EFFICIENT LATTICE RESCORING USING RECURRENT NEURAL NETWORK LANGUAGE MODELS

X. Liu, Y. Wang, X. Chen, M. J. F. Gales & P. C. Woodland

Cambridge University Engineering Dept,
Trumpington St., Cambridge, CB2 1PZ U.K.
Email: {*xl207,yw293,xc257,mjfg,pcw*}*@eng.cam.ac.uk*

ABSTRACT

Recurrent neural network language models (RNNLM) have become an increasingly popular choice for state-of-the-art speech recognition systems due to their inherently strong generalization performance. As these models use a vector representation of complete history contexts, RNNLMs are normally used to rescore N-best lists. Motivated by their intrinsic characteristics, two novel lattice rescoring methods for RNNLMs are investigated in this paper. The first uses an n-gram style clustering of history contexts. The second approach directly exploits the distance measure between hidden history vectors. Both methods produced 1-best performance comparable with a 10k-best rescoring baseline RNNLM system on a large vocabulary conversational telephone speech recognition task. Significant lattice size compression of over 70% and consistent improvements after confusion network (CN) decoding were also obtained over the N-best rescoring approach.

Index Terms: recurrent neural network, language model, speech recognition

1. INTRODUCTION

In order to handle the data sparsity problem associated with conventional back-off n-gram language models (LM), language modelling techniques that represent preceding history contexts in a continuous and lower dimensional vector space, such as neural network language models (NNLM) [2, 24, 22, 14, 27, 11], can be used. NNLMs are widely used in state-of-the-art speech recognition systems due to their inherently strong generalization performance. Depending on the network architecture being used, they can be categorised into two major categories: feedforward NNLMs [2, 24, 22, 11], which use a vector representation of preceding contexts of a finite number of words, and recurrent NNLMs (RNNLM) [14, 15, 27], which use a recurrent vector representation of longer and potentially variable length histories. In recent years RNNLMs have been shown to give significant improvements over conventional back-off n-gram LMs and feedforward NNLMs, thus gaining increasing research interest [14, 15, 4, 10, 27, 6, 28, 25].

One important practical issue associated with RNNLMs is the suitable decoding method to use. As RNNLMs use a complex vector space representation of full history contexts, it is generally not possible to apply these models in the early stage of speech recognition systems, or to directly rescore the word lattices produced by them. Instead, only a subset of the hypotheses encoded in a previously generated word lattice are used and converted into a linear, or prefix tree structured [25], N-best list. This practical constraint limits the possible improvements that can be obtained from RNNLMs for downstream applications that favor a more compact lattice representation, for example, when confusion network (CN) based decoding techniques [8] are used [28]. Several recent attempts have been made to address this issue [4, 5, 10, 6]. A sampling based approach was used to generate text data from an RNNLM to train a back-off n-gram LM as an approximation [4, 6]. A discrete quantization of RNNLMs into a weighted finite state transducer (WFST) [18] representation was proposed in [10]. Unfortunately neither of these two schemes were able to produce error rates that are comparable with the conventional N-best rescoring approach.

This paper aims to derive alternative decoding methods for RNNLMs that are more closely related to their modelling characteristics. First, the recursion through the full history produces a gradually diminishing effect of the information represented by the most distant contexts on the RNNLM probabilities. This allows complete histories that are partially overlapped or similar in the more recent contexts to share a similar distribution. It is thus possible to approximate RNNLMs based on truncated histories of sufficient length, similar to feedforward NNLMs. Second, in a more general case, RNNLMs internally cluster different histories encoded by the most recent word and hidden vector representing the remaining context via the similarity measure between them. Hence, it is also possible to explicitly use a history context vector distance measure to determine the sharing of RNNLM probabilities.

