EURASIP Journal on
Audio, Speech, and Music Processing
a SpringerOpen Journal

RESEARCH Open Access

Empirically combining unnormalized NNLM and back-off *N*-gram for fast *N*-best rescoring in speech recognition

Yongzhe Shi*, Wei-Qiang Zhang, Meng Cai and Jia Liu

Abstract

Neural network language models (NNLM) have been proved to be quite powerful for sequence modeling, including feed-forward NNLM (FNNLM), recurrent NNLM (RNNLM), etc. One main issue concerned for NNLM is the heavy computational burden of the output layer, where the output needs to be probabilistically normalized and the normalizing factors require lots of computation. How to fast rescore the *N*-best list or lattice with NNLM attracts much attention for large-scale applications. In this paper, the statistic characteristics of normalizing factors are investigated on the *N*-best list. Based on the statistic observations, we propose to approximate the normalizing factors for each hypothesis as a constant proportional to the number of words in the hypothesis. Then, the unnormalized NNLM is investigated and combined with back-off *N*-gram for fast rescoring, which can be computed very fast without the normalization in the output layer, with the complexity reduced significantly. We apply our proposed method to a well-tuned context-dependent deep neural network hidden Markov model (CD-DNN-HMM) speech recognition system on the English-Switchboard phone-call speech-to-text task, where both FNNLM and RNNLM are trained to demonstrate our method. Experimental results show that unnormalized probability of NNLM is quite complementary to that of back-off *N*-gram, and combining the unnormalized NNLM and back-off *N*-gram can further reduce the word error rate with little computational consideration.

Keywords: Neural network language model; *N*-best rescoring; Speech recognition

1 Introduction

The output of the speech-to-text (STT) system is usually a multi-candidate form encoded as lattice or *N*-best list. Rescoring via more accurate models, as a second pass of the STT system, has been widely used to further improve the performance. Fast rescoring with neural network language models is investigated in the paper.

Neural network language models (NNLMs), including feed-forward NNLM (FNNLM) [1,2] and recurrent NNLM (RNNLM) [3-5], have achieved very good results on many tasks [6-8], especially for RNNLM. Distributed word representations and the associated probability estimates are jointly computed in a feed-forward or recurrent neural network architecture. This approach provides automatic smoothing and leads to better generalization for unseen *N*-grams. The main drawback of NNLM is the great computational burden of the output layer that contains tens of thousands of nodes corresponding to the words in the vocabulary, where the output needs to be probabilistically normalized for each word with the softmax function and this softmax-normalization requires lots of computations. Thus, *N*-best list for its simplicity is usually rescored and reranked by NNLM, and the evaluation speed of NNLM needs to be improved further for large-scale applications.

Most of the previous work focuses on the speedup of the training of NNLM via word clustering to structure the output layer [4,9,10]. One typical method, the class-based output layer method, was proposed, recently, for speeding up RNNLM training [4], based on word frequency. This method divides the cumulative probability into *C* partitions to form *C* frequency binnings which correspond

*Correspondence: shiyz09@gmail.com
Tsinghua National Laboratory for Information Science and Technology, Department of Electronic Engineering, Tsinghua University, Beijing 100084, China

to C clusters. The words are assigned to classes proportionally. Based on the frequency clustering method, the closed-form solution of the output layer complexity can be written as $O((C + |\mathbf{V}|/C)H)$, where $|\mathbf{V}|$ and H denote the number of nodes in the output layer and the hidden layer, respectively. Another method [9,11,12] is to factorize the output layer with a tree structure that needs to be carefully constructed based on expert knowledge [13] or other clustering method [14]. Although the structure-based methods can speed up the evaluation of NNLM, the complexities of these methods are still quite high in real-time systems.

In this paper, the statistic characteristics of normalizing factors are investigated for the N-best hypotheses. Based on the statistic observations, we proposed to approximate the normalizing factors for each hypothesis as a constant proportional to the number of words in the hypothesis, and the normalizing factors can be easily absorbed into the word penalty. Then, the unnormalized NNLM is investigated and combined with back-off N-gram for fast rescoring, which can be computed very fast without the normalization in the output layer, with the complexity reduced significantly.

We apply our proposed method to a well-tuned context-dependent deep neural network hidden Markov model (CD-DNN-HMM) speech recognition system on the English-Switchboard speech-to-text task. Both feed-forward NNLM and recurrent NNLM are well-trained to verify the effectiveness of our method. Experimental results show that unnormalized probability of NNLM is quite complementary to that of back-off N-gram, and combining the unnormalized NNLM and back-off N-gram can further improve the performance of speech recognition with little computational resource.

