Two Efficient Lattice Rescoring Methods Using Recurrent Neural Network Language Models

Xunying Liu, *Member, IEEE*, Xie Chen, *Member, IEEE*, Yongqiang Wang, *Member, IEEE*, Mark J. F. Gales, *Fellow, IEEE*, and Philip C. Woodland, *Fellow, IEEE*

***Abstract*—An important part of the language modelling problem for automatic speech recognition (ASR) systems, and many other related applications, is to appropriately model long-distance context dependencies in natural languages. Hence, statistical language models (LMs) that can model longer span history contexts, for example, recurrent neural network language models (RNNLMs), have become increasingly popular for state-of-the-art ASR systems. As RNNLMs use a vector representation of complete history contexts, they are normally used to rescore N-best lists. Motivated by their intrinsic characteristics, two efficient lattice rescoring methods for RNNLMs are proposed in this paper. The first method uses an n-gram style clustering of history contexts. The second approach directly exploits the distance measure between recurrent hidden history vectors. Both methods produced 1-best performance comparable to a 10 k-best rescoring baseline RNNLM system on two large vocabulary conversational telephone speech recognition tasks for US English and Mandarin Chinese. Consistent lattice size compression and recognition performance improvements after confusion network (CN) decoding were also obtained over the prefix tree structured N-best rescoring approach.**

***Index Terms*—Language model, lattice rescoring, recurrent neural network, speech recognition.**

I. Introduction

A key part of the statistical language modelling problem for automatic speech recognition (ASR) systems, and many other related tasks, is to model the long-distance context dependencies in natural languages. Directly modelling long-span history contexts in their surface form can lead to a severe data sparsity problem. This presents a significant challenge for conventional back-off n-gram language models (LMs).

Manuscript received April 10, 2015; revised November 13, 2015, February 12, 2016, and April 3, 2016; accepted April 8, 2016. Date of publication April 27, 2016; date of current version May 27, 2016. This work was supported by EPSRC under Grant EP/I031022/1 (Natural Speech Technology) and DARPA under the Broad Operational Language Translation and RATS programs. The work of X. Chen was supported by Toshiba Research Europe Ltd, Cambridge Research Lab. The paper does not necessarily reflect the position or the policy of US Government and no official endorsement should be inferred.The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Bin Ma.

X. Liu is with the Department of Systems Engineering and Engineering Management, Chinese University of Hong Kong, Shatin, Hong Kong (e-mail: xyliu@se.cuhk.edu.hk).

Y. Wang was with the Department of Engineering, University of Cambridge, Cambridge, CB2 1PZ U.K. He is now with Microsoft Corporation, Redmond, WA 98052 USA (e-mail: yw293@cam.ac.uk).

X. Chen, P. C. Woodland, and M. J. F. Gales are with the Department of Engineering, University of Cambridge, Cambridge CB2 1PZ, U.K. (e-mail: xc257@cam.ac.uk; pcw@eng.cam.ac.uk; mjfg@eng.cam.ac.uk).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TASLP.2016.2558826

In order to address this issue, language modelling techniques that can represent longer span preceding history contexts in a continuous and lower dimensional vector space, such as neural network language models (NNLMs) [1]–[7], can be used. NNLMs are currently widely used in state-of-the-art speech recognition systems due to their inherently strong generalization performance. Depending on the underlying network architecture being used, they can be classified into two major categories: feedforward NNLMs [1]–[3], [7], which model a vector representation of the preceding context of a fixed number of words, and recurrent NNLMs (RNNLM) [4]–[6], 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 on a range of speech recognition tasks [4]–[6], [8]–[12], as well as other related applications including spoken language understanding [13], and machine translation [14]–[16], thus gaining increasing research interest.

When employing RNNLMs for speech recognition tasks, an important practical issue is the suitable decoding method to use. As RNNLMs use a vector space representation of full history contexts, it is non-trivial to apply these models in the early stage of ASR systems, or to directly rescore the word lattices produced by them. Instead, normally only a subset of the hypotheses encoded in a previously generated word lattice are used and converted into a linear [4], [5], or prefix tree structured [12], [17], 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 [18], [19] are used [11].

In order to address this issue, a range of techniques have been studied in recent years [8]–[10], [20]–[23]. Among these earlier works, a sampling based approach was used to generate text data from an RNNLM to train a back-off n-gram LM as an approximation [8], [10]. A discrete quantization of RNNLMs into a weighted finite state transducer (WFST) [24] representation was proposed in [9]. An iterative lattice rescoring approach was first proposed in [20] and further investigated in [21]. Unfortunately these earlier schemes were unable to produce 1-best error rates comparable to the conventional N-best rescoring approach [8], [9], or generate a compact lattice representation of the hypothesis space that is suitable for downstream applications such as CN decoding [20], [21]. Several later works that were more successful exploited the lattice internal hypothesis ranking produced by an earlier decoding pass. This allows an

approximate partial expansion of the underlying word graph to be performed during RNNLM rescoring [21], [22].

In contrast to the above existing methods, this paper aims to derive alternative lattice rescoring methods for RNNLMs that are independent of the acoustic and language model scores based hypothesis rank ordering produced in previous decoding stages derived using other LMs. The ultimate goals of the proposed RNNLM rescoring methods are: producing 1-best decoding performance comparable to the conventional N-best rescoring approach; and generating a compact lattice representation that is suitable for downstream applications including CN decoding.

The two techniques proposed in this paper are inspired by two intrinsic modelling characteristics of RNNLMs. 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, which is similar to feedforward NNLMs. Second, in a more general case, RNNLMs internally cluster different histories encoded by the most recent word and the hidden vector representing the remaining context via the similarity measure between them. Hence, it is also possible to explicitly use a hidden history vector distance based measure to determine the sharing of RNNLM probabilities. It is hoped that these characteristics can be exploited during decoding to improve computational efficiency.

Motivated by the above hypotheses, two efficient RNNLM lattice rescoring methods are investigated in this paper. The first uses an n-gram style clustering of history contexts [25], [26]. The second approach explicitly exploits the distance measure between recurrent hidden history vectors [25]. The rest of the paper is organized as follows. Recurrent neural network LMs are reviewed in Section II. Two history contexts clustering schemes for RNNLMs are proposed in Section III. A generalized lattice rescoring algorithm for RNNLMs is presented in Section IV. In Section V the proposed RNNLM lattice rescoring techniques are evaluated on two large vocabulary conversational telephone speech (CTS) transcription tasks for US English and Mandarin Chinese respectively. Section VI is the conclusion and discusses possible future work.

II. Recurrent Neural Network LMs

Unlike feedforward NNLMs, recurrent NNLMs [4] encode the full, non-truncated history $h_1^{i-1} = <w_{i-1}, \ldots, w_1>$ for the current word w_i being predicted using a 1-of-k encoding of the most recent preceding word w_{i-1} and a continuous vector $\boldsymbol{v}_{i-2}$ for the remaining history context. For an empty history, this is initialized, for example, to a vector of all ones. The topology of the recurrent neural network used to compute LM probabilities $P_{\text{RNN}}(w_i|h_1^{i-1}) = P_{\text{RNN}}(w_i|w_{i-1}, \boldsymbol{v}_{i-2})$ consists of three layers. An example RNNLM with an unclustered, full output layer is shown in Fig. 1. The full history vector, obtained by concatenating those of w_{i-1} and $\boldsymbol{v}_{i-2}$, is fed into the input layer. The hidden layer compresses the information from these two inputs

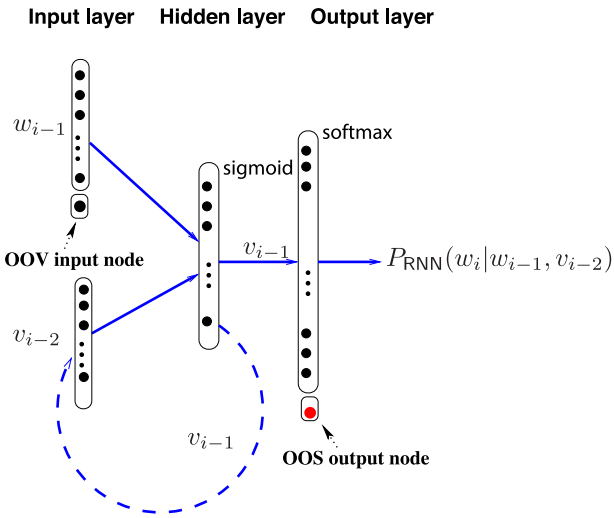


Fig. 1. An example RNNLM with an full output layer and OOS nodes.

and computes a new representation $\boldsymbol{v}_{i-1}$ 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 function, as well as recursively fed back into the input layer as the "future" remaining history to compute the LM probability for the following word $P_{\text{RNN}}(w_{i+1}|w_i, \boldsymbol{v}_{i-1})$.