Motivated by the above hypotheses, two novel RNNLM lattice rescoring methods are investigated in this paper. The first uses an n-gram style approximation of history contexts. The second approach explicitly exploits the distance measure between hidden history vectors. The rest of the paper is organized as follows. Recurrent neural network LMs are reviewed in section 2. Two history contexts clustering schemes for RNNLMs are proposed in section 3. A generalized lattice rescoring algorithm for RNNLMs is presented in section 4. In section 5 the proposed RNNLM lattice rescoring techniques are evaluated on a state-of-the-art conversational telephone speech transcription task. Section 6 is the conclusion and future work.

2. RECURRENT NEURAL NETWORK LMS

In contrast to feedforward NNLMs, recurrent NNLMs [14] represent the full, non-truncated history $h_1^{i-1} = <w_{i-1}, \ldots, w_1>$ for word w_i using the 1-of-k encoding of the most recent preceding word

w_{i-1} and a continuous vector $\boldsymbol{s}_{i-2}$ 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 $P_{\mathsf{RNN}}(w_i|w_{i-1}, \boldsymbol{s}_{i-2})$ consists of three layers, as is shown in figure 1. The full history vector, obtained by concatenating the those of w_{i-1} and $\boldsymbol{s}_{i-2}$, is fed into the input layer. The hidden layer compresses the information of these two inputs and computes a new representation $\boldsymbol{s}_{i-1}$ 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 $P_{\mathsf{RNN}}(w_{i+1}|w_i, \boldsymbol{s}_{i-1})$.
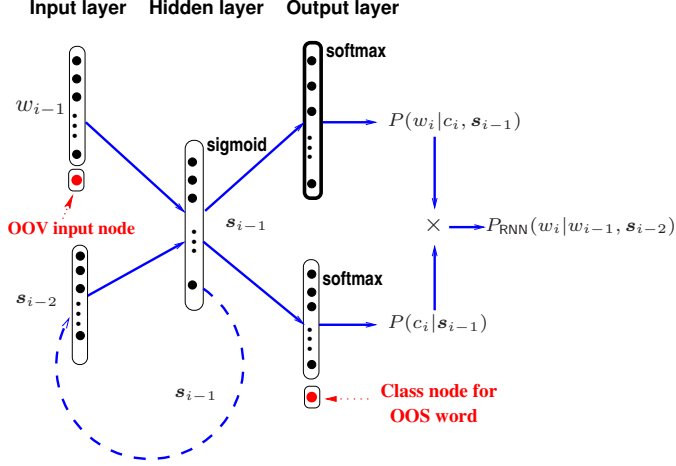

Fig. 1. *An RNNLM with an OOS output node.*

To reduce computational cost, a shortlist based output layer vocabulary limited to the most frequent words can be used. This was previously used for feedforward NNLMs [24, 7]. A similar approach may also be used at the input layer when a large vocabulary is used. In order to reduce the bias to in-shortlist words during NNLM training, two alternative network architectures that model a full vocabulary at the output layer can be considered. The first uses a class based factorized output layer structure [17]. Each word in the output layer vocabulary is attributed to a unique class based on frequency counts. The LM probability assigned to a word is factorized into two individual terms, for example, for the RNNLM shown in figure 1, as

$$P_{\mathsf{RNN}}(w_i|w_{i-1}, \boldsymbol{s}_{i-2}) \;=\; P(w_i|c_i, \boldsymbol{s}_{i-1})P(c_i|\boldsymbol{s}_{i-1}). \quad (1)$$

As the number of classes are normally significantly smaller than the output layer vocabulary size, training time speed-ups can be achieved for both feedforward NNLMs [17] and RNNLMs [15]. The second explicitly models the probability mass of out-of-shortlist (OOS) words using an additional output node [22, 11]. This ensures that all training data are used in training, and the probabilities of in-shortlist words are smoothed by the OOS probability mass to obtain a more robust parameter estimation. Drawing from both, this paper considers an RNNLM architecture that uses a factorized output layer for in-shortlist words and a separate output node to represent the probability mass of OOS words, as is shown in figure 1. This form of RNNLMs is used in the rest of this paper.

