RECURRENT NEURAL NETWORK LANGUAGE MODELS FOR KEYWORD SEARCH

X. Chen[1], A. Ragni[1], J. Vasilakes [1], X. Liu[2], K. Knill[1], M.J.F. Gales[1]

[1] University of Cambridge Engineering Department, Cambridge, U.K.
[2] Department of Systems Engineering and Engineering Management, Chinese University of Hong Kong, Hong Kong

ABSTRACT

Recurrent neural network language models (RNNLMs) have becoming increasingly popular in many applications such as automatic speech recognition (ASR). Significant performance improvements in both perplexity and word error rate over standard n-gram LMs have been widely reported on ASR tasks. In contrast, published research on using RNNLMs for keyword search systems has been relatively limited. In this paper the application of RNNLMs for the IARPA Babel keyword search task is investigated. In order to supplement the limited acoustic transcription data, large amounts of web texts are also used in large vocabulary design and LM training. Various training criteria were then explored to improved RNNLMs' efficiency in both training and evaluation. Significant and consistent improvements on both keyword search and ASR tasks were obtained across all languages.

***Index Terms*—** speech recognition, keyword search, language model, recurrent neural network

1. INTRODUCTION

Language models are crucial components in many speech and language processing applications, such as speech recognition and machine translation. n-gram LMs have been the dominant language modelling approach for several decades. However, there are two well know issues associated with n-gram LMs, which are data sparsity and the n^{th} order Markov assumption [1]. RNNLMs provide a feasible solution for the two key n-gram issues. Furthermore, RNNLMs have been shown to produce significant improvements over n-gram LMs on a wide range of applications including speech recognition [2, 3], machine translation [4], spoken language understanding [5].

In previous research, RNNLMs have been widely used to improve the performance of speech recognition systems. In contrast, There are only limited works on applying neural network language models to the task of keyword search. In [6], feedforward neural network language models (FF-NNLMs) trained with a modified objective function to improve predition of rare words led to significantly improvements in keyword search performance. Unfortunately, this approach also led to a degradation in speech recognition performance. In order to obtain balanced improvements for both speech recognition and keyword search, a series of RNNLM training approaches featuring the efficient noise contrastive estimation (NCE) and variance regularisation (VR) criteria were investigated on state-of-the-art Cambridge University BABEL evaluation systems. These techniques allow large vocabulary RNNLMs to be efficiently trained and evaluated, as well as appropriately leverage large amounts of mixed in-domain acoutic data and out-of-domain general web texts. Significant performance improvements were obtained for both speech recognition and keyword search across five different languages. To our best knowledge, this is the first work dedicated to systematically designing and evaluating RNNLMs that are optimized for both speech recognition and keyword search tasks.

The rest of this paper is organised as follows. RNNLMs are briefly reviewed in Section 2. In Section 3, the task of keyword search for Babel project is described. Section 4 details the training of RNNLMs for keyword search in the Babel project. Experimental results are presented in Section 5 and the conclusion is drawn in Section 6.

2. RECURRENT NEURAL NETWORK LMS

The topology of the recurrent neural network [2] used to compute LM probabilities $P_{\mathsf{RNN}}(w_i|w_{i-1}, \mathbf{v_{i-2}})$ consists of three layers. The full history vector, obtained by concatenating w_{i-1} and $\mathbf{v_{i-2}}$, is fed into the input layer. The hidden layer compresses the information from these two inputs and computes a new representation $\mathbf{v_{i-1}}$ using a sigmoid activation to achieve non-linearity. This is then passed to the output layer to produce normalised 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 $P_{\mathsf{RNN}}(w_{i+1}|w_i, \mathbf{v_{i-1}})$.

An example RNNLM architecture with an unclustered, full output layer is shown in Figure 1. RNNLMs can be trained using back propagation through time (BPTT) [7]. To reduce the computational cost, a shortlist [8, 9] on output layer limited to the most frequent words can be used. An out-of-vocabulary (OOV) input node is used to represent any input word not in the chosen recognition vocabulary. 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 [10, 11, 12].