Training and decoding using RNNLMs are both computationally expensive. A major part of the computation is required at the output layer. In order to reduce computational cost, a shortlist based output layer vocabulary limited to the most frequent words can be used. This approach was previously proposed for feedforward NNLMs [2], [27]. A similar approach may also be used at the input layer when a large vocabulary is used. An additional OOV input node can also be used to model words that are not in the input layer vocabulary, as is shown in Fig. 1.

A. Modelling Full Output Layer Vocabulary

Two issues arise when using a shortlist vocabulary at the output layer for RNNLMs. First, RNNLM parameters are trained only using the statistics of in-shortlist words thus introduces an undue bias to them. Secondly, as there is no explicit modelling of probabilities of *out-of-shortlist* (OOS) words in the output layer, statistics that are associated with these words are also discarded during RNNLM training. In order to address these issues, two alternative RNNLM network architectures that can model a full vocabulary at the output layer are preferred.

The first RNNLM architecture explicitly models the probability mass of OOS words using an additional output layer node [3], [7], as is shown in the example RNNLM in Fig. 1. This ensures that all training data are used in training. It also allows the probabilities of in-shortlist words are smoothed by the OOS probability mass during RNNLM training to obtain a more robust parameter estimation.

The second architecture uses a class based factorized output layer structure [28]. 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

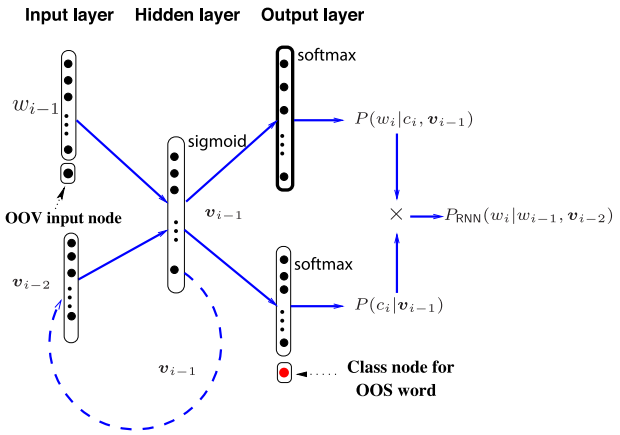


Fig. 2. An example RNNLM with a class-based output layer and OOS nodes.

terms as

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

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 [28] and RNNLMs [5].

It is also possible to draw from the strengths of both of these RNNLM architectures. Along this line, RNNLMs with a factorized class based output layer for in-shortlist words and a separate output node to represent the probability mass of OOS word can be used. An example of such an RNNLM is shown in Fig. 2. This form of class based RNNLM and the full output based RNNLM shown in Fig. 1 are considered in the rest of this paper.

B. Efficient Training of RNNLMs

RNNLMs can be trained using an extended form of the standard back propagation algorithm, back propagation through time (BPTT) [29], [30]. During BPTT based training, the error is propagated through recurrent connections back in time for a specific number of time steps, for example, 4 or 5 [5]. This allows the recurrent network to record information for several time steps in the hidden layer.

The above BPTT method based RNNLM training is computationally expensive. This practical issue limits the quantity of data and the number of possible application areas for RNNLMs. In order to solve this problem, recently there has been increasing research interest in deriving efficient parallel training algorithms for RNNLMs [31]–[34]. In particular, RNNLMs with a full output layer were efficiently trained on a graphics processor unit (GPU) using a spliced sentence bunch based parallel training algorithm in [34]. A training speedup of 27 times was obtained over class based RNNLMs trained on a CPU. In [35] this technique is further extended and applied to class based RNNLMs. A modified version of the RNNLM toolkit [36] supporting the above GPU based parallel RNNLM training method and the RNNLM architectures shown in Figs. 1 and 2 is used in this paper.

C. Combination Between n-gram LMs and RNNLMs

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 [2]–[4], [6], [7], [27]. For RNNLMs, the interpolated LM probability of the current word w_i given the full history context $h_1^{i-1} = <w_{i-1}, \ldots, w_1>$ is given by

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

where λ is the linear interpolation weight assigned to the back-off n-gram LM distribution $P_{\text{NG}}(\cdot)$, and kept fixed at 0.5 in all experiments of this paper.

In the above interpolation scheme, the probability mass of OOS words assigned by the RNNLM component needs to be re-distributed among all OOS words [3], [7]. This can be achieved using the n-gram LM statistics $P_{\text{NG}}(\cdot)$ as,

$$\tilde{P}_{\text{RNN}}(w_i|h_1^{i-1}) = \begin{cases} P_{\text{RNN}}(w_i|h_1^{i-1}) & w_i \in V_{\text{sl}} \\ \beta(w_i|h_1^{i-1})P_{\text{RNN}}(w_{\text{oos}}|h_1^{i-1}) & \text{otherwise} \end{cases}$$

$$\beta(w_i|h_1^{i-1}) = \frac{P_{\text{NG}}(w_i|h_1^{i-1})}{\sum_{\tilde{w}_i \notin V_{\text{sl}}} P_{\text{NG}}(\tilde{w}_i|h_1^{i-1})} \quad (3)$$

where V_{sl} is output shortlist vocabulary, and w_{oos} the OOS word. The above form of OOS probability normalization is used throughout this paper for RNNLM perplexity evaluation.

The above normalisation can be very expensive for LVCSR tasks. In order to improve decoding efficiency, assuming that the OOS probability mass assigned by the RNNLM and n-gram LM are equal, an approximate form of normalisation can be used [3]. The following form of OOS probability normalization is used throughout this paper during RNNLM lattice rescoring:

$$\tilde{P}_{\text{RNN}}(w_i|h_1^{i-1}) \approx \begin{cases} P_{\text{RNN}}(w_i|h_1^{i-1}) & w_i \in V_{\text{sl}} \\ P_{\text{NG}}(w_i|h_1^{i-1}) & \text{otherwise.} \end{cases} \quad (4)$$

III. History Context Clustering for RNNLMs

In current speech recognition systems, an efficient use of language model information requires that the context dependent states representing different histories during search can be appropriately shared among multiple hypotheses [24], [37], [38]. This principle applies to both conventional back-off n-gram LMs and feedforward NNLMs. For these language 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,

$$\boldsymbol{\Psi}_{\text{NG}}(h_1^{i-1}) = h_{i-N+1}^{i-1} = < w_{i-1}, \ldots, w_{i-N+1} > . \quad (5)$$

The resulting n-gram LM distribution shared among multiple decoding paths is thus computed as

$$P_{\text{NG}}(\cdot|\boldsymbol{\Psi}_{\text{NG}}(h_1^{i-1})) \equiv P(\cdot|w_{i-1}, \ldots, w_{i-N+1}). \quad (6)$$

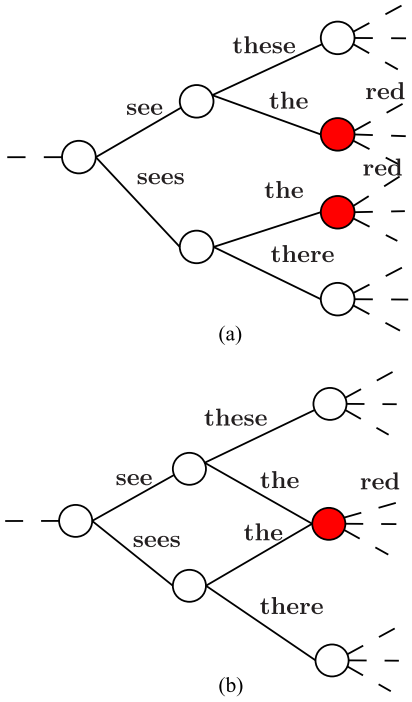


Fig. 3. Example parts of a prefix tree structured n-best list (a) and lattice (b).

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 $h_1^{i-1} = <w_{i-1}, \ldots, w_1>$

$$\mathbf{\Psi}_{\mathrm{RNN}}(h_1^{i-1}) = h_1^{i-1} = < w_{i-1}, \boldsymbol{v}_{i-2} > . \tag{7}$$

For this reason, the number of distinct RNNLM context states can grow exponentially as the sentence length increases. 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, a large part of the previous research has been focused on using a N-best list rescoring based framework for RNNLM performance evaluation [4]–[6], [11], [12]. For efficiency, prefix tree structured n-best lists [12] can be used to represent partial histories that are identical among different hypotheses. Example parts of a prefix tree structured n-best list and a word lattice are shown in Fig. 3(a) and (b). As is shown in the figure, prefix tree structured n-best lists require distinct nodes associated with word "the" to be created once the preceding histories along the two associated paths differ from each other. In contrast, a more compact lattice structure allows these two paths to be merged when a 2-gram language model is used.