As our method is theoretically founded on the statistic observations, we first introduce the experimental setup, including the speech recognizer, N-best hypotheses, NNLM structure, and NNLM training, in Section 2 for convenience. The remainder of this paper is organized as follows: The statistics of the normalizing factors on the hypotheses are investigated and the constant normalizing factor approximation is proposed in Section 3. How to combine the unnormalized NNLM and back-off N-gram is presented in Section 4, followed by complexity analysis and speed comparisons in Section 5. Detailed experimental evaluations for N-best rescoring are presented in Section 6. Discussions on the related work are given in Section 7, followed by the conclusions in Section 8.

2 Experimental setup

The experimental setup for the speech recognizer, N-best hypotheses, the NNLM structure, and the NNLM training in our work was introduced here, since our method is theoretically founded on statistical observations.

2.1 Speech recognizer and *N*-best hypotheses

The effectiveness of our proposed method is evaluated on the STT task with the 309-hour Switchboard-I training set [15]. The 13-dimensional perceptual linear prediction features (PLP) with rolling-window mean-variance normalization and up to third-order derivatives are reduced to 39 dimensions by heteroscedastic linear discriminant analysis (HLDA). The speaker-independent three-state cross-word triphones share 9,308 tied states. The GMM-HMM baseline system has 40-Gaussian mixtures per state, trained with maximum likelihood (ML), and refined discriminatively (DT) with the minimum phone error (MPE) criterion. The well-tuned CD-DNN-HMM system replaces the Gaussian mixtures with scaled likelihoods derived from DNN posteriors. The input to the DNN contains 11 (5-1-5) frames of 39-dimensional features, where the DNN uses the architecture of 429-2048×7-9308. The data for system development is the 1831-segment Switchboard part of the NIST 2000 Hub5 eval set (Hub5'00-SWB). The Fisher half of the 6.3h Spring 2003 NIST rich transcription set (RT03S-FSH) acts as the evaluation set.

The 2000h Fisher transcripts, containing about 23 million words, are taken as our training corpus for language modeling. Based on Kneser-Ney smoothing, a back-off trigram language model (KN3) was trained on the 2000h Fisher transcripts for decoding, where the vocabulary is limited to 53K words and unknown words are mapped into a special token <unk>. Note that no additional text is used to train LMs for interpolations to ensure the repeatability. The out-of-vocabulary rate is 0.80% for the training corpus, 0.53% for the development corpus, and 0.017% for the evaluation corpus. The pronouncing dictionary comes from CMU [16]. The HDecode[a] command is used to decode the utterance with KN3 to output the lattice, and then the N-best hypotheses are extracted from the lattice using the lattice-tool[b] command. In the setup, top 100-best hypotheses are rescored and reranked by other language models, such as back-off 5-gram, FNNLM, and RNNLM, to improve the performance.

2.2 Structure and training of NNLM

The typical structures of NNLMs are shown in Figures 1 and 2, corresponding to FNNLM and RNNLM, respectively. We also define $\mathbf{V}$, H and N as the vocabulary, the size of hidden layer and the order of FNNLM, respectively. The projection matrix $\boldsymbol{E} \in \Re^{|\mathbf{V}|\times H}$ maps each word to the feature vector as the distributional representation and fed into the hidden layer.

Based on the structures of NNLM, the hidden state $\boldsymbol{h}_t$ of FNNLM can be computed as $\boldsymbol{h}_t = \tanh\left(\sum_{o=1}^{N-1} \boldsymbol{W}_{iho}\boldsymbol{v}_{t-o}\right)$, while that of RNNLM can be computed as $\boldsymbol{h}_t = \text{sigmoid}(\boldsymbol{W}_{hh}\boldsymbol{h}_{t-1}+\boldsymbol{v}_t)$, where tanh and sigmoid are the activation functions. The probability of the next word is computed via the softmax function in the

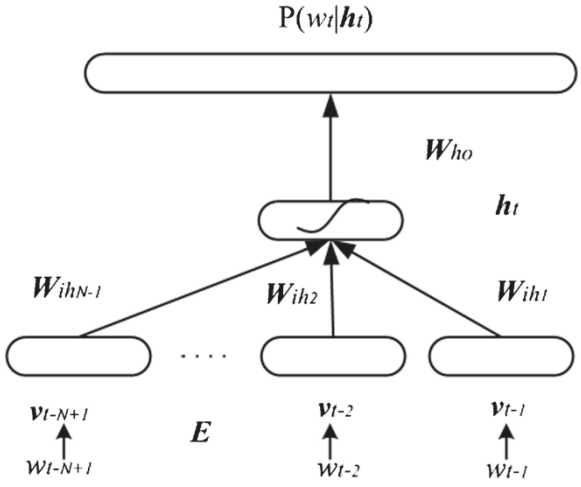

Figure 1 Structure of feed-forward neural network language model.