The full output layer based RNNLMs can be efficiently trained with the CUED-RNNLM toolkit [13, 14] on GPU with bunch mode. Besides the standard cross entropy criterion, two improved training criteria: variance regularisation (VR) and noise contrastive estimation (NCE), are supported in the CUED-RNNLM toolkit.

Xie Chen is supported by Toshiba Research Europe Ltd, Cambridge Research Lab. This work was supported by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Defense U. S. Army Research Laboratory (DoD/ARL) contract number W911NF-12-C-0012. The U. S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoD/ARL, or the U. S. Government..

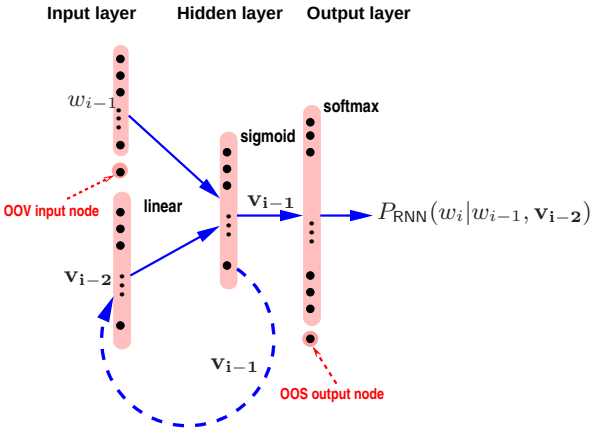

Fig. 1. *A full output layer RNNLM with OOS nodes.*

2.1. Cross Entropy (CE)

The conventional objective function used in RNNLM training is based on cross entropy (CE),

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

where N_w is the number of words in training corpus. Full output layer RNNLMs with moderate output layer size (e.g. 20K) can be trained on GPU efficiently [15]. However, if the output layer size increases to more than 100K, the computation of normalisation term for softmax is time-consuming, as shown in Eqn (2), where $\mathbf{a_i}$ is the weight vector associated with word w_i in the output layer. The computation of normalisation term $Z(h_i)$ is slow in both training and test time.

$$P_{\mathsf{RNN}}(w_i|h_i) = \frac{e^{\mathbf{v_{i-1}^T a_i}}}{\sum_j e^{\mathbf{v_{i-1}^T a_j}}} = \frac{e^{\mathbf{v_{i-1}^T a_i}}}{Z(h_i)} \qquad (2)$$

One solution to this issue is to learn a constant, history independent softmax normalisation term during RNNLM training. If the normalisation term $Z(h_i)$ could be approximated as constant C, unnormalised RNNLM probabilities are be used in test time as,

$$P_{\mathsf{RNN}}(w_i|h_i) \approx \frac{e^{\mathbf{v_{i-1}^T a_i}}}{C} \qquad (3)$$

2.2. Variance Regularisation (VR)

Variance regularisation explicitly adds the variance of normalisation term into the standard CE objective function [16]. The associated objective function is given by

$$J^{\mathsf{VR}}(\theta) \;=\; J^{\mathsf{CE}}(\theta) + \frac{\gamma}{2}\frac{1}{N_w}\sum_{i=1}^{N_w}(\ln(Z(h_i)) - (\overline{\ln Z}))^2 \qquad (4)$$

where $\overline{\ln Z}$ is the mean of log normalisation term. The second term added to the CE objective function models the variance of the log normalisation term. γ is a parameter to tune the effect of variance term over the CE criterion. In test time, RNNLM probabilities can be approximated as unnormalised probabilities in Eqn (3) and fast evaluation speed can be obtained.