In this paper, a general solution adopted to solve the above problem is to derive appropriate *history clustering* methods for RNNLMs to allow a compact sharing of context states [25]. Once a suitable form of equivalence between different complete histories is established, a discrete, finite state representation of RNNLMs becomes possible. 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 Kullback-Leibler (KL) divergence between the associated RNNLM distributions $P_{\mathrm{RNN}}(\cdot|h_1^{i-1})$ and $P_{\mathrm{RNN}}(\cdot|\tilde{h}_1^{j-1})$.

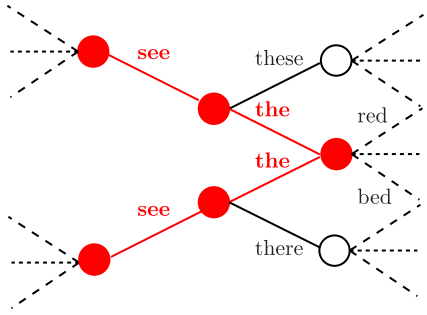


Fig. 4. An example of 3-gram based RNNLM history clustering.

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

A. *n-gram Based History Clustering*

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

$$\widetilde{\mathbf{\Psi}}_{\mathrm{RNN}}(h_1^{i-1}) = \begin{cases} \mathbf{\Psi}_{\mathrm{RNN}}(\tilde{h}_1^{j-1}) \text{ if } \quad \exists\ \tilde{h}_1^{j-1} \text{in cache and} \\ h_1^{i-1} \cap \tilde{h}_1^{j-1} = \mathbf{\Psi}_{\mathrm{NG}}(h_1^{i-1}) \\ \mathbf{\Psi}_{\mathrm{RNN}}(h_1^{i-1}) \ \text{ otherwise} \end{cases} \tag{8}$$

where the shared n-gram style truncated history based LM state $\mathbf{\Psi}_{\mathrm{NG}}(h_1^{i-1})$ was previously defined in equation (5). It is equivalent to the intersection (common most recent truncated n-gram histories) between two full histories $h_1^{i-1} = <w_{i-1}, \ldots, w_1>$ and $\tilde{h}_1^{j-1} = <\tilde{w}_{j-1}, \ldots, \tilde{w}_1>$. For example, when a 3-gram history clustering is used, two complete histories sharing the common most recent two words "see" and "the" are considered equivalent. This is illustrated in Fig. 4.

As the truncated history length increases, the approximate RNNLM probabilities are expected to be increasingly closer to the true ones. In this paper, the above history clustering algorithm is implemented as a hash table based cache during lattice rescoring. This cache stores the RNNLM probabilities associated with a set of distinct context histories, as well as the associated recurrent hidden vectors encoding these histories. When accessing the cache for a given full history $h_1^{i-1} = <w_{i-1}, \ldots, w_1>$ using the n-gram history clustering

in equation (8), the aim to find in the cache a history $\tilde{h}_1^{j-1} = <\tilde{w}_{j-1}, \ldots, \tilde{w}_1>$ that shares the common truncated n-gram context given in equation (5) with the given history h_1^{i-1}. If such history $\tilde{h}_1^{j-1}$ is found in the cache, its associated RNNLM probabilities and recurrent hidden vectors are used as an approximation to those associated with h_1^{i-1}. Otherwise the given history h_1^{i-1} is used to compute the necessary RNNLM probabilities and recurrent vectors and create new entries in the cache.

As this clustering algorithm directly uses the surface form information, it can be used by beam search decoders [37], [38], where RNNLM probabilities can be computed on-the-fly by request and accessed via the cache. In a similar way, a cache based RNNLM first pass decoding technique [26] was also successfully applied to a weighted finite state transducer [24] based speech recognizer.

B. History Vector Based Clustering

For both feedforward and recurrent NNLMs, their strong generalization power is rooted in a continuous vector representation of history contexts. When clustering histories, it is thus possible to directly exploit the similarity in their vector representation measured by the underlying RNNLM being used. The clustering method proposed here for RNNLMs aims to find the equivalence between two complete histories $h_1^{i-1} = <w_{i-1}, \ldots, w_1>$ and $\tilde{h}_1^{j-1} = <\tilde{w}_{j-1}, \ldots, \tilde{w}_1>$ by comparing the identity of the most recent word w_{i-1} and $\tilde{w}_{j-1}$, and the distance measure $D(\boldsymbol{v}_{i-2}, \tilde{\boldsymbol{v}}_{j-2})$ between their respective recurrent hidden history vectors $\boldsymbol{v}_{i-2}$ and $\tilde{\boldsymbol{v}}_{j-2}$. A related beam pruning approach was previously used for variable length category based n-gram LMs [39]. The approximate RNNLM state for the complete history h_1^{i-1} is computed as

$$\widetilde{\boldsymbol{\Psi}}_{\mathrm{RNN}}(h_1^{i-1}) = \begin{cases} \boldsymbol{\Psi}_{\mathrm{RNN}}(\tilde{h}_1^{j-1}) \text{ if } \quad \exists\, \tilde{h}_1^{j-1}, w_{i-1} = \tilde{w}_{j-1} \\ \text{and } D(\boldsymbol{v}_{i-2}, \tilde{\boldsymbol{v}}_{j-2}) \leq \gamma \\ \boldsymbol{\Psi}_{\mathrm{RNN}}(h_1^{i-1}) \ \text{ otherwise} \end{cases} \tag{9}$$

where γ is a hidden history vector distance beam. It can be tuned to flexibly adjust the trade-off between modelling precision and the compactness of the underlying RNNLM state representation.

When sharing the common most recent word, full histories that have a minimum hidden vector difference below the distance beam are considered equivalent. For example, when two complete histories are sharing the most recent word "the," and their respective recurrent vectors representing the remaining history contexts "we see" and "he sees" are also sufficiently close, they are considered equivalent. This is illustrated in Fig. 5.

As with the n-gram history based history clustering scheme of Section III-A, this hidden vector distance based clustering method is also implemented as a cache during lattice rescoring in this paper. A cache access scheme similar to that of the n-gram history clustering in Section III-A is used. Using the form of history vector clustering in equation (9), a cache hit is determined by the availability of a history $\tilde{h}_1^{j-1}$ that shares the most recent word with the given history h_1^{i-1}, and the distance measure $D(\boldsymbol{v}_{i-2}, \tilde{\boldsymbol{v}}_{j-2})$ between their respective recurrent hidden vectors $\boldsymbol{v}_{i-2}$ and $\tilde{\boldsymbol{v}}_{j-2}$ is below the distance beam γ. This

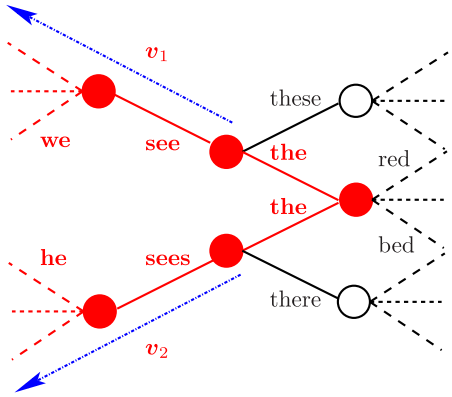


Fig. 5. An example of hidden vector based RNNLM history clustering.

method can also be used by beam search based decoders [37], [38]. However, due to the introduction of the distance beam γ, this technique is non-trivial to be directly used in generic WFST [24] based decoding approaches.

A range of distance measures may be considered for the distance measure $D(\boldsymbol{v}_{i-2}, \tilde{\boldsymbol{v}}_{j-2})$. 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. Sigmoid activation functions are used at the hidden layer for all RNNLMs in this paper. As they provide a well bounded dynamic range for the recurrent hidden history vector representation, the distance measure used in this paper is based on the Euclidean distance between $\boldsymbol{v}_{i-2}$ and $\tilde{\boldsymbol{v}}_{j-2}$. This is given by

$$D(\boldsymbol{v}_{i-2}, \tilde{\boldsymbol{v}}_{j-2}) = \frac{\sqrt{\sum_k (\boldsymbol{v}_{i-2,k} - \tilde{\boldsymbol{v}}_{j-2,k})^2}}{d} \tag{10}$$

where d is the dimensionality of the hidden history vectors.