RNNLMs can be trained using an extended form of the standard back propagation algorithm, back propagation through time [23], where the error is propagated through recurrent connections back

in time for a specific number of time steps. This allows the recurrent network to record information for several time steps in the hidden layer. A modified version of the RNNLM toolkit [16] supporting the above modified architecture with an output layer OOS node is used.

In state-of-the-art speech recognition systems, NNLMs are often linearly interpolated with n-gram LMs to obtain both a good coverage of contexts and strong generalisation ability [24, 7, 22, 14, 27, 11]. The interpolated LM probability is given by

$$P(w_i|h_1^{i-1}) \;=\; \lambda P_{\mathsf{NG}}(w_i|h_1^{i-1}) + (1-\lambda)P_{\mathsf{RNN}}(w_i|h_1^{i-1}) \quad (2)$$

λ is the weight assigned to the back-off n-gram LM distribution $P_{\mathsf{NG}}(\cdot)$, 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 [22, 11].

3. HISTORY CONTEXT CLUSTERING FOR RNNLMS

Efficient use of language models in speech recognizers [21, 20, 18] requires that the context dependent states representing different histories during search can be appropriately shared among multiple paths. This applies to both conventional back-off n-gram and feedforward NNLMs. For these models, the underlying LM context state used to predict the current word is represented by a truncated, fixed length history of a maximum $N-1$ preceding words,

$$\boldsymbol{\Psi}_{\mathsf{NG}}(h_1^{i-1}) \;=\; h_{i-N+1}^{i-1} \;=\; <w_{i-1}, \ldots, w_{i-N+1}> . \quad (3)$$

The resulting n-gram LM distribution shared among multiple paths is thus $P_{\mathsf{NG}}(\cdot|\boldsymbol{\Psi}_{\mathsf{NG}}(h_1^{i-1})) \equiv P(\cdot|w_{i-1}, \ldots, w_{i-N+1})$.

In contrast, the context state of an RNNLM to predict a given word is represented by an ordered pair that encodes the full, complete history $h_1^{i-1} = <w_{i-1}, \ldots, w_1>$.

$$\boldsymbol{\Psi}_{\mathsf{RNN}}(h_1^{i-1}) \;=\; h_1^{i-1} \;\equiv\; <w_{i-1}, \boldsymbol{s}_{i-2}> \quad (4)$$

For this reason, the number of distinct RNNLM context states can grow exponentially as the search space is widened. Hence, it is generally non-trivial to apply RNNLMs in the early stage of speech recognition systems, or to directly rescore word lattices previously generated using these systems. Instead, previous research has been focused on using N-best list rescoring for RNNLM performance evaluation [14, 15, 27, 28, 25].

A general solution to the above problem is to derive appropriate *history clustering* methods for RNNLMs to allow a compact sharing of context states. Once a suitable form of equivalence between different complete histories is established, a discrete, finite state representation of RNNLMs becomes possible. Inspired by the entropy based pruning of back-off n-gram LMs [26], an optimal clustering method that merges two full histories, $h_1^{i-1} = <w_{i-1}, \ldots, w_1>$ and $\tilde{h}_1^{j-1} = <\tilde{w}_{j-1}, \ldots, \tilde{w}_1>$ together, is expected to minimize the KL divergence between the associated RNNLM distributions $P_{\mathsf{RNN}}(\cdot|h_1^{i-1})$ and $P_{\mathsf{RNN}}(\cdot|\tilde{h}_1^{j-1})$.

As discussed in section 1, both the decaying effect from the most distant history contexts and the similarity between hidden history vectors are exploited by RNNLMs during model estimation to achieve good generalization. These underlying modelling characteristics allow statistics to be distributed among different sequences that are "similar" or "related" by either their surface form or hidden vector representations. Both useful features can be also be exploited to derive suitable history clustering schemes for RNNLMs in decoding.