2.3. Noise Contrastive Estimation (NCE)

In NCE training, each word in the training corpus is assumed to be generated by two different distributions [17]. One is data distribution, which is RNNLM, and the other is noise distribution, where unigram is normally used. The objective function is to discriminate these two distributions over the training data and a group of randomly generated noise samples. This is given by,

$$\begin{aligned} J^{\mathsf{NCE}}(\theta) \;=\; & -\frac{1}{N_w}\sum_{i=1}^{N_w}\big(\ln P(C_{w_i}^{\mathsf{RNN}}=1|w_i,h_i) \\ & +\sum_{j=1}^{k}\ln P(C^n_{\check{w}_{i,j}}=1|\check{w}_{i,j},h_i)\big) \end{aligned} \qquad (5)$$

where w_i is the ith target word, $\check{w}_{i,j}$ is the jth noise word generated for word w_i, and k is the number of noise samples. $P(C_{w_i}^{\mathsf{RNN}}=1|w_i,h_i)$ is the posterior probability of word w_i is generated by the RNNLM, and $P(C^n_{\check{w}_{i,j}}=1|\check{w}_{i,j},h_i)$ the posterior probability of word $\check{w}_{i,j}$ is generated by a noise distribution. During NCE training, the normalisation term $Z(h_i)$ is constrained to be constant implicitly. The training is only associated to the target word and k samples in the output layer, instead of the whole output layer. Hence, the output layer computational cost is no longer sensitive to vocabulary size and can be reduced significantly. In common with variance regularisation, unnormalised probabilities in Eqn. (3) can be used in test time. Hence, a large speedup on both training and test time can be achieved. More details about NCE training can be found from [18].

In many applications, RNNLMs are linearly interpolated with n-gram LMs to obtain both a good context coverage and strong generalisation [2, 19, 8]. The interpolated LM probability is given by

$$P(w_i|h_i) \;=\; \lambda P_{\mathsf{NG}}(w_i|h_i) + (1-\lambda)P_{\mathsf{RNN}}(w_i|h_i) \qquad (6)$$

where λ is the weight of the n-gram LM $P_{\mathsf{NG}}(\cdot)$, and is kept fixed at 0.5 in 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. KEYWORD SEARCH FOR BABEL PROJECT

The keyword search in the Babel program is a speech processing task based on speech recognition technology to find all occurrences of a word or word sequence, in a large audio corpus. Lattices are first generated from the ASR system and the query (i.e. keyword) is searched among all possible paths in lattices. The indexing and search of our KWS system are based on the weight finite state transducer (WFST) framework [20, 21].

During search, a query is represented as a weighted finite state acceptor (WFSA), and subsequently the composition operation is carried out to retrieve detection postings. More specifically, each in-vocabulary (IV) query term is converted to a word WFSA, and composed with the word index. If one IV term does not get any return, it is converted to a grapheme WFSA and searched again in the grapheme index. This is known as cascade search. On the other hand, the search for out-of-vocabulary (OOV) term are operated only on the grapheme level, i.e., all OOVs are represented as grapheme WFSAs, and composed with the grapheme index. Language model scores are ignored in OOV search. To further boost the OOV detection performance, a query expansion using grapheme-to-grapheme confusability (NBestP2P) [22] is applied. NBestP2P is set to 100 in all the experiments for this paper. Finally the IV and OOV search

posting lists are merged and STO score normalisation is applied to generate the final KWS output.

The performance of keyword search is evaluated using maximum term-weighted value (MTWV), to reflect the weighted cost of miss error and false alarm during keyword search. It is worth noting that the system is built for both speech recognition and keyword search, and the language model scale is tuned separately for speech recognition and keyword search.

4. RNNLMS FOR KEYWORD SEARCH

As stated in the above section, lattices are generated first by using a n-gram LM for first-pass decoding. RNNLMs are then used for lattice rescoring with n-gram approximation [23]. The resulting lattices from RNNLM rescoring are then used for keyword search.

Two sources of text are used to build language models. One is from the acoustic transcription, which is referred as FLP data. The other is obtained from the web using search engines, e.g. Wikipedia, Ted talk and Tweets. In [24], additional WEB data was used for language modelling and significant improvement was obtained on both the MTWV and WER metrics. The amount of the FLP data is normally much smaller than that of the WEB data. The statistics of the FLP and WEB data used in this paper can be found in Table 1. In order to reduce the OOV rate for both speech recognition and keyword search systems, a large vocabulary was also produced by using the additional WEB data.