C. Lattice Node Score Ranking Conditioned Cache Update

Both the n-gram and hidden vector based clustering schemes presented in this paper are implemented using an efficient cache based approach during lattice rescoring. One strength of this approach is that no explicit knowledge of existing acoustic and language model scores based rank ordering of the lattice paths is required during rescoring. Such generic feature allows the proposed RNNLM history clustering algorithms to be used also in first pass recognition. For example, the n-gram based history clustering approach was successfully applied to a weighted finite state transducer [24] based first pass decoding [26]. On the other hand, it may also introduce a performance sensitivity to the network traversing order during lattice rescoring. In this paper, lattice nodes are accessed in a topologically sorted order during rescoring in all experiments. Such traversing order does not guarantee the cached RNNLM histories to be sufficiently representative of the lattice paths of higher acoustic and language model scores. Hence, it introduces a performance sensitivity to the lattice traversing order during rescoring. Such sensitivity is expected to be more prominent when decreasing the truncated history length in n-gram based history clustering in equation (8),

or increasing the recurrent vector distance beam for the hidden vector based based history clustering in equation (9).

In order to address this issue, a lattice node scores rank ordering conditioned cache update scheme can be used. In addition to satisfying the respective cache hit conditions for the above two history clustering schemes, an additional constraint is introduced that the cached RNNLM probabilities must be computed using the history of a lattice node whose posterior probability (computed using lattice internal existing acoustic and n-gram LM scores) is equal or higher than that of the current lattice node being rescored. Otherwise, the cached RNNLM probabilities are updated using the current lattice node's history contexts. This allows the cached history clusters to be more representative of the lattice paths of higher rank ordering. The performance sensitivity to the lattice traversing order during rescoring can thus be reduced.

For example, consider the case of computing the RNNLM probability of a lattice node for word "bed" at the end of a sentence "I see the bed," when a 3-gram history clustering is used and additional lattice node score ranking conditioning enforced. If there is already a cached RNNLM history context "see the" previously computed using a different node for word "red" at the end of an alternative path "we see the red," but with a lower node posterior probability than the current lattice node "bed," an update of the cached RNNLM probabilities is required even if both histories "I see the" and "we see the" share the common most recent two words "see" and "the."

IV. Lattice Rescoring Using RNNLMs

All the RNNLM lattice rescoring experiments in this paper used an on-the-fly lattice expansion algorithm. A precursor of this algorithm was originally proposed for lattice rescoring using an interpolation between multiple n-gram LMs with context free or dependent interpolation weights [40], [41]. The lattice expansion process during an interpolated LM based rescoring uses a union between component n-gram LM context states. This allows the longest available distinct context histories modelled by the interpolated model to be preserved in the resulting expanded lattices.

The original algorithm proposed in [40] can not be directly applied to RNNLM based lattice rescoring. In this paper, this algorithm is further extended to support a much wider range of language models including back-off n-gram LMs, feedforward NNLMs, recurrent NNLMs and various forms of interpolation between them. A central part of the algorithm requires the LM state representation for the underlying LM being used. For example, for back-off n-gram LMs and feedforward NNLMs, this was previously defined in equation (5). For RNNLMs, the LM state was based on either equation (8) or (9) depending on the underlying history clustering technique being used. The LM state representation for an interpolated LM is again derived from a union between those of its component LMs. The corresponding pseudo-code algorithm for this on-the-fly lattice expansion method is given below. It was implemented and released in the current HTK version 3.5 [42] lattice processing tools.

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 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 $\mathbf{\Psi}(h_{n_0}^{n'_j})$ of expanded node n'_j;
13: compute LM probability $P(n_k|\mathbf{\Psi}(h_{n_0}^{n'_j}))$;
14: find a new LM state $\mathbf{\Psi}(h_{n_0}^{n_k})$ for node n_k;
15: **if** $\exists$ node $n'_l \in N'_k$ representing state $\mathbf{\Psi}(h_{n_0}^{n_k})$ **then**
16: $\mathbf{\Psi}(h_{n_0}^{n_k}) \leftarrow n'_l$;
17: **else**
18: add a new node n'_l to N'_k for state $\mathbf{\Psi}(h_{n_0}^{n_k})$;
19: $\mathbf{\Psi}(h_{n_0}^{n_k}) \leftarrow n'_l$;
20: **end if**
21: create a new arc a'_l from n'_j to n'_l;
22: assign score $\ln P(n_k|\mathbf{\Psi}(h_{n_0}^{n'_j}))$ to a'_l;
23: add arc a'_l to the expanded outbound arc list A'_i.
24: **end for**
25: **end for**
26: **end for**
27: Rebuild new network using $\{N'_i\}$ and $\{A'_i\}$.

In the above algorithm, depending on the underlying LM being applied in rescoring, the generic LM context state for a given history associated with a lattice path from node n_0 to n_k, $\Psi(h_{n_0}^{n_k})$, may take one of the following forms: a) the n-gram LM context state $\Psi_{\rm NG}(\cdot)$ in equation (5) for back-off LMs; b) the RNNLM context state $\Psi_{\rm RNN}(\cdot)$ approximated via equation (8) or (9) depending on the RNNLM history clustering scheme being used; c) or a union between the context states of an n-gram LM and RNNLM when a linear interpolation between them is used in rescoring. Line 9 of the pseudo code visits the expanded set of nodes $n'_j \in N'_i$ associated with an original lattice node n_i, while line 10 visits the original node n_i itself to access its outbound arcs.

V. Experiments and Results

In this section the performance of the proposed RNNLM lattice rescoring methods are evaluated using two HTK-based large vocabulary speech recognition systems. The first was developed for English conversational telephone speech used in the

2004 DARPA EARS evaluation [43]. The second system for Mandarin Chinese conversational speech was used in the 2014 DARPA BOLT evaluation [44]. A series of experiments were conducted on these two tasks.

A. Experiments on English CTS Data

The 2004 CU English CTS LVCSR system [43] was trained on approximately 2000 hours of Fisher conversational speech released by the LDC. A 59 k recognition word list was used in decoding. The system uses a multi-pass recognition framework. The initial lattice generation used gender dependent cross-word triphone acoustic models. These acoustic models include conversation side level normalization of PLP [45] features; HLDA [46], [47] projection; HMM parameter estimation using MPE [48]; and unsupervised MLLR [49] speaker adaptation. For efficiency, a pruned interpolated 3-gram LM was used in lattice generation prior to rescoring using a large unpruned 4-gram LM. The resulting lattices are then used in rescoring experiments to evaluate performance of various LMs. A more detailed description of the baseline system can be found in [43]. The 3 hour **dev04** set, which includes 72 Fisher conversation sides and contains on average 10.8 words per segment, was used as a test set. The 3 hour **eval04** set of a comparable number of Fisher conversations was also used.

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 [50], **UWWeb**, of 525 million words (weight 0.25). The 20 M words of **Fisher** data, which contains on average 12.7 words per sentence, was used to train a feedforward 4-gram NNLM "$\mathrm{nn}_{\mathrm{w4g}}$" using the OOS output layer architecture proposed in [3], and an RNNLM "rnn" using the comparable class-based OOS architecture in Fig. 2 of Section II with 500 output layer classes. A 38 k word input layer vocabulary and 20 k word output layer shortlist selected using 1-gram frequency counts with respective cut-offs of 1 and 2 were used for both the feedforward NNLM and RNNLM. Both NNLMs used a total of 500 hidden layer nodes. A total of 1 billion words of text data were generated from the baseline RNNLM "rnn" using the sampling technique described in [8] to train a 4-gram back-off LM "rnn.sample.4g" as an approximation to the original RNNLM. These three LMs (the feedforward 4-gram NNLM "$\mathrm{nn}_{\mathrm{w4g}}$," RNNLM "rnn" and sampling technique approximated 4-gram back-off LM "rnn.sample.4g") were then interpolated with the baseline 4-gram LM "w4g."

The 1-best and CN word error rates (WER) of various baseline LMs together with their perplexity performance are shown from the 1st to the 7th line in Table I. These include the back-off 4-gram LM "w4g," the feedforward NNLM system "w4g+$\mathrm{nn}_{\mathrm{w4g}}$," the RNNLM system "w4g+$\mathrm{rnn}.*$best" evaluated by re-ranking N-best lists of various depth from top 50 up to 10 k unique entries, and the RNNLM sampled data trained 4-gram LM "w4g+rnn.sample.4g." The RNNLM re-ranked N-best lists were then converted to prefix tree structured lattices [12] and used for CN decoding. The lattice density (Arcs/Sec) measure of the HTK formatted lattices for all the above baseline systems are also shown in the last column of Table I. For the RNNLM N-best rescoring baseline systems, the lattice density measure before and after N-best list prefix tree structuring (shown in brackets in the last column of Table I) are both given.