3.1. n-gram Based History Clustering

This intuitive clustering method is motivated by the fact that the recursion through the full preceding history can gradually diminishes the effect of the information represented by the most distant end of the contexts on the RNNLM probabilities. It is thus possible to cluster histories based on the common, most recent truncated contexts of $N - 1$ words maximum. The approximated RNNLM state for the complete history h_1^{i-1} is given by

$$\widetilde{\boldsymbol{\Psi}}_{\mathsf{RNN}}(h_1^{i-1}) = \begin{cases} \boldsymbol{\Psi}_{\mathsf{RNN}}(\tilde{h}_1^{j-1}) \text{ if } \exists \, \tilde{h}_1^{j-1} \text{ and} \\ \qquad\qquad h_1^{i-1} \cap \tilde{h}_1^{j-1} = \boldsymbol{\Psi}_{\mathsf{NG}}(h_1^{i-1}) \\ \boldsymbol{\Psi}_{\mathsf{RNN}}(h_1^{i-1}) \text{ otherwise} \end{cases} \quad (5)$$

where the shared n-gram style truncated history based LM state $\boldsymbol{\Psi}_{\mathsf{NG}}(h_1^{i-1})$ was previously defined in equation (3), and is equivalent to the intersection between h_1^{i-1} and $\tilde{h}_1^{j-1}$. As the truncation history length increases, the approximated RNNLM probabilities are expected to be increasingly closer to the true ones.

The above history clustering algorithm in practice operates as a LM state cache that stores the RNNLM probabilities associated with distinct truncated n-gram histories derived from $\boldsymbol{\Psi}_{\mathsf{NG}}(\cdot)$. By default, if a particular truncated history based state $\boldsymbol{\Psi}_{\mathsf{NG}}(h_1^{i-1})$ is not found in the cache, the full history h_1^{i-1} that subsumes the truncated context is used to create a new entry in the cache. As this algorithm uses the surface form information, it can be easily adapted and used by both beam search decoders [21, 20] where RNNLM probabilities can be computed on-the-fly by request and accessed via the cache, and WFST [18] style lattice rescoring where a previously generated network can be used to extract and explicitly build all possible shared RNNLM states into a WFST.

3.2. History Vector Based Clustering

For both feedforward and recurrent NNLMs, their strong generalization power is rooted from a continuous vector representation of history contexts in these models. When clustering histories, it is thus possible to directly exploit the similarity in their vector representation. The clustering method proposed here for RNNLMs aims to find the equivalence between two complete histories h_1^{i-1} and $\tilde{h}_1^{j-1}$ by comparing the identity of the most recent word w_{i-1} and $\tilde{w}_{j-1}$, and the distance measure $D(\boldsymbol{s}_{i-2}, \tilde{\boldsymbol{s}}_{j-2})$ between their respective hidden history vectors $\boldsymbol{s}_{i-2}$ and $\tilde{\boldsymbol{s}}_{j-2}$. A related beam pruning approach was previously used for variable length category based n-gram LMs [19]. The approximated RNNLM state for the complete history h_1^{i-1} is

$$\widetilde{\boldsymbol{\Psi}}_{\mathsf{RNN}}(h_1^{i-1}) = \begin{cases} \boldsymbol{\Psi}_{\mathsf{RNN}}(\tilde{h}_1^{j-1}) \text{ if } \exists \, \tilde{h}_1^{j-1}, w_{i-1} = \tilde{w}_{j-1} \\ \qquad\qquad \text{and } D(\boldsymbol{s}_{i-2}, \tilde{\boldsymbol{s}}_{j-2}) \leq \gamma \\ \boldsymbol{\Psi}_{\mathsf{RNN}}(h_1^{i-1}) \text{ otherwise} \end{cases} \quad (6)$$

where γ is a distance measure beam and can be tuned. When sharing the common most recent word, full histories that have a minimum vector difference below the beam are considered equivalent. The trade-off between modelling precision and the compactness of RNNLM state representation can be flexibly adjusted by the tuning of γ. In common with the n-gram history based scheme, this clustering method can also be implemented as a cache, and can be integrated into beam search based decoders [21, 20]. However, due to the introduction of the distance beam γ, it is not trivial to be used in generic WFST [18] based decoding approaches.

