# 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 used in RNNLM training in this paper. It does not require the above normalization during both training and testing and is insensitive to the output layer size. On a large vocabulary conversational telephone speech recognition task, a doubling in training speed and 56 time speed up in test time evaluation 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 to produce 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 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, 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 against classs based RNNLMs trained on CPUs.

One key factor that limits the scalability of RNNLMs is the normalization term needs to be explicitly calculated at the output layer. This highly expensive operation has a significant impact on both the training and testing speed, 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 introduced an additional variance regularization term to the conventional 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.

In order to handle this problem, techniques that alleviate the need for such explicit normalization term computation in both training and testing time are preferred. One such technique investigated in this paper is based on noise contrastive estimation (NCE) [17]. By performing a nonlinear logistic regression to discriminate between the observed data and some artificially generated noise data, the algorithm presents solution to the both of abovementioned problems. It allows normalized statistical models, for example, NNLMs, to be both estimated in training and used in testing as "unnormalized" models without explicitly computing the normalization term, while retaining the desired sum-to-one constraint of normalized models. Along this line, 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 distributed representation of words and phrases [20]. In this paper, NCE is used to improve the training and testing speed of RNNLMs for automatic speech recognition.

The rest of this paper is organized as follows. In section 2, recurrent neural network LMs are reviewed. Noise contrastive estimation is presented in section 3. The detailed implement of NCE training 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 the conclusion.

## 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

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 authos also would like to thanks Ashish Vaswani from USC for suggestions and discussion on training of NNLMs with NCE.

$w_i$ using the 1-of-$k$ encoding of previous word $w_{i-1}$ and a continuous 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 $P_{\mathsf{RNN}}(w_i|w_{i-1}, v_{i-2})$ 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 of 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 $P_{\mathsf{RNN}}(w_{i+1}|w_i, v_{i-1})$. 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.

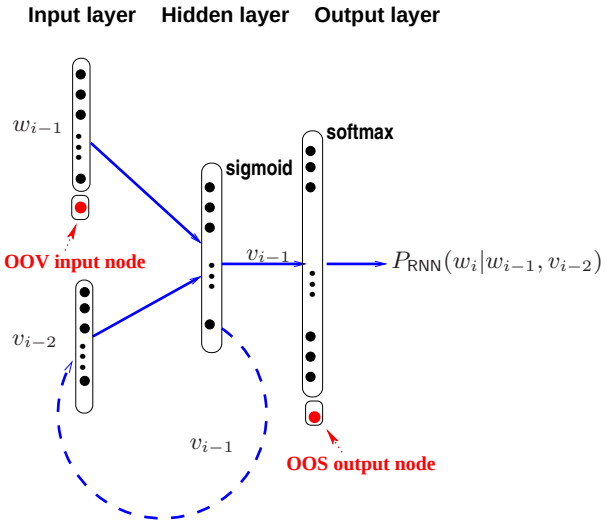


**Fig. 1**. *An example RNNLM with OOS nodes.*

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 record 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

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

where

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

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

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

The denominator term in equation (2) requires normalization among the whole 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

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

where $\lambda$ is the weight of the $n$-gram LM $P_{\mathsf{NG}}(\cdot)$, 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 needs to be re-distributed among all OOS words [25, 26].

## 3. NOISE CONTRASTIVE ESTIMATION

As discussed in section 1, the explicit computation of the the normalization term required at the output layer 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. Along this time, one such technique investigated in this paper is based on noise contrastive estimation (NCE) [17].

NCE based training 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. It allows normalized statistical models, for example, NNLMs, to be both estimated in training and used in testing as "unnormalized" models without explicitly computing the normalization term, while retaining the desired sum-to-one probabilistic constraint of normalized models. The NCE algorithm therefore presents a unique dual purpose solution to improve both the training and evaluation efficiency for RNNLMs.

