** data, was used to train RNNLMs in a sentence independent mode on a GPU. An Nvidia GeForce GTX TITAN GPU was used. A 32k vocabulary was used in the input layer and a 20k shortlist at the output layer. The size of hidden layer was 512. The number of BPTT steps was set as 5. The bunch size was set to 128. A more detailed description of the baseline GPU based bunch mode RNNLM training configuration is in [15]. The learning rate was 0.0117 per sample for NCE training and 0.0156 per sample for CE training. In order to ensure NCE training to be stable, setting the number of noise samples k above 10 was found necessary. In this paper, a 10 noise samples were generated from a unigram LM distribution for each predicted word. RNNLM weight parameters were randomly initialized between -0.1 and 0.1. The resulting RNNLMs were interpolated with the baseline 4-gram LM using equal weighting. 100-best rescoring was used to evaluate the performance of various RNNLMs on Intel Xeon E5-2670 2.6GHz CPUs with 16 physical cores.

As discussed in sections 1 and 3, an important attribute of NCE based RNNLM training is that the variance of the RNNLM output layer normalization term Z of equation (8) can be implicitly constrained to be minimum during parameter estimation. This effect is

[1] A more general case of NCE training also allows the normalization term to vary across different histories, thus incurring the same cost as in conventional CE based training [17].

illustrated in figure 2 on log scale over a total of 12 epochs on the validation data set. At the first epoch, the variance of the normalization term is slightly increased from 0.035 to 0.06 before gradually reduced again to 0.043 at the last epoch.

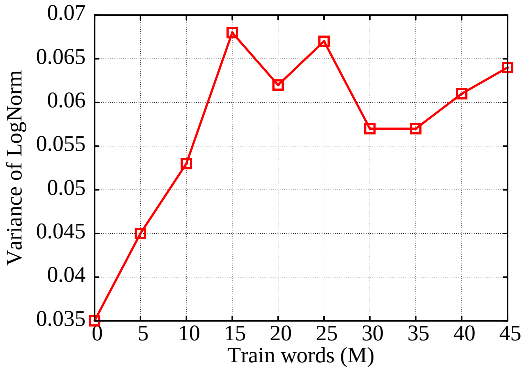

Fig. 2. *Variance of the output layer log-normalization term on validation data at different epochs during NCE based RNNLM training*

The WER and PPL performance of an NCE trained RNNLM are shown in table 1. 12 epochs were required for both the conventional CE and NCE based training to converge. As discussed in section 4, the log-normalization term $\ln Z$ in Equation (8) was fixed as 9. The perplexity scores in table 1 were obtained by explicitly computing the output layer normalization term. During N-best rescoring, normalized RNNLM probabilities were used for the CE trained RNNLM baseline, while unnormalized probabilities were used for the NCE trained RNNLM. As expected, when unnormalized probabilities were used by the CE trained RNNLM, a large degradation in performance was found. As is shown in table 1, the NCE trained RNNLM gave comparable performance to the CE trained baseline. At the same time, the training speech was doubled. This is expected as the time consumed on output layer is approximately half of the total training time required for conventional CE training.

LM Type	Train Crit	train speed(w/s)	train time (hours)	PPL	WER
NG4		-		51.8	16.68
+RNNLM	CE	10.1k	7.4	46.3	15.22
	NCE	19.7k	3.8	46.8	15.37

Table 1. *Performance and training speed of NCE trained RNNLMs*

Similarly a large testing time speed up of 56 times over the CE trained RNNLM on CPUs was also obtained, as is shown in table 5.1[2]. As the computation of the normalization term is no longer necessary for NCE trained RNNLMs, the computational cost incurred at the output layer can be significantly reduced.

Train Crit	Eval speed
CE	0.14k
NCE	7.9k

Table 2. *Testing speed of NCE trained RNNLMs on CPUs*

For the same reason above, the NCE training speed is largely invariant to the size of the output layer, thus improves the scalability of RNNLM training when a very large output vocabulary is used. This highly useful feature is clearly shown in table 3, where CE training speed is reduced rapidly when the output layer size increases. In contrast, the NCE training speed remains constant against different output layer vocabulary sizes.