Consistent with the previous research reported in [8], the RNNLM sampled 1 billion word data trained 4-gram LM "w4g+rnn.sample.4g" (line 7 in Table I) gave comparable performance to the feedforward NNLM system "w4g+$\mathrm{nn}_{\mathrm{w4g}}$" (line 2 in Table I). Both systems obtained less than half of the total WER reductions produced by the 10 k-best RNNLM rescoring (line 6 in Table I) during 1-best and CN decoding over the 4-gram LM baseline "w4g" (1st line in Table I). Similarly a much reduced perplexity reduction of 0.9 points over the 4-gram LM baseline "w4g" was obtained using the sampling data trained 4-gram LM "w4g+rnn.sample.4g", in contrast of the 5.5 point perplexity reduction obtained using the non-approximated RNNLM (line 2 to 5 in Table I) .

Applying prefix tree structuring to N-bests lists [12] significantly reduced the size of the converted lattices. These are shown in brackets in the last column of Table I from line 3 to 6. As discussed in Section I, CN decoding requires a more compact lattice representation that encodes rich alternative hypotheses. In order to obtain the largest improvements from CN decoding, RNNLM rescored N-best lists need to be as deep as 10 k (line 6 in Table I). On the **dev04** set, this 10 k-best RNNLM rescoring baseline gave the lowest 1-best and CN WER of 15.3% and 15.0% respectively. It has a density of 10.2 k arcs/sec measured on the lattices converted from the prefix tree structured 10 k-best lists.

The performance of using the n-gram style approximation based RNNLM lattice rescoring method presented in Section III-A are shown from line 8 to 12 in Table I. Using these n-gram history approximated RNNLMs, perplexities comparable to the non-approximated baseline RNNLM (line 3 to 4 in Table I) were obtained. When the truncated history is increased to 5 words, the resulting 6-gram approximate RNNLM system produced 1-best and CN error rates of 15.4% and 15.0% respectively on **dev04**. Both results are comparable to the baseline 10 k-best RNNLM rescoring system (line 6 in Table I). Compared with the baseline 10 k-best RNNLM rescoring system, this 6-gram approximate RNNLM also gave a 70% relative reduction in lattice density from 10.2 k down to 3 k arcs/sec. Similar trends are also found on the **eval04** data. The 6-gram approximation gave the same CN error rate as the 10 k-best RNNLM rescoring baseline as well as a consistent reduction in lattice density by 70% relative. 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 suggested in Sections I and III of the decaying effect from the remote history contexts on the true RNNLM probabilities.

The performance of using the recurrent hidden history vector distance based RNNLM lattice rescoring method proposed in Section III-B is shown in the bottom section of Table I from lines 13 to 20. By adjusting the hidden vector distance beam γ in equation (9), a range of approximate RNNLMs

TABLE I
PERPLEXITY, 1-BEST, CN DECODING PERFORMANCE AND HTK LATTICE DENSITY MEASURED IN ARCS PER SECOND OBTAINED USING LMS ON FISHER **DEV04** AND **EVAL04** SETS

	dev04				eval04			
LM	PPlex	1best	CN	LatDensity	PPlex	1best	CN	LatDensity
1. w4g	51.8	16.7	16.1	421	52.1	19.1	18.7	430
2. w4g+$\mathrm{nn}_{\mathrm{w4g}}$	50.0	16.3	15.8	555	50.9	18.7	18.2	574
3. w4g+rnn.50best	46.3	15.4	15.4	188(97)	46.6	17.9	17.9	200(98)
4. w4g+rnn.100best		15.3	15.3	365(175)		17.9	17.7	389(177)
5. w4g+rnn.1000best		15.3	15.1	3416(1298)		17.8	17.6	3607(1313)
6. w4g+rnn.10000best		15.3	15.0	32 277(10 212)		17.8	17.5	33 607(10 275)
7. w4g+rnn.sample.4g	50.9	16.2	15.9	462	51.1	18.9	18.4	472
8. w4g+rnn.approx3g	46.4	15.8	15.4	428	46.7	18.4	17.9	478
9. w4g+rnn.approx4g	46.3	15.7	15.2	555	46.6	18.1	17.6	574
10. w4g+rnn.approx5g	46.3	15.6	15.1	1266	46.6	18.2	17.7	1305
11. w4g+rnn.approx6g	46.3	**15.4**	**15.0**	**3025**	46.6	**17.9**	**17.5**	**3068**
12. w4g+rnn.approx7g	46.3	15.4	15.0	7140	46.6	17.9	17.5	7146
13. w4g+rnn.hvd0.00450	46.4	15.8	15.4	465	46.6	18.4	17.6	478
14. w4g+rnn.hvd0.00300	46.3	15.6	15.2	539	46.6	18.1	17.7	556
15. w4g+rnn.hvd0.00200	46.3	15.6	15.1	699	46.6	18.1	17.6	720
16. w4g+rnn.hvd0.00100	46.3	15.6	15.1	1345	46.6	18.1	17.6	1367
17. w4g+rnn.hvd0.00075	46.3	15.5	15.1	1842	46.6	18.1	17.5	1857
18. w4g+rnn.hvd0.00050	46.3	**15.4**	**15.0**	**2818**	46.6	18.1	17.5	2762
19. w4g+rnn.hvd0.00025	46.3	15.4	15.0	4725	46.6	17.9	17.5	4500
20. w4g+rnn.hvd0.00001	46.3	15.4	15.0	6836	46.6	**17.9**	**17.4**	**6705**

"w4g" is a 4-gram back-off LM and "w4g+$\mathrm{nn}_{\mathrm{w4g}}$" 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 1 billion words of texts sampled from "rnn." "w4g+rnn.approx$*$g" and "w4g+rnn.hvd$*$" used n-gram and hidden vector distance based RNNLM history clustering respectively.

comparable in both perplexity and error rate to the truncated history based approach, but also giving more compact lattices, were produced. On the **dev04** set, for example, setting $\gamma = 0.002$ (line 15 in Table I) produced 1-best and CN error rates of 15.6% and 15.1% that are equivalent to the 5-gram history approximate "w4g+rnn.approx5g" system (line 10 in Table I), and a 45% reduction in lattice size from 1266 arcs/sec down to 699 arcs/sec. The best WER performance on the **dev04** data was obtained by setting $\gamma = 0.00050$ (line 18 in Table I). It 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 10 k-best rescoring system (line 6 in Table I), and the best n-gram history clustering rescoring system "w4g+rnn.approx6g" (line 11 in Table I) respectively. In practice, this "w4g+rnn.hvd0.00050" system can be used to rescore more heavily pruned lattices at a speed over 10 times faster than the 10 k-best rescoring system while producing comparable 1-best and CN error rates of 15.4% and 15.1% on **dev04**. On the **eval04** set, the best CN decoding performance was obtained by setting a larger hidden vector distance beam $\gamma = 0.00001$ (line 20 in Table I). It outperformed the 10 k-best rescoring system (line 6 in Table I) by 0.1% and reduced the lattice density by 35% relative from 10.2 k arcs/sec down to 6705 arcs/sec.

As discussed in Section III-C, in order to reduce the performance sensitivity to the lattice traversing order during rescoring, a lattice node score ranking conditioned cache update scheme can be used. The performance of a total of 8 different n-gram history clustering and hidden vector distance clustering based RNNLM scoring systems with the additional lattice node score ranking condition enforced in cache access are shown in Table II. Compared with the 3-gram based approximation "w4g+rnn.approx3g," and the hidden vector distance approximation "w4g+rnn.hvd0.00450" systems of Table I where no such score ranking condition is enforced, consistent 1-best and CN decoding WER reductions of 0.1%–0.3% were obtained for both systems. When higher oder n-gram history clustering or similarly tighter hidden vector distance beam settings are used, for example, using a 6-gram approximation or setting $\gamma = 0.00200$ and above, no further performance improvements were obtained. This is expected as the history clustering sensitivity to lattice traversing order has a larger impact on shorter n-gram context and larger vector distance beam based history clustering. Such sensitivity is reduced when higher order n-gram approximation or tighter vector distance beams increase the modelling precision over different context histories.

TABLE II
1-BEST, CN DECODING PERFORMANCE AND LATTICE DENSITY MEASURED IN ARCS PER SECOND OBTAINED USING RNNLMS WITH A LATTICE NODE POSTERIOR RANKING CONDITIONED CACHE UPDATE ON FISHER **DEV04** AND **EVAL04**

	dev04			eval04		
LM	1best	CN	LDen	1best	CN	LDen
8. w4g+rnn.approx3g	15.6	15.2	428	18.2	17.8	478
9. w4g+rnn.approx4g	15.6	15.2	555	18.1	17.6	574
10. w4g+rnn.approx5g	15.6	15.1	1266	18.1	17.6	1305
11. w4g+rnn.approx6g	15.4	15.0	3025	17.9	17.5	3068
13. w4g+rnn.hvd0.00450	15.7	15.1	566	18.1	17.6	587
14. w4g+rnn.hvd0.00300	15.6	15.2	612	18.1	17.6	636
15. w4g+rnn.hvd0.00200	15.6	15.1	748	18.0	17.6	774
16. w4g+rnn.hvd0.00100	15.6	15.1	1379	18.1	17.6	1406

Naming convention same as Table I.