In NCE training 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 $P^{\mathsf{NCE}}_{\mathsf{RNN}}(\cdot|h_i)$, and some noise distribution $P_n(\cdot|h_i)$ that satisfying 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 $\frac{1}{k+1}P^{\mathsf{NCE}}_{\mathsf{RNN}}(\cdot|h_i) + \frac{k}{k+1}P_n(\cdot|h_i)$. The posterior probabilities of some word sample $\tilde{w}$ is generated from the RNNLM, or

noise distribution are

$$P(C_{\tilde{w}}^{\mathsf{RNN}}=1|\tilde{w},h_i) = \frac{P_{\mathsf{RNN}}^{\mathsf{NCE}}(\tilde{w}|h_i)}{P_{\mathsf{RNN}}^{\mathsf{NCE}}(\tilde{w}|h_i)+kP_n(\tilde{w}|h_i)}$$
$$P(C_{\tilde{w}}^{n}=1|\tilde{w},h_i) = \frac{kP_n(\tilde{w}|h_i)}{P_{\mathsf{RNN}}^{\mathsf{NCE}}(\tilde{w}|h_i)+kP_n(\tilde{w}|h_i)} \quad (5)$$

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

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

where a total of $k$ noise samples $\{\check{w}_{i,j}\}$ 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

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

where the NCE trained RNNLM distribution is given by

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

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 requiring explicitly computing the normalization term at the output layer.

## 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 operation. 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 indenpendent 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 target word $w$, 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 grandient 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, which allows RNNLMs with larger vocabulary to be trained. Finally, the normalization term is constrained to be a constant during NCE training. This can avoid the re-computation of the normalization term for different histories, therefore allows the normalized RNNLM probabilities to calculate in test time with the same efficiency as unnormalized probabilities. A $|V|$ times speed up at the output layer during test time can thus be achieved.

## 5. EXPERIMENTS

### 5.1. Experiments on 20M CTS task

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 PPL of 51.8 and WER of 16.7% on **dev04** measured using lattice rescoring.

The **20M Fisher** data, containing on average 12.7 words per sentence, is used to train RNNLMs. A 32k shortlist is used in input layer and 20k shortlist used in output layer. RNNLMs are trained with sentence independent. The size of hidden layer is 512, BPTT step is 5 and bunch size is set to 128. The learning rate is 0.0117 per sample for NCE training and 0.0156 per sample for CE training. When $k$ is larger than 10, NCE training is stable. In this paper, 10 noise samples are generated from unigram distribution for each predicted word. The weights of RNNLMs are randomly initialized between -0.1 and 0.1. RNNLMs were interpolated with the baseline 4-gram LM using a fixed weight 0.5. 100 best rescoring is used to evaluate the performance of RNNLMs for ASR system. The CPU used in this paper is dual Intel Xeon E5-2670 2.6GHz processors with a total of 16 physical cores, and Nvidia GeForce GTX TITAN GPU is used. A more detailed description of system with RNNLMs could be found in [15].

The first experiment is to see how the variance of log normalization (LogNorm) term changes with different amount of training data

[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].

in NCE training. The variance of LogNorm on validation during training is shown in Figure 2. For RNNLMs with random initialization , the variance of LogNorm is small (0.035). It remains small and constraint during NCE training. The variance of LogNorm of NCE training ends with 0.043.

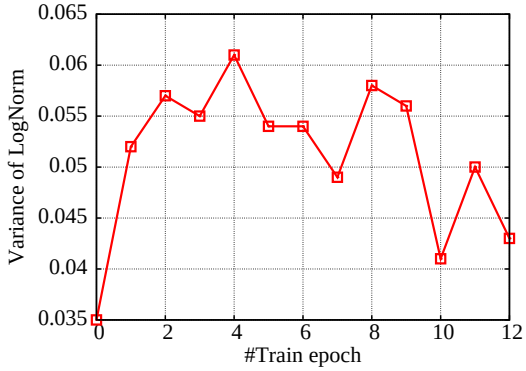


**Fig. 2**. *The change of variance of LogNorm on validation set in NCE training.*

Table 1 gives the WER and PPL results for NCE based RNNLMs. Both of NCE and CE based training take 12 epoches for convergence. The PPLs in the table are normalized. WER with normalized RNNLM probability is reported for CE training, while RNNLM trained with NCE uses unnormalized RNNLM probability. The log of normalization term $Z$ in Equation 8 is fixed with 9 in this work [2]. The PPL and WER of NCE trained model are slightly worse than CE trained model. However, it doubles train speed, and only half of training time is required. It is as expected since the time consumed on output layer is about half of the training time for CE training. While for NCE training, the time on output layer could be reduced significantly.