#output layer	train speed (w/s)	
	CE	NCE
20k	10.1k	19.7k
25k	9.1k	19.7k
30k	8.0k	19.7k

Table 3. *Training speed against the size of RNNLM output layer*

5.2. Experiments on Google's one billion word benchmark

A new benchmark corpus was released by Google for measuring performance of statistical language models [29]. Two categories of text normalization are provided. One is for machine translation (StatMT) and the other is for ASR (by Cantab Research)[3]. The later was to further evaluation the performance of NCE trained RNNLMs in this paper. A total of 800 million words were used in LM training. A test set of 160k words (obtained from the first split from held-out data) was used for perplexity evaluation. A modified KN smoothed 5-gram LM was trained using the SRILM toolkit [30] with zero cut-offs and no pruning. In order to reduce the computational cost in training, an input layer vocabulary of 60k most frequent words and a 20k word output layer shortlist were used. RNNLMs with 1000 hidden layer nodes were either CE or NCE trained on a GPU using a bunch size of 128. The other training configurations were the same as the experiments presented in section 5.1. A total of 10 epochs were required to reach convergence for both CE and NCE based training. The perplexity performance of these two RNNLMs are shown in table 5.2. Consistent with the trend found in table 1, the CE and NCE trained RNNLMs gave comparable perplexity when interpolated with the 5-gram LM. A large perplexity reduction of 21% relative over the 5-gram LM was obtained.

LMs	Train Crit	PPL
NG5	-	83.7
+RNNLM	CE	65.8
	NCE	66.0

Table 4. *Perplexity of RNNLMs on Google's 1 billion word data*

6. CONCLUSION

Noise contrastive estimation (NCE) training was investigated for RNNLMs in this paper. Experimental results on a large vocabulary conversational telephone speech recognition system and the Google 1 billion word data suggest that the proposed technique can effectively alleviate the need for an explicit normalization term computation at the output layer in both training and testing time. A doubling in training speed and 56 times speed up in test time evaluation were both obtained. Future research will focus on improving the scalability of NCE training on larger data sets.

[2] As expected, this improvement is comparable to the speed up obtained using variance regularisation based RNNLM training [14, 16].

[3] All sources are available in https://code.google.com/p/1-billion-word-language-modeling-benchmark/. The machine translation normalized version of this data was previously used in [29] for RNNLM training.

7. REFERENCES

[1] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur, "Recurrent neural network based language model," in *Proc. ISCA Interspeech*, Makuhari, Japan, 2010, pp. 1045–1048.

[2] Tomas Mikolov, Stefan Kombrink, Lukas Burget, J.H. Cernocky, and Sanjeev Khudanpur, "Extensions of recurrent neural network language model," in *Proc. IEEE ICASSP*, Prague, Czech Republic, 2011, pp. 5528–5531.

[3] Anoop Deoras, Tomas Mikolov, Stefan Kombrink, Martin Karafiát, and Sanjeev Khudanpur, "Variational approximation of long-span language models for LVCSR," in *Proc. IEEE ICASSP*, Prague, Czech Republic, 2011, pp. 5532–5535.

[4] Gwénolé Lecorvé and Petr Motlicek, "Conversion of recurrent neural network language models to weighted finite state transducers for automatic speech recognition," in *Proc. ISCA Interspeech*, Portland, OR, 2012.

[5] Martin Sundermeyer, Ilya Oparin, Jean-Luc Gauvain, Ben Freiberg, Ralf Schluter, and Hermann Ney, "Comparison of feedforward and recurrent neural network language models," in *Proc. IEEE ICASSP*, Vancouver, Canada, May 2013, pp. 8430–8434.

[6] Grégoire Mesnil, Xiaodong He, Li Deng, and Yoshua Bengio, "Investigation of recurrent-neural-network architectures and learning methods for spoken language understanding," in *ISCA Interspeech*, Lyon, France, 2013, pp. 3771–3775.

[7] Kaisheng Yao, Geoffrey Zweig, Mei-Yuh Hwang, Yangyang Shi, and Dong Yu, "Recurrent neural networks for language understanding," in *ISCA Interspeech*, Lyon, France, 2013, pp. 2524–2528.

[8] Grégoire Mesnil, Xiaodong He, Li Deng, and Yoshua Bengio, "Investigation of recurrent-neural-network architectures and learning methods for spoken language understanding," in *ISCA Interspeech*, Lyon, France, 2013, pp. 3771-3775.

[9] Michael Auli, Michel Galley, Chris Quirk, and Geoffrey Zweig, "Joint language and translation modeling with recurrent neural networks," in *Proc. Proc. ACL EMNLP*, Seattle, Washington, 2013, pp. 1044–1054.

[10] Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard Schwartz, and John Makhoul, "Fast and robust neural network joint models for statistical machine translation," in *Proc. ACL*, Association for Computational Linguistics, Baltimore, Maryland, USA, June 2014, pp. 1370–1380.