A range of distance measures may be considered for the distance measure $D(\boldsymbol{s}_{i-2}, \tilde{\boldsymbol{s}}_{j-2})$. As discussed above, the selection

of the appropriate metric to use in general can be determined based on the correlation between the underlying candidate metric and the KL divergence between the two RNNLM distributions to be merged. As the use of sigmoid activation at the hidden layer provides a well bounded dynamic range for the hidden history vector representation, the distance measure used is based on the Euclidean distance between $\boldsymbol{s}_{i-2}$ and $\tilde{\boldsymbol{s}}_{j-2}$. This is given by

$$D(\boldsymbol{s}_{i-2}, \tilde{\boldsymbol{s}}_{j-2}) = \frac{\sum_k \sqrt{(\boldsymbol{s}_{i-2,k} - \tilde{\boldsymbol{s}}_{j-2,k})^2}}{d} \quad (7)$$

where d is the dimensionality of the hidden history vectors.

4. LATTICE RESCORING USING RNNLMS

All the lattice rescoring experiments in this paper used an on-the-fly lattice expansion algorithm [13] suitable for a wide range of language models including back-off n-grams, feedforward NNLMs, recurrent NNLMs and their interpolated form [12]. A central part of the algorithm requires the LM state representation for the underlying model being used. For example, for back-off n-gram and feedforward NNLMs, this was given in equation (3). For approximated RNNLMs, this was based on equation (5) or (6) depending on the history clustering technique being used. The interpolated LM's state representation is derived from a union of those of component LMs. The corresponding pseudo-code algorithm for is given below.

1: **for** every node n_i in the network **do**
2:   initialize its expanded node list $N_i' = \{\}$;
3:   initialize its expanded outbound arc list $A_i' = \{\}$;
4: **end for**
5: add n_0 to its expanded node list, $N_0' = \{n_0\}$;
6: add all n_0's outbound arcs to its expanded arc list, $A_0' = A_0$;
7: Start depth first network traversal from the initial node n_0;
8: **for** every node n_i being visited **do**
9:   **for** every expanded node $n_j' \in N_i'$ of node n_i **do**
10:    **for** every outbound arc a_k from n_i **do**
11:     find the destination node n_k of arc a_k;
12:     find the LM state $\boldsymbol{\Psi}(h_{n_0}^{n_j'})$ of expanded node n_j';
13:     compute LM probability $P(n_k|\boldsymbol{\Psi}(h_{n_0}^{n_j'}))$;
14:     find a new LM state $\boldsymbol{\Psi}(h_{n_0}^{n_k})$ for node n_k;
15:     **if** $\exists$ node $n_l' \in N_k'$ representing state $\boldsymbol{\Psi}(h_{n_0}^{n_k})$ **then**
16:      return the found node n_l';
17:     **else**
18:      add a new node n_l' to N_k' to represent state $\boldsymbol{\Psi}(h_{n_0}^{n_k})$;
19:     **end if**
20:     create a new arc a_l' from n_j' to n_l';
21:     assign score $\ln P(n_k|\boldsymbol{\Psi}(h_{n_0}^{n_j'}))$ to a_l';
22:     add arc a_l' to the expanded outbound arc list A_i'.
23:    **end for**
24:   **end for**
25: **end for**
26: Rebuild new network using $\{N_i'\}$ and $\{A_i'\}$.

The above on-the-fly lattice expansion algorithm was implemented as an extension to the CU-HTK lattice processing tools.

5. EXPERIMENTS AND RESULTS

In this section performance of RNNLM lattice rescoring methods 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 uses a multi-pass recognition framework. A detailed description of the baseline system can be found in [9]. The 3 hour **dev04** data, which includes 72 Fisher conversations and contains on average 10.8 words per segment, was used as a test set.