output layer, where $\boldsymbol{W}_{ho} \in \Re^{|\mathbf{V}|\times H} = [\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_{|\mathbf{V}|}]^T$ is the predicting matrix and $\boldsymbol{\theta}_{\forall i} \in \Re^{H\times 1}$ corresponds to each output node.

The transcripts of the Hub5'00-SWB set and the RT03S-FSH set act the development set and the evaluation set, respectively, for NNLM training. One FNNLM and one RNNLM are well-trained on the training corpus with the open source toolkits, CSLM [17] and RNNLM [18], respectively, where both of the hidden layers contain 300 nodes.

To speed up the training of the RNNLM, a frequency-based partition method [4] is used to factorize the output layer with 400 classes. The truncated backpropagation through time algorithm (BPTT) [19] is used to train the RNNLM with 10 time steps, with the initial learning rate set to 0.1. The learning rate is halved, when the perplexity decreases very slowly or increases. On the contrary, the training of FNNLM can be speeded up with

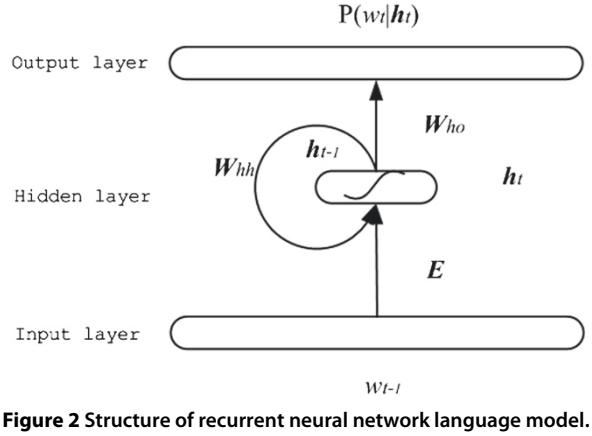

Figure 2 Structure of recurrent neural network language model.

128 context-word pairs as a mini-batch based on GPU implementation, so that no class layer was used, as the class layer usually sacrifices the performance of NNLM for speedup. The learning rate is empirically set as $lr = lr_0/(1+\text{count}\times\text{wdecay})$, where the initial learning rate lr_0 is set to 1.0, the weight decay 'wdecay' is set to 2×10^{-8}, and the parameter 'count' denotes the number of samples processed, so that the learning rate will decay with the training of model. The basic back-off 5-gram language model (KN5) is also trained with the modified Kneser-Ney smoothing algorithm.

3 Statistics of normalizing factors on *N*-best hypotheses

3.1 Review of N-best rescoring

The output from the first decoding pass is usually a multi-candidate form encoded as lattice or N-best list. Each path in lattice or N-best list is a candidate time-aligned transcript $W = w_1, w_2, \ldots, w_n$ of the speech utterance X. N-best list for its simplicity is widely used, and N-best rescoring in LVCSR is reviewed here.

Given the acoustic model Λ, the language model L, and a speech utterance X_i, N-best hypotheses from ASR's decoding are denoted as $H_i = W_{i1}, W_{i2}, \ldots, W_{iN}$, where the score of each hypothesis W_{ij} is computed as

$$g(X_i, W_{ij}|\Lambda, L) = \log\mathrm{P}(X_i|W_{ij}, \Lambda) + \alpha\cdot\log\mathrm{P}(W_{ij}|L) + n_{ij}\cdot\text{wdpenalt}, \tag{1}$$

where the first two items correspond to acoustic scores and language scores, respectively, and the last one denotes the word penalty that balances insertions and deletions. Also, α denotes a scaling factor for language scores and n_{ij} denotes the number of words in the hypothesis W_{ij}. The global score for each hypothesis in H_i is computed and reranked. The top hypothesis is selected as the output for evaluation. Generally, better performance is expected with more accurate models.