The WEB data can be viewed as out-of-domain data and the FLP data as in-domain data. This can be reflected from the interpolation weight of the FLP data during the construction of n-gram LMs (6th column in Table 1). Given the in-domain and out-of-domain corpora, RNNLMs can be efficiently adapted with fine-tuning or incorporation of topic feature as discussed in [25]. Considering that the WEB data was collected by Columbia University and provided utterance by utterance, there is no explicit boundary for each document to train a topic model. Hence, the fine-tune method is adopted. RNNLMs are trained with two stages. They are first trained with all training data (denoted as WEB data) and then fine-tuned on the FLP data for adaptation purpose.

As discussed above, a larger output layer vocabulary significantly increases the CE training time for RNNLMs. In practice this may take up to one week to complete for some languages. Furthermore, these models are also computationally expensive in test time. One solution to improve the training and evaluation time efficiency is to use alternative training criteria such as variance regularisation (VR) [16] and noise contrastive estimation (NCE) [18]. Variance regularisation explicitly minimizes the variance of the softmax normalization term during RNNLM training, the normalization term at the output layer can be ignored during testing time thus gaining significant improvements in speed. The NCE algorithm further allows the output layer normalization term to be ignored also during training time, and provides a dual purpose solution to improve both the training and evaluation efficiency for RNNLMs. Three fine-tuning procedures were proposed to improve the RNNLMs efficiency.

The first approach (CE+VR) performs standard CE training on the WEB data, and then VR based training on the FLP data for fine-tuning. The model trained with VR was used for fast lattice rescoring without computing the normalisation term at the output layer.

In order to further improve the training efficiency on the WEB data, the second method (NCE+VR) performs NCE training on the WEB data first before VR based training on the FLP data. As discussed above, in addition to improve the evaluation time efficiency, NCE training also produced significantly faster training speed on larger WEB data set.

A third approach investigated in this paper performs NCE training on both the WEB and FLP data. As many vocabulary words originally selected from the WEB data do not appear in the FLP data, and the weight matrix parameters associated with these words can not be robustly trained, it is problematic to perform NCE based fine-tuning on the limited FLP data. This was found in practice lead to performance degradation. The same issue exists using noise samples drawn either from an interpolated unigram model trained on both the FLP and WEB data or a unigram model trained on FLP data only. In order to handle this issue and assigning sufficient weighting to the in-domain FLP data during RNNLM training, 9 copies of the FLP data were appended to the end of the WEB data to construct a "extended" corpus. NCE training is then performed on this "extended" corpus. All the three methods mentioned above were investigated on the Babel corpora and experimental results are presented in the following section.

5. EXPERIMENTS

In this paper, a total of 5 languages from the IARPA Babel program were chosen for experiments. Their corpora IDs in the Babel language releases are Pashto (104, IARPA-babel104b-v0.4bY), Igbo (306, IARPA-babel306b-v2.0c), Mongolian (401, IARPA-babel401b-v2.0b), Javanese (402, IARPA-babel402b-v1.0b) and Georgian (404, IARPA-babel404b-v1.0a) respectively.

5.1. Acoustic Models

The full language pack (FLP) in the Babel Program contains about 100-200 hours of transcribed audio training data ($\sim$60-80 hours speech) for acoustic model training. Tandem and Hybrid systems were built with speaker adaptation using CMLLR transferred features [26]. To obtain better performance, a 4-way joint decoding system [27], combining two Tandem and two Hybrid systems, was build using HTK toolkit [28]. The multi-lingual bottleneck features [29] provided by IBM and Aachen University were incorporated. A detailed description of the acoustic models can be found in [27].