The baseline 4-gram back-off LM "w4g" 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 [3], **UWWeb**, of 525 million words (weight 0.25). The 20M words of **Fisher** data, which contains on average 12.7 words per sentence, was used to train a feedforward 4-gram NNLM "nn_{w4g}" using the OOS architecture proposed in [22], and an RNNLM "rnn" using the modified architecture described in section 2 with 500 output layer classes. The same 38k word input layer vocabulary and 20k word output layer shortlist were used for both feedforward and recurrent NNLMs both with 500 hidden layer nodes. A total of 1 billion words of text data was generated from this RNNLM "rnn" using the sampling technique described in [4] to train a 4-gram back-off LM "rnn.sample.4g" as an approximation to the original RNNLM. These three LMs were then interpolated with the baseline 4-gram LM "w4g".

| | dev04 | | LatDensity |
LM	1-best	CN	(Arcs/Sec)
1. w4g	16.7	16.1	421
2. w4g+nn_{w4g}	16.3	15.8	555
3. w4g+rnn.50best	15.4	15.4	188(97)
4. w4g+rnn.100best	15.3	15.3	365(175)
5. w4g+rnn.1000best	15.3	15.1	3416(1298)
6. w4g+rnn.10000best	15.3	15.0	32277(10212)
7. w4g+rnn.sample.4g	16.2	15.9	462
8. w4g+rnn.approx3g	15.8	15.4	428
9. w4g+rnn.approx4g	15.7	15.2	555
10. w4g+rnn.approx5g	15.6	15.1	1266
11. w4g+rnn.approx6g	15.4	15.0	3025
12. w4g+rnn.approx7g	15.4	15.0	7140
13. w4g+rnn.hvd0.00450	15.8	15.4	465
14. w4g+rnn.hvd0.00300	15.6	15.2	539
15. w4g+rnn.hvd0.00200	15.6	15.1	699
16. w4g+rnn.hvd0.00100	15.6	15.1	1345
17. w4g+rnn.hvd0.00075	15.5	15.1	1842
18. w4g+rnn.hvd0.00050	15.4	15.0	2818
19. w4g+rnn.hvd0.00025	15.4	15.0	4725
20. w4g+rnn.hvd0.00001	15.4	15.0	6836

Table 1. 1-best, CN performance and HTK lattice density measured in arcs per second obtained using LMs on **dev04**. "w4g" is a 4-gram back-off LM and "w4g+nn_{w4g}" an interpolated LM combining "w4g" with a 4-gram feedforward NNLM. "w4g+rnn" interpolates "w4g" with an RNNLM "rnn". "w4g+rnn.∗best" used N-best rescoring. "w4g+rnn.sample.4g" combines "w4g" with a 4-gram back-off LM trained on texts sampled from "rnn". "w4g+rnn.approx∗g" and "w4g+rnn.hvd∗" used n-gram and hidden vector distance based RNNLM history clustering respectively.

The 1-best and CN word error rates (WER) of various baseline LMs are shown from the 1st to the 7th line in table 1. These include the back-off 4-gram LM "w4g", the feedforward NNLM system "w4g+nn_{w4g}", the RNNLM system "w4g+rnn.∗best" evaluated by re-ranking N-best lists of various depth from top 50 up to 10k entries, and the RNNLM sampled data trained 4-gram LM baseline "w4g+rnn.sample.4g". The RNNLM re-ranking N-best lists were then converted to prefix tree structured lattices [25] and used for CN decoding. The HTK formatted lattice density (Arcs/Sec) measure for all the above baseline systems are also shown in the last column of table 1. For the RNNLM N-best rescoring baseline systems, the lattice density measure before and after prefix tree structuring of N-bests lists are both given. As expected, prefix tree structuring of N-bests lists significantly reduced the size of the converted lattices (shown in brackets in the same column). As discussed in section 1, CN decoding favors a more efficient lattice representation that encodes rich alternative hypotheses. To achieve the same improvements from CN decoding, RNNLM rescored N-best list need to be as deep as 10k. This 10k-best RNNLM rescoring baseline gave the lowest 1-best and CN WER of 15.3% and 15.0% respectively, with a density of 10.2k arcs/sec measured on the lattices converted from the prefix tree structured N-bests lists.