TABLE III
PERPLEXITY, 1-BEST, CN DECODING CHARACTER ERROR RATE AND HTK LATTICE DENSITY MEASURED IN ARCS PER SECOND OBTAINED USING LMS ON CONVERSATIONAL MANDARIN **DEV14** AND **EVAL97** SETS

	dev14				eval97			
LM	PPlex	1best	CN	LatDensity	PPlex	1best	CN	LatDensity
1. w4g	151.4	35.7	35.3	273	140.0	31.3	31.1	273
2. w4g+rnn.50best	127.1	34.7	34.6	185(102)	125.0	30.5	30.5	238(128)
3. w4g+rnn.100best		34.6	34.5	360(187)		30.5	30.5	458(232)
4. w4g+rnn.1000best		34.5	34.4	3329(1417)		30.5	30.4	3861(1610)
5. w4g+rnn.10000best		34.6	34.3	30 597(11 007)		30.5	30.4	31 423(10 848)
6. w4g+rnn.sample.4g	140.6	35.2	34.8	310	135.0	31.2	30.9	326
7. w4g+rnn.approx3g	127.5	34.8	34.5	305	130.0	30.9	30.5	295
8. w4g+rnn.approx4g	127.1	34.8	34.4	554	125.3	30.7	30.4	476
9. w4g+rnn.approx5g	127.1	34.7	34.3	1285	124.9	30.7	30.4	995
10. w4g+rnn.approx6g	127.1	**34.7**	**34.2**	**2852**	124.8	**30.6**	**30.3**	**2026**
11. w4g+rnn.approx7g	127.1	34.6	34.3	6012	124.9	30.7	30.4	3884
11. w4g+rnn.approx8g	127.1	34.7	34.2	12 695	124.8	30.6	30.3	7317
12. w4g+rnn.hvd0.00900	127.3	34.8	34.5	500	125.4	30.8	30.5	432
13. w4g+rnn.hvd0.00800	127.1	34.8	34.4	564	125.3	30.9	30.5	477
14. w4g+rnn.hvd0.00700	127.1	34.9	34.3	658	125.3	30.8	30.5	539
15. w4g+rnn.hvd0.00600	127.1	34.8	34.4	802	125.3	30.8	30.5	636
16. w4g+rnn.hvd0.00500	127.1	34.8	34.3	1034	125.2	30.7	30.5	781
17. w4g+rnn.hvd0.00400	127.1	34.7	34.3	1430	125.2	30.6	30.4	1020
18. w4g+rnn.hvd0.00300	127.1	34.7	34.3	2112	125.1	30.6	30.5	1440
19. w4g+rnn.hvd0.00195	127.1	**34.6**	**34.2**	**3501**	125.1	30.6	30.3	2287
20. w4g+rnn.hvd0.00185	127.1	34.6	34.2	3705	125.1	30.6	30.3	2404
21. w4g+rnn.hvd0.00175	127.1	34.6	34.2	3905	125.1	**30.5**	**30.3**	**2533**
22. w4g+rnn.hvd0.00150	127.1	34.6	34.2	4427	125.1	30.6	30.4	2895
23. w4g+rnn.hvd0.00100	127.1	34.6	34.2	5652	125.0	30.6	30.3	3603
24. w4g+rnn.hvd0.00010	127.1	34.6	34.2	8098	125.0	30.5	30.3	5916
25. w4g+rnn.hvd0.00001	127.1	34.6	34.2	8215	125.0	30.6	30.3	6052

Naming convention the same as Table I.

B. Experiments on Mandarin Chinese CTS Data

The 2014 CU CTS Mandarin Chinese LVCSR system [44] was then used to further evaluate the two proposed RNNLM lattice rescoring methods. The system was trained on 300 hours of Mandarin Chinese conversational telephone speech data released by the LDC for the DARPA BOLT program. A 63 k recognition word list was used in decoding. The system uses the same multi-pass recognition framework as described in Section V-A.

The initial lattice generation stage used CMLLR [51] based speaker adaptively trained cross-word triphone tandem [52] HMM acoustic models with MPE [48] based parameter estimation and unsupervised MLLR [49] speaker adaptation. HLDA [46], [47] projected and speaker level normalized PLP [45] features augmented with pitch features were used. 26 dimensional DNN bottle neck features [53] extracted from a deep neural network [54] consisting of 4 hidden layers of 1 k nodes each and modelling 6 k context dependent states at the output layer, were also used. An interpolated 4-gram baseline LM was used. A 4.5 hour test set of Mandarin Chinese conversational telephone speech data used in the BOLT program, **dev14**, consisting of 57 speakers from 19 conversations, was used for performance evaluation. An additional 1.6 hour test set, **eval97**, consisting of 49 speakers from 20 conversations, was also used. Manual audio segmentation was also used to allow translation outputs to be accurately scored.

The baseline 4-gram back-off LM "w4g" was trained using a total of 1 billion words from the following two types of text sources: 2.6 M words of acoustic transcripts including the LDC Call Home Mandarin (CHM), Call Friend Mandarin (CFM) and HKUST collected conversational Mandarin telephone speech data (weight 0.78); 1 billion words of additional web data collected under the DARPA EARS and GALE programs (weight 0.22). The acoustic transcripts contain on average 7.5 words per sentence. This baseline 4-gram LM has a total of 48 M 2-grams, 133 M 3-grams and 143 M 4-grams. It gave a perplexity score of 151.4, 1-best and CN character error rates (CER) of 35.7% and 35.3% respectively on **dev14**. These results are shown in the 1st line in Table III.

In order to further improve the RNNLM's coverage and generalization, the 2.6 M words of acoustic transcripts data were augmented with 15 M words of its paraphrase variants. These were automatically produced using the statistical paraphrase induction and generation method described in [55]. The above combined data set was then used to train a paraphrastic RNNLM [56] "rnn" on a GPU in bunch mode [34]. The full output layer with an OOS node based RNNLM architecture in Fig. 1 of Section II was used. A total of 512 hidden layer nodes were used. A 27 k word input layer vocabulary and 20 k word output layer shortlist were also used. In common with the previous experiments of Section V-A, a total of 1 billion words of text data were also generated from the RNNLM "rnn" using the same sampling technique described in [8] to train a 4-gram back-off LM "rnn.sample.4g" as an approximation. Both the RNNLM and the sampled data trained 4-gram LM were then interpolated with the baseline 4-gram LM "w4g" for performance evaluation.

The perplexity, 1-best and CN decoding CER performance of the baseline RNNLM and various approximation schemes are shown in Table III. Consistent with the trend previously found in Table I, the sampling approach based RNNLM approximation "w4g+rnn.sample.4g" (line 6 in Table III) only retained a part of the improvement of the original RNNLM (line 2 to 5 in Table III) over the baseline 4-gram LM in terms of both perplexity and error rate. Using the prefix tree structured N-bests lists again significantly reduced the density of the resulting lattices. The best CN decoding performance was obtained using a 10 k-best RNNLM rescoring baseline system (line 5 in Table III). On the **dev14** data, it gave a 1-best and CN CER of 34.6% and 34.3% respectively. It has a density of 11 k arcs/sec measured on the lattices converted from the prefix tree structured 10 k-best lists.

The performance of the n-gram history clustering based RNNLM lattice rescoring of Section III-A are shown from line 7 to 11 in Table III. A 6-gram approximate RNNLM system produced 1-best and CN error rates of 34.7% and 34.2% respectively on **dev14**. Both results are comparable to the baseline 10 k-best RNNLM rescoring (line 5 in Table III). It also gave a significant 74% reduction in lattice density from 11 k to 2852 arcs/sec. Further increasing the truncated history to 6 words or more gave no improvement while only increased the resulting lattice size. A similar trend is also found on the **eval97** data.

