RECURRENT NEURAL NETWORK LANGUAGE MODELS FOR KEYWORD SEARCH

X. Chen¹, A. Ragni¹, J. Vasilakos¹, X. Liu², K. Knill¹, M.J.F. Gales¹

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 become 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 nth 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 prediction 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 acoustic 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 \( P_{\text{RNN}}(w_i|w_{i-1}, v_{1-2}) \) consists of three layers. The full history vector, obtained by concatenating \( w_{i-1} \) and \( v_{1-2} \), is fed into the input layer. The hidden layer compresses the information from these two inputs and computes a new representation \( v_{1-1} \) 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 \( P_{\text{RNN}}(w_{i+1}|w_i, v_{i-1}) \).

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.
2.1. Cross Entropy (CE)

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

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

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 $a_i$ 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.

$$ P_{\text{RNN}}(w_i|h_i) = \frac{e^{V_i \cdot a_i}}{e^{V_i \cdot a_i}} = \frac{e^{V_i \cdot a_i}}{Z(h_i)} $$

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,

$$ P_{\text{RNN}}(w_i|h_i) \approx \frac{e^{V_i \cdot a_i}}{C} $$

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

$$ J_{VR}(\theta) = J_{CE}(\theta) + \frac{1}{2N_w} \sum_{i=1}^{N_w} \left( \ln(Z(h_i)) - (\ln Z)^2 \right) $$

where $\ln Z$ is the mean of log normalisation term. The second term added to the CE objective function models the variance of the log normalisation term. $\gamma$ 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,

$$ J_{NCE}(\theta) = -\frac{1}{N_w} \sum_{i=1}^{N_w} \left( \ln P(C_{w_i}^{\text{RNN}} = 1|w_i, h_i) \right) $$

where $w_i$ is the $i$th target word, $\tilde{w}_{i,j}$ is the $j$th noise word generated for word $w_i$, and $k$ is the number of noise samples. $P(C_{w_i}^{\text{RNN}} = 1|w_i, h_i)$ is the posterior probability of word $w_i$ is generated by the RNNLM, and $P(C_{w_i}^{\text{NCE}} = 1|\tilde{w}_{i,j}, h_i)$ the posterior probability of word $\tilde{w}_{i,j}$ 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

$$ P(w_i|h_i) = \lambda P_{\text{NCE}}(w_i|h_i) + (1-\lambda) P_{\text{RNN}}(w_i|h_i) $$

where $\lambda$ is the weight of the $n$-gram LM $P_{\text{NCE}}(\cdot)$, 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 WFSA, 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 maxi-
mum term-weighted value (MTWV), to reflect the weighted cost of 
miss error and false alarm during keyword search. It is worth not-
ing 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 latt-
ces 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 incorpora-
tion 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 signifi-
cantly increases the CE training time for RNNLMs. In practice this 
may take up to one week to complete for some languages. Further-
more, 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 nor-
malization 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 dis-
cussed 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 train-
ing on both the WEB and FLP data. As many vocabulary words orig-
inally 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 “ex-
tended” 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 pro-
gram 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 (~60-80 hours 
speech) for acoustic model training. Tandem and Hybrid systems 
were built with speaker adaptation using CMU CLLR transferred fea-
tures [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 ofthe FLP and 
WEB data or a unigram model trained on FLP data only. In 
practice this may take up to one week to complete for some languages. Further-
more, 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 cannot 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 (~60-80 hours speech) for acoustic model training. Tandem and Hybrid systems were built with speaker adaptation using CMU CLLR 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 $\gamma$ 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-
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 5th and 6th lines adopted NCE 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.

<table>
<thead>
<tr>
<th>Language</th>
<th>Train Crit</th>
<th>WEB</th>
<th>FLP</th>
<th>WER</th>
<th>MTWV</th>
</tr>
</thead>
<tbody>
<tr>
<td>3-gram</td>
<td>CE</td>
<td>122</td>
<td>3.8</td>
<td>33.8</td>
<td></td>
</tr>
<tr>
<td>+RNN</td>
<td>CE</td>
<td>136</td>
<td>4.3</td>
<td>3.02</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CE</td>
<td>136</td>
<td>4.3</td>
<td>3.02</td>
<td></td>
</tr>
<tr>
<td></td>
<td>VR</td>
<td>130</td>
<td>1.6</td>
<td>1.6</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NCE</td>
<td>107</td>
<td>2.0</td>
<td>2.0</td>
<td></td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Language</th>
<th>PPL</th>
<th>WER</th>
<th>KWS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Igbo</td>
<td>43.6</td>
<td>40.34</td>
<td>4.086</td>
</tr>
<tr>
<td>Mongolian</td>
<td>43.0</td>
<td>40.10</td>
<td>4.853</td>
</tr>
<tr>
<td>Javanese</td>
<td>43.0</td>
<td>40.35</td>
<td>4.862</td>
</tr>
<tr>
<td>Georgian</td>
<td>43.0</td>
<td>40.35</td>
<td>4.835</td>
</tr>
</tbody>
</table>

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.

6. CONCLUSION
7. REFERENCES


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