The performance of using the n-gram approximation based RNNLM lattice rescoring methods presented in section 3 are shown in the 5th section of table 1 from line 8 to 12. When the truncated history is increased to 5 words, the resulting 6-gram approximated RNNLM system produced 1-best and CN error rates of 15.4% and 15.0%, both comparable with the standard RNNLM 10k-best rescoring baseline, and a significant 70% reduction in lattice size from 10k to 3k arcs/sec. Further increasing the truncated history length to 6 words via a 7-gram approximation gave no further improvement while only increased the size of the resulting lattices. This confirms the hypothesis raised in sections 1 and 3 over the decaying effect from remote history contexts on RNNLM probabilities.

The performance of using the hidden history vector distance based RNNLM lattice rescoring method proposed in section 3 are shown in the bottom section of table 1. By adjusting the hidden vector distance beam γ in equation (6), a range of approximated RNNLM comparable in error rates with the truncated history based approach but more compact lattices were produced. For example, setting $\gamma = 0.002$ produced equivalent 1-best and CN error rates of 15.6% and 15.1% as the 5-gram history approximated "w4g+rnn.approx5g" system, and a 45% reduction in lattice size from 1266 down to 699 arcs/sec. The best performance was obtained by setting $\gamma = 0.00050$ (3rd line from bottom in table 1), which gave 1-best and CN error rates of 15.4% and 15.0%, with a 72.4% and 7% reduction in lattice size over the 10k-best rescoring baseline, and the best n-gram history clustering rescoring system "w4g+rnn.approx6g" respectively. In practice, this "w4g+rnn.hvd0.00050" system can be used to rescore more heavily pruned lattices at 0.9 time real time (RT) while producing comparable 1-best and CN error rates of 15.4% and 15.1%. In contrast, the 1k-best and 10k-best rescoring systems used 1.8 and 17 times RT.

6. CONCLUSION AND RELATION TO PRIOR WORK

Two efficient lattice rescoring methods for RNNLMs were investigated in this paper. Both methods produced 1-best and CN decoding performance comparable with a 10k-best rescoring RNNLM baseline as well as over 70% compression in lattice size. In contrast, previously research on approximation of NNLMs in decoding [4, 10] were not able to either produce comparable error rate as the N-best rescoring, or produce lattices that are suitable for CN decoding [28]. Future research will focus on improving history clustering methods and efficiency in lattice rescoring using NNLMs.

7. REFERENCES

[1] E. Arisoy, S. F. Chen, B. Ramabhadran, and A. Sethy (2013), "Converting neural network language models into back-off language models for efficient decoding in automatic speech recognition," in *Proc. ICASSP*, Vancouver, Canada, 2013, pp. 8242–8246.

[2] Y. Bengio and R. Ducharme (2003), "A neural probabilistic language model," *Journal of Machine Learning Research*, vol. 3, pp. 1137–1155, 2003.

[3] I. Bulyko, M. Ostendorf, and A. Stolcke (2003), "Getting more mileage from web text sources for conversational speech language modeling using class-dependent mixtures," in *Proc. HLT*, Edmonton, Canada, 2003.

[4] A. Deoras, T. Mikolov, S. Kombrink, M. Karafiat, and S. Khudanpur (2011), "Variational approximation of long-span language models for LVCSR," in *Proc. ICASSP*, Prague, Czech Republic, 2011, pp. 5532–5535.

[5] A. Deoras, T. Mikolov and K. Church (2011), "A fast re-scoring strategy to capture long-distance dependencies", *Proc. EMNLP*, Edinburgh, UK, 2011, pp. 1116–1127.

[6] A. Deoras, T. Mikolov, S. Kombrink, and K. Church (2013), "Approximate inference: A sampling based modeling technique to capture complex dependencies in a language model," *Speech Communication*, vol. 55, no. 1, pp. 162–177, January 2013.