The performance of using the recurrent hidden history vector distance based RNNLM lattice rescoring method proposed in Section III-B with varying distance beam settings are also shown in Table III from line 12 to 25. On the **dev14** set, setting the hidden vector distance beam $\gamma = 0.00195$ (line 19 in Table III) gave the best CER performance among all systems in Table III. This approximate system gave a 1-best and CN error rates of 34.6% and 34.2% respectively. It also gave a 68.2% relative reduction in lattice density over the prefix tree structured 10 k-best rescoring system (line 5 in Table III) from 11 k down to 3501 arcs/sec. On the **eval97** set, the best performance was obtained by setting the vector distance beam $\gamma = 0.00175$ (line 21 in Table III). It outperformed the 10 k-best rescoring system by 0.1% in CN decoding. It also reduced the lattice density by 77% relative from 11 k arcs/sec down to 2533 arcs/sec.

VI. Conclusion and Future Work

Two efficient lattice rescoring methods for RNNLMs were investigated in this paper. The proposed techniques produced 1-best and confusion network decoding performance comparable with a 10 k-best rescoring RNNLM baseline systems on two large vocabulary conversational telephone speech recognition tasks for US English and Mandarin Chinese. These methods also produced highly compact lattice representation after RNNLM rescoring. Consistent compression in lattice size was obtained over the prefix tree structured n-best rescoring RNNLM baseline systems. These results demonstrate the advantages of the proposed techniques over the standard N-best rescoring framework, as well as their strong generalization and applicability to multiple languages and tasks. Future research will focus on further improving efficiency in decoding using RNNLMs.

References

[1] Y. Bengio and R. Ducharme, "A neural probabilistic language model," *J. Mach. Learn. Res.*, vol. 3, pp. 1137–1155, 2003.

[2] H. Schwenk, "Continuous space language models," *Comput. Speech Lang.*, vol. 21, no. 3, pp. 492–518, 2007.

[3] J. Park, X. Liu, M. J. F. Gales, and P. C. Woodland, "Improved neural network based language modelling and adaptation," in *Proc. Int. Conf. Spoken Lang. Process. Interspeech*, Makuhari, Japan, 2010, pp. 1041–1044.

[4] T. Mikolov, M. Karafiat, L. Burget, J. Cernocky, and S. Khudanpur, "Recurrent neural network based language model," in *Proc. Int. Conf. Spoken Lang. Process. Interspeech*, Makuhari, Japan, 2010, pp. 1045–1048.

[5] T. Mikolov, S. Kombrink, L. Burget, J. H. Cernocky, and S. Khudanpur, "Extensions of recurrent neural network language model," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, Prague, 2011, pp. 5528–5531.

[6] M. Sundermeyer, R. Schlüter, and H. Ney, "LSTM neural networks for language modeling," in *Proc. Int. Conf. Spoken Lang. Process. Interspeech*, Portland, OR, USA, 2012, pp. 194–197.

[7] H.-S. Le, I. Oparin, A. Allauzen, J. Gauvain, and F. Yvon, "Structured output layer neural network language models for speech recognition," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 21, no. 1, pp. 197–206, Jan. 2013.

[8] A. Deoras, T. Mikolov, S. Kombrink, M. Karafiat, and S. Khudanpur, "Variational approximation of long-span language models for LVCSR," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, Prague, Czech Republic, 2011, pp. 5532–5535.

[9] G. Lecorvé and P. Motlicek, "Conversion of recurrent neural network language models to weighted finite state transducers for automatic speech recognition," in *Proc. ISCA Interspeech*, Portland, OR, USA, pp. 1668–1671, 2012.

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

[11] M. Sundermeyer, I. Oparin, J. L. Gauvain, B. Freiberg, R. Schlüter, and H. Ney, "Comparison of feedforward and recurrent neural network language models," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, Vancouver, Canada, 2013, pp. 8430–8434.

[12] Y. Si, Q. Zhang, T. Li, J. Pan, and Y. Yan, "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.

[13] K. Yao, G. Zweig, M.-Y. Hwang, Y. Shi, and D. Yu, "Recurrent neural networks for language understanding," in *Proc. ISCA Interspeech*, Lyon, France, 2013, pp. 2524–2528.

[14] M. Auli, M. Galley, C. Quirk, and G. Zweig, "Joint language and translation modeling with recurrent neural networks," in *Proc. ACL Conf. Empirical Methods Natural Lang. Process.*, Seattle, WA, USA, 2013, pp. 1044–1054.

[15] N. Kalchbrenner and P. Blunsom, "Recurrent continuous translation models," in *Proc. ACL Conf. Empirical Methods Natural Lang. Process.*, 2013, pp. 1700–1709.

[16] J. Devlin, R. Zbib, Z. Huang, T. Lamar, R. Schwartz, and J. Makhoul, "Fast and robust neural network joint models for statistical machine translation," in *Proc. 52nd Annu. Meeting Assoc. Comput. Linguistics*, Baltimore, MD, USA, 2014, pp. 1370–1380.

[17] T. Hori, Y. Kubo, and A. Nakamura, "Real-time one-pass decoding with recurrent neural network language model for speech recognition," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, Florence, Italy, 2014, pp. 6414–6418.

[18] L. Mangu, E. Brill, and A. Stolcke, "Finding consensus in speech recognition: Word error minimization and other applications of confusion networks," *Comput. Speech Lang.*, vol. 14, no. 4, pp. 373–400, 2000.

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

[20] A. Deoras, T. Mikolov, and K. Church, "A fast re-scoring strategy to capture long-distance dependencies," in *Proc. ACL Conf. Empirical Methods Natural Lang. Process.*, Edinburgh, U.K., 2011, pp. 1116–1127.

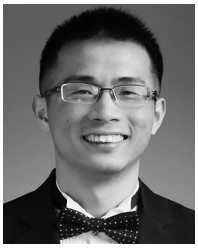
[21] S. Jalalvand and D. Falavigna, "Direct word graph rescoring using A* search and RNNLM," in *Proc. ISCA Interspeech*, 2014, pp. 2630–2634.

[22] M. Sundermeyer, Z. Tüske, R. Schlüter, and H. Ney, "Lattice decoding and rescoring with long-span neural network language models," in *Proc. ISCA Interspeech*, Singapore, 2014, pp. 661–665.