5.2. Language Models

As mentioned before, two sources of data, the WEB data and FLP data, are used to construct language models. Description of these 5 languages is shown in Table 1. The vocabulary size varies from 28K (in Igbo) to 376K (in Pashto) and the amount of the WEB data varies from 2M (in Igbo) to 141M (in Mongolian). The size of FLP data is stable among these 5 languages, which lies in the range of 400K to 500K words. 3-gram LMs were first built on each source of text and then interpolated. The interpolation weights of the FLP data on each language are also given in the Table. For the training of RNNLMs, all train data is first used, and then fine-tuned on the FLP data. The vocabulary was chosen as the input layer and the three fourths most frequent words of the vocabulary modelled at the output layer. All RNNLMs were parallel trained with a spliced bunch mode on GPU [15] with a hidden layer of 100 hidden nodes. The bunch size is set to 128 at the first stage of training (on all data) and 64 at the second stage (on FLP data only). The empirically weighting parameter γ in equation (4) ranging from 0.5 to 2 was chosen for VR based training. For NCE training, 1000 noise samples were drawn from a unigram distribution and shared within each bunch. The 3-

gram approximation described in [23] is used for RNNLM rescoring of lattices initially generated by 3-gram back-off LMs.

Table 1. Statistics for 5 languages in Babel corpora.

Language	LM	#Data		FLP	OOV	
		Words	Vocab	Wgt	ASR	KWS
Pashto	FLP	535K	14.4K	-	1.96	11.38
	WEB	105M	376.3K	0.98	0.68	3.05
Igbo	FLP	549K	16.9K	-	2.59	11.97
	WEB	2M	28K	0.98	2.33	10.74
Mongolian	FLP	511K	24.0K	-	4.19	12.19
	WEB	139M	199.8K	0.93	2.10	5.62
Javanese	FLP	409K	16.5k	-	4.84	12.75
	WEB	73M	268.1K	0.98	2.13	4.18
Georgian	FLP	406K	34.3K	-	8.16	14.93
	WEB	137M	278.6K	0.91	3.02	5.22

The Pashto language is used to investigate the performance of RNNLMs first. Table 2 shows the perplexity (PPL) and WER results of the 3-gram LM and RNNLM. RNNLM gave an absolute 1.0% reduction in WER and 21% relative reduction in PPL. However, the RNNLM for Pashto contains 282k words in the output layer. It is time-consuming to train and evaluate RNNlms with CE due to the explicit normalisation at the RNNLM output layer. The time information can be found from the first row in the Table 3.

Table 2. PPL and WER results of RNNLM on Pashto.

LM	PPL	WER
3-gram	172	43.8
+RNN	136	42.8

In order to address the training and evaluation efficiency issues associated with CE training of large vocabulary RNNLMs, the three methods discussed in Section 4 were evaluated on the Pashto data for keyword search. The training and evaluation time of these three methods are shown in Table 3. It can be clearly seen that NCE significantly speeds up the training time on the WEB data by more than 10 times. VR also improves the evaluation speed on a similar scale.

Table 3. Train and evaluation time of various criteria on Pashto.

Train Crit		Times (Hrs)	
WEB	FLP	Train	Eval
CE	CE	125.0	23.0
CE	VR	130.0	1.6
NCE	VR	10.7	2.0

The WER and MTWV metric scores obtained using various RNNLM training criteria are shown in Table 4. The 1st block (i.e. 1st line) gives the baseline results using the 3-gram LM. The 2nd block shows results with RNNLMs. The 2nd to 4th lines present the performance of RNNLM trained with CE in the first stage, but different strategies for fine-tuning. The 2nd line is the result without fine-tuning on the FLP data, the 3rd line used CE criterion and the 4th line applied VR for fine-tuning. It can be seen that fine-tuning improved performance in both WER and MTWV performance. The first approach (CE+VR) discussed in Section 4 gave slight degradation in WER and MTWV compared to the CE fine-tuning in the 3rd line, but a large speedup in test time as shown in Table 3. The 5th and 6th lines adopted NCE in the first stage and compare performance with and without fine-tuning. Similar as CE training in the first stage, fine-tuning gives consistent improvement in WER and MTWV. The results of the second training method (NCE+VR) are shown in the 6th line in the Table. It gave comparable WER and MTWV scores as the RNNLM trained with CE (3rd line), while significant speedup in both training and evaluation time was obtained, as shown earlier in Table 3. The results of the third method (NCE on WEB + 9 copies of FLP data) are shown in the last line in Table 4. This system is also benefited from the large training and evaluation efficiency improvements of NCE, though further degradation in both WER and MTWV scores were also found.