[7] A. Emami and L. Mangu (2007), "Empirical study of neural network language models for Arabic speech recognition," in *Proc. ASRU*, Kyoto, Japan, 2007, pp. 147–152.

[8] G. Evermann and P. C. Woodland (2000), "Posterior probability decoding, confidence estimation and system combination," in *Proc. Speech Transcription Workshop*, College Park, MD, 2000.

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

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

[11] H.-S. Le, I. Oparin, A. Allauzen, J. Gauvain, and F. Yvon (2013), "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.

[12] X. Liu, M. J. F. Gales, J. L. Hieronymus, and P. C. Woodland (2010), "Language model combination and adaptation using weighted finite state transducers," in *Proc. ICASSP*, Dallas, TX, 2010, pp. 5390–5393.

[13] X. Liu, M. J. F. Gales, and P. C. Woodland (2013), "Use of contexts in language model interpolation and adaptation," *Computer Speech & Language*, vol. 27, no. 1, pp. 301–321, January 2013.

[14] T. Mikolov, M. Karafiat, L. Burget, J. Cernocky, and S. Khudanpur (2010), "Recurrent neural network based language model," in *Proc. ISCA Interspeech*, Makuhari, Japan, 2010, pp. 1045–1048.

[15] T. Mikolov, S. Kombrink, L. Burget, J. H. Cernocky, and S. Khudanpur (2011), "Extensions of recurrent neural network language model," in *Proc. ICASSP*, Prague, Czech Republic, 2011, pp. 5528–5531.

[16] T. Mikolov, S. Kombrink, L. Burget, J. H. Cernocky and S. Khudanpur (2011), "RNNLM - Recurrent neural network language modeling toolkit", in demo session of *IEEE ASRU2011*, Hawaii.

[17] F. Morin and Y. Bengio (2005), "Hierarchical probabilistic neural network language model," in *Proc. International workshop on artificial intelligence and statistics*, Barbados, 2005, pp. 246–252.

[18] M. Mohri (1997), "Finite-state transducers in language and speech processing," *Computational linguistics*, vol. 23, no. 2, pp. 269–311, 1997.

[19] T. R. Niesler and P. C. Woodland (1996), "A variable-length category-based n-gram language model," in *Proc. ICASSP*, Atlanta, GA, 1996, vol. 1, pp. 164–167.

[20] H. Ney and S. Ortmanns (1999), "Dynamic programming search for continuous speech recognition," *IEEE Signal Processing Magazine*, vol. 16, no. 5, pp. 64–83, 1999.

[21] J. J. Odell, V. Valtchev, P. C. Woodland, and S. J. Young (1994), "A one pass decoder design for large vocabulary recognition," in *Proc. HLT*, Stroudsburg, PA, 1994, pp. 405–410.

[22] J. Park, X. Liu, M. J. F. Gales, and P. C. Woodland (2010), "Improved neural network based language modelling and adaptation," in *Proc. ISCA Interspeech*, Makuhari, Japan, 2010, pp. 1041–1044.

[23] D. E. Rumelhart, G. E. Hintont, and R. J. Williams (1986), "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

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

[25] Y. Si, Q. Zhang, T. Li, J. Pan, and Y. Yan (2013), "Prefix tree based n-best list re-scoring for recurrent neural network language model used in speech recognition system," in *Proc. ISCA Interspeech*, Lyon, France, 2013, pp. 3419–3423.

[26] A. Stolcke (1998), "Entropy-based pruning of backoff language models," in *Proc. DARPA Broadcast News Transcription and Understanding Workshop*, Landsdowne, VA, 1998, pp. 270–274.

[27] M. Sundermeyer, R. Schlüter, and H. Ney (2012), "LSTM neural networks for language modeling," in *Proc. ISCA Interspeech*, Portland, OR, 2012.

[28] M. Sundermeyer, I. Oparin, J. L. Gauvain, B. Freiberg, R. Schluter, and H. Ney (2013), "Comparison of feedforward and recurrent neural network language models," in *Proc. ICASSP*, Vancouver, Canada, 2013, pp. 8430–8434.