[11] Yangyang Shi, Mei-Yuh Hwang, Kaisheng Yao, and Martha Larson, "Speed up of recurrent neural network language models with sentence independent subsampling stochastic gradient descent," in *Proc. ISCA Interspeech*, Lyon, France, 2013, pp. 1203-1206.

[12] Zhiheng Huang, Geoffrey Zweig, Michael Levit, Benoit Dumoulin, Barlas Oguz, and Shawn Chang, "Accelerating recurrent neural network training via two stage classes and parallelization," in *Proc. IEEE ASRU*, Olomouc, Czech Republic, December, 2013, pp. 326-331.

[13] Yongzhe Shi, Wei-Qiang Zhang, Meng Cai, and Jia Liu, "Efficient one-pass decoding with nnlm for speech recognition," *IEEE Signal Processing Letters*, vol. 21, no. 4, pp. 377–381, 2014.

[14] Yongzhe Shi, Wei-Qiang Zhang, Meng Cai, and Jia Liu, "Variance regularization of rnnlm for speech recognition," in *Proc. IEEE ICASSP*, Florence, Italy, 2014, pp. 4893–4897.

[15] Xie Chen, Yongqiang Wang, Xunying Liu, Mark Gales, and Phil Woodland, "Efficient training of recurrent neural network language models using spliced sentence bunch," in *Proc. ISCA Interspeech*, Singapore, 2014, pp. 641–645.

[16] Xie Chen, Xunying Liu, Mark Gales, and P. C. Woodland, "Improving the training and evaluation efficiency of recurrent neural network language models," to appear in *Proc. IEEE ICASSP*, Brisbane, Australia, 2015.

[17] Michael U Gutmann and Aapo Hyvärinen, "Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics," *Journal of Machine Learning Research*, vol. 13, no. 1, pp. 307–361, 2012.

[18] Andriy Mnih and Yee Whye Teh, "A fast and simple algorithm for training neural probabilistic language models," in *Proc. ICML*, Edinburgh, UK, 2012, pp. 1751–1758.

[19] Ashish Vaswani, Yinggong Zhao, Victoria Fossum, and David Chiang, "Decoding with large-scale neural language models improves translation," in *Proc. ACL EMNLP*, Seattle, Washington, 2013, pp. 1387–1392.

[20] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeff Dean, "Distributed representations of words and phrases and their compositionality," in *Proc. NIPS*, Lake Tahoe, NV, 2013, pp. 3111–3119.

[21] Xunying Liu, Yongqiang Wang, Xie Chen, Mark Gales, and P. C. Woodland, "Efficent lattice rescoring using recurrent neural network language models," in *Proc. IEEE ICASSP*, Florence, Italy, 2014, pp. 4941–4945.

[22] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams, *Learning representations by back-propagating errors*, MIT Press, Cambridge, MA, USA, 1988.

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

[24] Ahmad Emami and Lidia Mangu, "Empirical study of neural network language models for arabic speech recognition," *Proc. ASRU*, Kyoto, Japan, 2007, pp. 147–152.

[25] Junho Park, Xunying Liu, Mark J. F. Gales, and P. C. Woodland, "Improved neural network based language modelling and adaptation," in *Proc. ISCA Interspeech*, Makuhari, Japan, 2010, pp. 1041–1044.

[26] Hai-Son Le, Ilya Oparin, Alexandre Allauzen, J. Gauvain, and François Yvon, "Structured output layer neural network language models for speech recognition," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, no. 1, pp. 197–206, 2013.

[27] G. Evermann, H. Y. Chan, M. J. F. Gales, B. Jia, D. Mrva, P. C. Woodland, and K. Yu, "Training LVCSR systems on thousands of hours of data," in *Proc. IEEE ICASSP*, Philadelphia, PA, 2005, vol. 1, pp. 209–212.

[28] Ivan Bulyko, Mari Ostendorf, and Andreas Stolcke, "Getting more mileage from web text sources for conversational speech language modeling using class-dependent mixtures," in *Proc. HLT-NAACL*, Stroudsburg, PA, USA, vol. 2 pp. 7–9.

[29] Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson, "One billion word benchmark for measuring progress in statistical language modeling," Tech. Rep, Google, 2013.

[30] Andreas Stolcke, "SRILM – an extensible language modeling toolkit," in *ISCA ICSLP*, Denver, CO, 2002, pp. 901–904.