3.2 Normalizing factor for one word

Given a word sequence $\bar{s}$, denote the t-th word as w_t. The identity of word w_t is denoted as $\mathrm{q}(w_t) = y_i \in \mathbf{V}$, where the subscript i of y_i is the word index in the vocabulary $\mathbf{V}$. The structures of FNNLM and RNNLM are shown in Figures 1 and 2, respectively, where $\boldsymbol{W}_{ho} \in \Re^{|\mathbf{V}|\times H} = \left[\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_{|\mathbf{V}|}\right]^T$ is the prediction matrix and $\boldsymbol{\theta}_{\forall i} \in \Re^{H\times 1}$ corresponds to each output node.

The predicted probability of NNLM is computed as

$$\mathrm{P}(\mathrm{q}(w_t) = y_j|\boldsymbol{h}_t) = \frac{\exp(s_t)}{z_t} \quad \text{with} \quad s_t = \boldsymbol{\theta}_j^T\boldsymbol{h}_t \quad \text{and} \quad z_t = \sum_{i=1}^{|V|}\exp\left(\boldsymbol{\theta}_i^T\boldsymbol{h}_t\right), \tag{2}$$

where $\exp(s_t)$ and z_t respectively correspond to the unnormalized probability and the softmax-normalizing factor. Computing this factor z_t results in heavy computational burden for normalization.

We evaluated our well-trained FNNLM and RNNLM on the 100-best hypotheses generated from the Hub5'00-SWB set (1,812 utterances), containing 147,454 hypotheses and 2,125,315 words. The $\log(z_t)$ for each word is computed and the probability density functions (PDFs) of the $\log(z_t)$ for FNNLM and RNNLM are plotted and shown in Figure 3. It shows that the log normalizing factor is widely distributed, ranging from 13 to 20 for FNNLM and from 7 to 20 for RNNLM, respectively. It seems that the variance of $\log(z_t)$ is so large that the normalizing factor $\log(z_t)$ can't be simply approximated as a constant for N-best rescoring. However, several findings from our firsthand experience have been noticed to help us approximate the normalizing factor, and we also conclude that some discriminative information of NNLM exists in the unnormalized probability for N-best rescoring in the next two sub-sections.

3.3 Normalizing factor for one hypothesis

The output of speech recognizer is usually encoded as N-best hypotheses, and the better hypothesis can be selected via rescoring with more accurate models. The language score for the hypothesis $W_{ij} = w_{ij1}, w_{ij2}, \ldots, w_{ijn_{ij}}$ is computed as

$$\begin{aligned} \log \mathrm{P}(W_{ij}|L) &= \sum_{t=1}^{n_{ij}} \log(\mathrm{P}(w_{ijt}|\boldsymbol{h}_{ijt})) \\ &= \sum_{t=1}^{n_{ij}} s_{ijt} - \sum_{t=1}^{n_{ij}} \log(z_{ijt}), \end{aligned} \tag{3}$$

where n_{ij} denotes the number of words in hypothesis W_{ij}, s_{ijt} can be efficiently computed with the dot product of two vectors, while z_{ijt} requires a lot of computing consideration.

We randomly selected one utterance from Hub5'00-SWB set and decoded it with HDecode for recognition. Top ten hypotheses are shown in Table 1. We notice that there are lots of similar contexts in N-best hypotheses, especially for the hypotheses with a low word error rate (WER), and differences usually exist in local. As a matter of fact, the normalizing factor z_{ijt} is completely determined by the context via a smooth function in Equation 2, and similar contexts will result in similar normalizing factors close to each other in value. Thus, lots of normalizing factors in the N-best hypotheses are the same or similar as for lots of the same or similar contexts, so that we roughly approximate $\sum_{t=1}^{n_{ij}} \log(z_{ijt})$ for hypothesis W_{ij} as a constant proportional to n_{ij} in this case, shown as

$$\mu_{ij} = \frac{1}{n_{ij}} \sum_{t=1}^{n_{ij}} \log(z_{ijt}) \approx \mu_i, \tag{4}$$

where μ_i is the constant corresponding to utterance X_i and μ_i can be estimated as $\mu_i \approx \overline{\mu_{ij}} = \frac{1}{N} \sum_{j=1}^{N} \mu_{ij}$.

This approximation for utterance X_i can be evaluated with the variance of μ_{ij} as $\mathrm{Var}(\mu_{ij}) = \frac{1}{N} \sum_{j=1}^{N} (\mu_{ij} - \overline{\mu_{ij}})^2$, and $\overline{\mu_{ij}}$ denotes the mean of μ_{ij} in H_i. As a matter of fact, many utterances need to be approximated with Equation 4, and these approximations can be evaluated as mean of $\mathrm{Var}(\mu_{ij})$ and variance of $\mathrm{Var}(\mu_{ij})$ for all utterances. The statistically smaller the $\mathrm{Var}(\mu_{ij})$, the more accurate the approximations.