[23] M. Sundermeyer, H. Ney, and R. Schlüter, "From feedforward to recurrent LSTM neural networks for language modeling," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 23, no. 3, pp. 517–529, Mar. 2015.
[24] M. Mohri, "Finite-state transducers in language and speech processing," *Comput. Linguistics*, vol. 23, no. 2, pp. 269–311, 1997.
[25] X. Liu, Y. Wang, X. Chen, M. J. F. Gales, and P. C. Woodland, "Efficient lattice rescoring using recurrent neural network language models," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, Florence, Italy, 2014, pp. 4941–4945.
[26] Z. Huang, G. Zweig, and B. Dumoulin, "Cache based recurrent neural network language model inference for first pass speech recognition," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, Florence, Italy, 2014, pp. 6404–6408.
[27] A. Emami and L. Mangu, "Empirical study of neural network language models for Arabic speech recognition," in *Proc. IEEE Workshop Autom. Speech Recog. Understanding*, Kyoto, Japan, 2007, pp. 147–152.
[28] F. Morin and Y. Bengio, "Hierarchical probabilistic neural network language model," in *Proc. Int. Workshop Artif. Intell. Statist.*, Barbados, 2005, pp. 246–252.
[29] D. E. Rumelhart, G. E. Hintont, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.
[30] M. Boden, "A guide to recurrent neural networks and backpropagation," The Dallas project, SICS technical report (2002).
[31] Y. Shi, M.-Y. Hwang, K. Yao, and M. Larson, "Speed up of recurrent neural network language models with sentence independent subsampling stochastic gradient descent," in *Proc. ISCA Interspeech*, Lyon, France, 2013, pp. 1203–1207.
[32] Z. Huang, G. Zweig, M. Levit, B. Dumoulin, B. Oguz, and S. Chang, "Accelerating recurrent neural network training via two stage classes and parallelization," in *Proc. IEEE Workshop Autom. Speech Recog. Understanding*, Olomouc, Czech Republic, 2013, pp. 326–331.
[33] B. Li, E. Zhou, B. Huang, J. Duan, Y. Wang, N. Xu, J. Zhang, and H. Yang, "Large scale recurrent neural network on GPU," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, Beijing, China, 2014, pp. 4062–4069.
[34] X. Chen, Y. Wang, X. Liu, M. J. F. Gales, and P. C. Woodland, "Efficient GPU-based training of recurrent neural network language models using spliced sentence bunch," in *Proc. ISCA Interspeech*, Singapore, 2014, pp. 641–645.
[35] X. Chen, X. Liu, M. J. F. Gales, and P. C. Woodland, "Improving the training and evaluation efficiency of recurrent neural network language models," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, Brisbane, 2015, pp. 5401–5405.
[36] T. Mikolov, S. Kombrink, L. Burget, J. H. Cernocky, and S. Khudanpur, "RNNLM—Recurrent neural network language modeling toolkit," in *Proc. IEEE Autom. Speech Recognition Understanding Workshop*, 2011.
[37] J. J. Odell, V. Valtchev, P. C. Woodland, and S. J. Young, "A one pass decoder design for large vocabulary recognition," in *Proc. ACL Workshop Human Lang. Technol.*, Stroudsburg, PA, USA, 1994, pp. 405–410.
[38] H. Ney and S. Ortmanns, "Dynamic programming search for continuous speech recognition," *IEEE Signal Process. Mag.*, vol. 16, no. 5, pp. 64–83, Sep. 1999.
[39] T. R. Niesler and P. C. Woodland, "A variable-length category-based n-gram language model," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, Atlanta, GA, USA, 1996, vol. 1, pp. 164–167.
[40] X. Liu, M. J. F. Gales, and P. C. Woodland, "Use of contexts in language model interpolation and adaptation," *Comput. Speech Lang.*, vol. 27, no. 1, pp. 301–321, 2013.
[41] X. Liu, M. J. F. Gales, J. L. Hieronymus, and P. C. Woodland, "Language model combination and adaptation using weighted finite state transducers," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, Dallas, TX, USA, 2010, pp. 5390–5393.
[42] S. Young G. Evermann, M. J. F. Gales, T. Hain, D. Kershaw, X. Liu, G. Moore, J. Odell, D. Ollason, D. Povey, A. Ragni, V. Valtchev, P. C. Woodland, and C. Zhang, "The HTK Book Version 3.5a," Speech Research Group, Cambridge University Engineering Department, 2015.
[43] G. Evermann, H. Y. Chan, M. J. F. Gales, B. Jia, D. Mrva, P. C. Woodland, and K. Yu, "Training LVCSR systems on thousands of hours of data," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, Philadelphia, PA, USA, 2005, vol. 1, pp. 209–212.
[44] X. Liu, F. Flego, L. Wang, C. Zhang, M. J. F. Gales, and P. C. Woodland, "The cambridge university 2014 BOLT conversational telephone Mandarin Chinese LVCSR system for speech translation," in *Proc. ISCA Interspeech*, Dresden, Germany, 2015, pp. 3145–3149.
[45] P. C. Woodland, M. J. F. Gales, D. Pye, and S. J. Young, "The development of the 1996 HTK broadcast news transcription system," in *Proc. DARPA Speech Recog. Workshop*, Harriman, NY, USA, 1996, pp. 73–78.
[46] N. Kumar, "Investigation of silicon-auditory models and generalization of linear discriminant analysis for improved speech recognition," Ph.D. dissertation, John Hopkins Univ., Baltimore, MD, USA, 1997.
[47] X. Liu, M. J. F. Gales, and P. C. Woodland, "Automatic complexity control for HLDA systems," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, Hong Kong, China, 2003, vol. 1, pp. 132–135.
[48] D. Povey and P. C. Woodland, "Minimum phone error and I-smoothing for improved discriminative training," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, Orlando, FL, USA, 2002, vol. 1, pp. 105–108.
[49] C. J. Leggetter and P. C. Woodland, "Maximum likelihood linear regression for speaker adaptation of continuous density HMMs," *Comput. Speech Lang.*, vol. 9, pp. 171–186, 1995.
[50] I. Bulyko, M. Ostendorf, and A. Stolcke, "Getting more mileage from web text sources for conversational speech language modeling using class-dependent mixtures," in *Proc. Conf. North Amer. Ch. Assoc. Comput. Linguistics Human Lang. Technol.*, Stroudsburg, PA, USA, 2003, vol. 2 pp. 7–9.
[51] M. J. F. Gales, "Maximum likelihood linear transformations for HMM-based speech recognition," *Comput. Speech Lang.*, vol. 12, no. 2, pp. 75–98, 1998.
[52] H. Hermansky, D. Ellis, and S. Sharma, "Tandem connectionist feature extraction for conventional HMM systems," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, Istanbul, Turkey, 2000, vol. 3, pp. 1635–1638.
[53] D. Yu and M. L. Seltzer, "Improved bottleneck features using pretrained deep neural networks," in *Proc. ISCA Interspeech*, Florence, Italy, pp. 237–240, 2011.
[54] G. Dahl, D. Yu, L. Deng, and A. Acero, "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition," *IEEE/ACM Trans. Audio, Speech, Language Process.*, vol. 20, no. 1, pp. 30–42, Jan. 2012.
[55] X. Liu, M. J. F. Gales, and P. C. Woodland, "Paraphrastic language models," *Comput. Speech Lang.*, vol. 28, no. 6, pp. 1298–1316, 2014.
[56] X. Liu, M. J. F. Gales, and P. C. Woodland, "Paraphrastic recurrent neural network language models," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, Brisbane, Australia, 2015, pp. 5406–5410.

Xunying Liu received the bachelor's degree from Shanghai Jiao Tong University, the Ph.D. degree in speech recognition, and the M.Phil. degree in computer speech and language processing both from the University of Cambridge, Cambridge, UK. He has been a Senior Research Associate at the Machine Intelligence Laboratory, Cambridge University Engineering Department, and from 2016 an Associate Professor in the Department of Systems Engineering and Engineering Management, the Chinese University of Hong Kong. He received the Best Paper Award at ISCA Interspeech 2010. His current research interests include large vocabulary continuous speech recognition, language modelling, noise robust speech recognition, speech synthesis, speech and language processing. He is a Member of ISCA.

Xie Chen received the bachelor's degree at Xiamen University, Fujian, China, in 2009 and the M.Phil. degree in electronic engineering from Tsinghua University, Beijing, China, in 2012. He is currently working toward the PhD degree at the Machine Intelligence Laboratory, working on automatic speech recognition, supervised by Prof. M. Gales. He joined the Cambridge University Engineering Department in 2012. His current research interests include large vocabulary continuous speech recognition, meeting transcription, and language modelling. He is a Student Member of ISCA.



Yongqiang Wang received the B.Eng. degree in electronic engineering from the University of Science and Technology of China, Hefei, China, the M.Phil. degree in computer science from the University of Hong Kong, Pok Fu Lam, Hong Kong, and the Ph.D. degree in engineering from the University of Cambridge, Cambridge, U.K. He is currently a Senior Speech Scientist with Microsoft. His research interests include robust speech recognition and large-scale deep learning.

Mark J. F. Gales received the B.A. degree in electrical and information sciences from the University of Cambridge, Cambridge, U.K., from 1985 to 1988. Following graduation, he was a Consultant at Roke Manor Research Ltd. In 1991, he took up a position as a Research Associate in the Speech Vision and Robotics group in the Engineering Department, Cambridge University. In 1995, he completed his doctoral thesis: Model-Based Techniques for Robust Speech Recognition supervised by Professor S. Young. From 1995 to 1997, he was a Research Fellow at Emmanuel College Cambridge. He was then a Research Staff Member in the Speech group at the IBM T.J.Watson Research Center until 1999 when he returned to the Cambridge University Engineering Department as a University Lecturer. He was appointed Reader in information engineering in 2004. He is currently a Professor of information engineering and a College Lecturer and Official Fellow of Emmanuel College. He a member of the Speech and Language Processing Technical Committee (2015–2017, previously a member from 2001–2004). He was an Associate Editor for the IEEE SIGNAL PROCESSING LETTERS from 2008 to 2011 and the IEEE TRANSACTIONS ON AUDIO, SPEECH, AND LANGUAGE PROCESSING from 2009 to 2013. He is currently on the Editorial Board of *Computer Speech and Language*. He received a number of paper awards, including a 1997 IEEE Young Author Paper Award for his paper on Parallel Model Combination and a 2002 IEEE Paper Award for his paper on Semi-Tied Covariance Matrices.

Philip C. Woodland is a Professor of information engineering at the University of Cambridge Engineering Department, where he leads the Speech Group, and a Professorial Fellow of Peterhouse. He has published almost 200 papers in speech technology, including the most cited paper in Computer Speech and Language. He developed techniques for speaker adaptation and discriminative training that have now become standard in speech recognition. His research team developed speech recognition systems which have frequently been the most accurate in international research evaluations organised by the US Government. He is well known as one of the original coauthors of the widely used HTK toolkit and has continued to play a major role in its development. He has been a member of the editorial board of *Computer Speech and Language* (1994–2009), and a current editorial board member of *Speech Communication*. One of his current major interests is developing flexible systems that can adapt to a wide range of speakers, acoustic conditions, and speaking styles with relatively limited training resources. His current work also includes techniques for improved language modelling and confidence estimation. An increasing trend in his work is the use of deep neural networks for both acoustic models and language models.He is a Fellow of ISCA.