Table 4. WER and MTWV results of RNNLM on Pashto.

LM	Train Crit		WER	MTWV		
	WEB	FLP		IV	OOV	Total
3-gram	-		43.8	0.4828	0.4083	0.4750
+RNN	CE	-	43.6	0.4852	0.4034	0.4842
		CE	42.8	0.4975	0.4048	0.4871
		VR	43.0	0.4958	0.4010	0.4853
	NCE	-	43.7	0.4916	0.4068	0.4824
		VR	43.0	0.4975	0.3953	0.4862
	NCE		43.2	0.4936	0.4038	0.4835

Considering the improvements on both system performance and efficiency, the second RNNLM training approach NCE+VR provided the best solution among all the three methods. It was then applied to the remaining 4 languages. The corresponding PPL, WER and MTWV results are presented in Table 5. It can be seen that consistent and significant improvements over 3-gram LMs in terms of WER and MTWV were obtained across all 4 languages.

Table 5. PPL, WER and KWS results of RNNLMs on 5 languages.

Language	LM	PPL	WER	MTWV		
				IV	OOV	Total
Pashto	3-gram	172	43.8	0.4828	0.4083	0.4750
	+RNN	136	43.0	0.4975	0.3953	0.4862
Igbo	3-gram	110	54.6	0.4079	0.3635	0.4030
	+RNN	94	53.7	0.4101	0.3718	0.4061
Mongolian	3-gram	134	47.0	0.5606	0.5171	0.5559
	+RNN	105	46.0	0.5708	0.5343	0.5666
Javanese	3-gram	219	50.1	0.5182	0.4801	0.5138
	+RNN	172	49.3	0.5229	0.4768	0.5173
Georgian	3-gram	472	37.8	0.7325	0.7300	0.7322
	+RNN	377	37.1	0.7386	0.7309	0.7375

6. CONCLUSION

In this paper, we present our recent work on recurrent neural network language models (RNNLMs) to improve the performance of Babel keyword search evaluation systems. In order to efficiently train and evaluate large vocabulary RNNLMs and appropriately use on large amounts of mixed in-domain and out-of-domain training data, a series of RNNLM training approaches featuring the efficient noise contrastive estimation and variance regularisation criteria were investigated on a combined corpus containing both general web texts and in-domain acoustic data. Significant improvements in both speech recognition and keyword search performance metrics were obtained across five different languages. To our best knowledge, this is the first work dedicated to systematically designing and evaluating RNNLMs that are optimized for both speech recognition and keyword search tasks.

7. REFERENCES

[1] Joshua Goodman, "A bit of progress in language modeling," *Computer Speech & Language*, vol. 15, no. 4, pp. 403–434, 2001.

[2] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur, "Recurrent neural network based language model.," in *Proc. ISCA INTERSPEECH*, 2010.

[3] Tomas Mikolov, Stefan Kombrink, Lukas Burget, J.H. Cernocky, and Sanjeev Khudanpur, "Extensions of recurrent neural network language model," in *Proc. ICASSP*. IEEE, 2011.

[4] Shujie Liu, Nan Yang, Mu Li, and Ming Zhou, "A recursive recurrent neural network for statistical machine translation.," in *ACL*, 2014, pp. 1491–1500.

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

[6] Ankur Gandhe, Florian Metze, Alex Waibel, and Ian Lane, "Optimization of neural network language models for keyword search," in *Proc. ICASSP*. IEEE, 2014, pp. 4888–4892.

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

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

[9] Ahmad Emami and Lidia Mangu, "Empirical study of neural network language models for Arabic speech recognition," in *ASRU, IEEE Workshop on*. IEEE, 2007.

[10] Junho Park, Xunying Liu, Mark Gales, and Phil Woodland, "Improved neural network based language modelling and adaptation," in *Proc. ISCA INTERSPEECH*, 2010.