**Table 1**. Experiments on CTS task

| LM Type | Train Crit | train speed(w/s) | train time (hours) | PPL | WER |
|---|---|---|---|---|---|
| NG4 | - | | | 51.8 | 16.7 |
| +RNNLM | CE | 10.1k | 7.4 | 46.3 | 15.22 |
| | NCE | 19.7k | 3.8 | 46.8 | 15.37 |

The next experiment is to investigate the train speed with different output layer size. The experimental results are shows in Table 2. CE training slows down quickly with increase of output layer size, while that of NCE training is invariable with output layer size. The RNNLMs trained with CE and NCE using different output layer size give comparable PPL and WER performance.

**Table 2**. Train speed with different size of output layer

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

Another experiment is carried out to compare the evaluation speed for the RNNLMs trained with CE and NCE using CPUs. As mentioned before, the RNNLM trained with CE need to be normalized in output layer. While for NCE training, the normalization is avoided. According to the results shown in Table 3, the evaluation speed of RNNLMs trained with NCE is 56 times faster than that trained with CE on CPUs.

**Table 3**. Evaluation speed for CE and NCE trained RNNLM on CPU

| Train Crit | Eval speed |
|---|---|
| CE | 0.14k |
| NCE | 7.9k |

### 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 in [29]. Two categories of normalization are provided [3]. One is for machine translation (StatMT) and the other one is for ASR (normalized by Cantab Research). The later one is chosen for training of LMs in this paper. It contains 0.8 billion words in train set and 160k words (obtained from the first split from heldout dat) are used for evaluation. The vocabulary consists of 60k most frequent words and used in input layer. 20k word list is used in output layer. RNNLMs with 1000 hidden layer nodes are trained with bunch size 128. The other configuration of RNNLMs are the same as 20M fisher corpus in the above section. Modified KN 5-gram LM is trained by SRI toolkit [30] without prunning, containing 1.0 Billion ngrams. RNNLMs are trained with NCE and CE criterion respectively.

The results are shown in Table 4. The interpolation of 5-gram and RNNLM (with weight 0.5) gives significant PPL reduction up to 24% relatively. NCE and CE trained RNNLM gives comparable PPLs after interpolating with the 5-gram LM.

It is also worth mentioning there are some differences for the corpus and RNNLM between this work and [29]. [29] used statsMT, which is a corpus for machine translation, for LM training. Besides, MaxEnt was adopted in [29] for the training of RNNLMs.

**Table 4**. PPL results on Google's One Billion Word on 60k vocabulary

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

## 6. CONCLUSION

Noise contrastive estimation (NCE) training was investigated for RNNLMs in this paper. Experimental results on a large vocabulary conversational telephone speech recognition task suggest 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 time speed up in test time evaluation were obtained. The required training time is also insensitive to the output layer size. Experiments on Google One Billion Word corpus also shows that it scale well to much larger data.

[2]WER is much worse for CE trained RNNLMs without normalization

[3]All sources are available in https://code.google.com/p/1-billion-word-language-modeling-benchmark/

## 7. REFERENCES


[1] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur, "Recurrent neural network based language model.," in *Proc. ISCA Interspeech*, 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. ICASSP*. IEEE, 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. ICASSP*. IEEE, 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," Tech. Rep., Idiap, 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. 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 *INTERSPEECH*, 2013, pp. 3771–3775.

[7] Kaisheng Yao, Geoffrey Zweig, Mei-Yuh Hwang, Yangyang Shi, and Dong Yu, "Recurrent neural networks for language understanding.," in *INTERSPEECH*, 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.," .

[9] Michael Auli, Michel Galley, Chris Quirk, and Geoffrey Zweig, "Joint language and translation modeling with recurrent neural networks.," in *EMNLP*, 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 *Association for Computational Linguistics*, 2014.

[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*, 2013.

[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 *ASRU, IEEE Workshop on*. IEEE, 2013.

[13] Yongzhe Shi, Wei-Qiang Zhang, Meng Cai, and Jia Liu, "Efficient one-pass decoding with nnlm for speech recognition," *Signal Processing Letterss on*, 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. of ICASSP*, 2014.

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

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

[17] Michael U Gutmann and Aapo Hyvärinen, "Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics," *The 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," *International Conference on Machine Learning*, 2012.

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

[20] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems*, 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. ICASSP*. IEEE, 2014.

[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 & 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," in *ASRU, IEEE Workshop on*. IEEE, 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*, 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," *Audio, Speech, and Language Processing, IEEE Transactions on*, 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. of ICASSP*, 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 *In Proc. HLT*. ACL, 2003, 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 et al., "Srilm-an extensible language modeling toolkit.," in *INTERSPEECH*, 2002.