[11] 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.

[12] Xunying Liu, Yongqiang Wang, Xie Chen, Mark Gales, and Phil Woodland, "Efficient lattice rescoring using recurrent neural network language models," in *Proc. ICASSP*. IEEE, 2014.

[13] Xie Chen, Xunying Liu, Mark Gales, and Phil Woodland, "CUED-RNNLM an open-source toolkit for efficient training and evaluation of recurrent neural network language models," in *Proc. ICASSP*. IEEE, 2015.

[14] Xie Chen, Xunying Liu, Yongqiang Wang, Mark Gales, and Phil Woodland, "Efficient training and evaluation of recurrent neural network language models for automatic speech recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2016.

[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 Phil Woodland, "Improving the training and evaluation efficiency of recurrent neural network language models," in *Proc. ICASSP*, 2015.

[17] Andriy Mnih and Yee Whye Teh, "A fast and simple algorithm for training neural probabilistic language models," *Proc. ICML*, 2012.

[18] Xie Chen, Xunying Liu, Mark Gales, and Phil Woodland, "Recurrent neural network language model training with noise contrastive estimation for speech recognition," in *Proc. ICASSP*, 2015.

[19] 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*, 2013.

[20] Brian Kingsbury, Jia Cui, Xiaodong Cui, Mark Gales, Kate Knill, Jonathan Mamou, Lidia Mangu, David Nolden, Michael Picheny, Bhuvana Ramabhadran, et al., "A high-performance Cantonese keyword search system," in *Proc. ICASSP*. IEEE, 2013, pp. 8277–8281.

[21] Cyril Allauzen, Mehryar Mohri, and Murat Saraclar, "General indexation of weighted automata: application to spoken utterance retrieval," in *Proc. HLT-NAACL*, 2004, pp. 33–40.

[22] Lidia Mangu, Hagen Soltau, Hong-Kwang Kuo, Brian Kingsbury, and George Saon, "Exploiting diversity for spoken term detection," in *Proc. ICASSP*. IEEE, 2013, pp. 8282–8286.

[23] Xunying Liu, Xie Chen, Yongqiang Wang, Mark Gales, and Phil Woodland, "Two efficient lattice rescoring methods using recurrent neural network language models," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 8, pp. 1438–1449, 2016.

[24] Gideon Mendels, Erica Cooper, Victor Soto, Julia Hirschberg, Mark Gales, Kate Knill, Anton Ragni, and Haipeng Wang, "Improving speech recognition and keyword search for low resource languages using web data," in *Proc. ISCA INTERSPEECH*, 2015.

[25] Xie Chen, Tian Tan, Xunying Liu, Pierre Lanchantin, Moquan Wan, Gales Mark, and Phil Woodland, "Recurrent neural network language model adaptation for multigenre broadcast speech recognition.," in *Proc. ISCA INTERSPEECH*, 2015.

[26] Mark Gales, "Maximum likelihood linear transformations for HMM-based speech recognition," *Computer Speech & Language*, vol. 12, no. 2, pp. 75–98, 1998.

[27] Haipeng Wang, Anton Ragni, Mark Gales, Kate Knill, Phil Woodland, and Chao Zhang, "Joint decoding of tandem and hybrid systems for improved keyword spotting on low resource languages," in *Proc. ISCA INTERSPEECH*, 2015.

[28] Steve Young, Gunnar Evermann, Mark Gales, Thomas Hain, Dan Kershaw, Xunying Liu, Gareth Moore, Julian Odell, Dave Ollason, Dan Povey, Anton Ragni, Valtcho Valtchev, Phil Woodland, and Chao Zhang, "The HTK book (for HTK version 3.5)," *Cambridge University Engineering Department*, 2015.

[29] Zoltan Tuske, David Nolden, Ralf Schluter, and Hermann Ney, "Multilingual MRASTA features for low-resource keyword search and speech recognition systems," in *Proc. ICASSP*. IEEE, 2014, pp. 7854–7858.