We evaluated the well-trained FNNLM and RNNLM on the 100-best hypotheses generated from Hub5'00-SWB

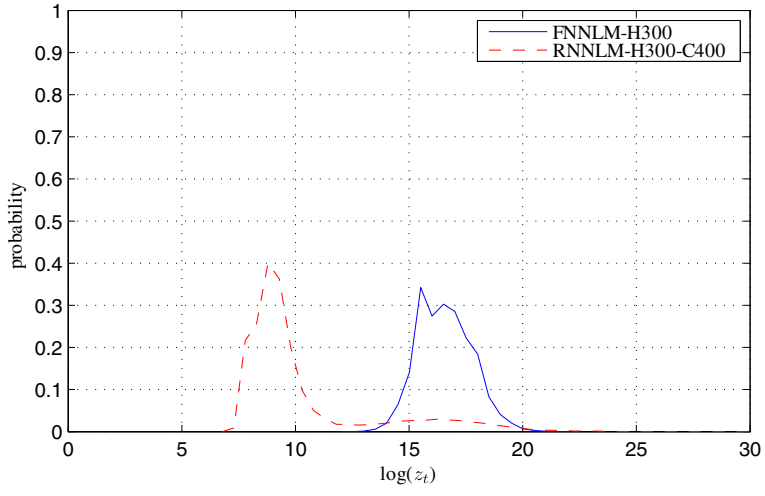

Figure 3 PDF curves of $\log(z_t)$ for each word on Hub5'00-SWB set.

Table 1 Ten hypotheses decoded from one utterance in Hub5'00-SWB set

	Hypotheses
1	No no it doesn't work on every issue
2	No no it doesn't work in every issue
3	No no it doesn't work and every issue
4	No no it doesn't work out every issue
5	Oh no no it doesn't work on every issue
6	No no it doesn't work then every issue
7	No no he doesn't work on every issue
8	Oh no no it doesn't work in every issue
9	On no no it doesn't work on every issue
10	No no he doesn't work in every issue

Lots of contexts are similar and differences usually exist in local.

set (1,812 utterances). The μ_{ij} for each hypothesis and the Var(μ_{ij}) for each 100-best list are computed, and the PDFs of Var(μ_{ij}) for FNNLM and RNNLM are shown in Figure 4. It shows that the PDFs are quite sharp and close to zero, just like an impulse function, and the constant approximation in Equation 4 for each utterance is accurate and reasonable to some extent.

3.4 Number of words in hypothesis

We also notice that the number of words for one hypothesis is similar with each other in the N-best list. As a matter of fact, N-best hypotheses are rescored and reranked according to the relative scores. If all the hypotheses for utterance X_i contain the same number of words, then the second item in Equation 3 for one hypothesis will be the same as that of others in the N-best list, based on Equation 4, shown as

$$\begin{aligned} & n_{ij}\mu_{ij} \approx n_{ik}\mu_{ik}, \quad \forall i,j,k \\ & \text{s.t.} \quad n_{ij} \approx n_{ik} \quad \text{and} \quad \mu_{ij} \approx \mu_i \approx \mu_{ik} \quad \forall i,j,k, \end{aligned} \tag{5}$$

That is to say, the normalizing factors for one hypothesis will not affect the ranking in the N-best rescoring, and μ_i for utterance X_i can be arbitrary. We further approximate the constant μ_i to a global constant, irrelevant with the utterance, shown as

$$\mu_{ij} \approx \mu_i \approx \mu, \tag{6}$$

where μ is the global constant and can be estimated as $\mu = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}\mu_{ij}$ on the validation set. M and N denote the number of utterances and the number of hypotheses for each utterance, respectively.

Please note that the approximations in Equation 5 depend on the assumption that the hypotheses for each utterance are equal in length. We count the number of words n_{ij} for each hypothesis W_{ij} and compute the variance of n_{ij} for each utterance X_i as

$$\text{Var}(n_{ij}) = \frac{1}{N}\sum_{j=1}^{N}(n_{ij} - \overline{n_{ij}})^2) \quad \text{with} \quad \overline{n_{ij}} = \frac{1}{N}\sum_{j=1}^{N} n_{ij} \tag{7}$$

The statistically smaller the Var(n_{ij}), the more accurate the approximation in Equations 5 and 6. The PDF of Var(n_{ij}) on the 100-best hypotheses generated from Hub5'00-SWB set is shown in Figure 5. It shows that the PDF of Var(n_{ij}) is sharp and most of the Var(n_{ij})

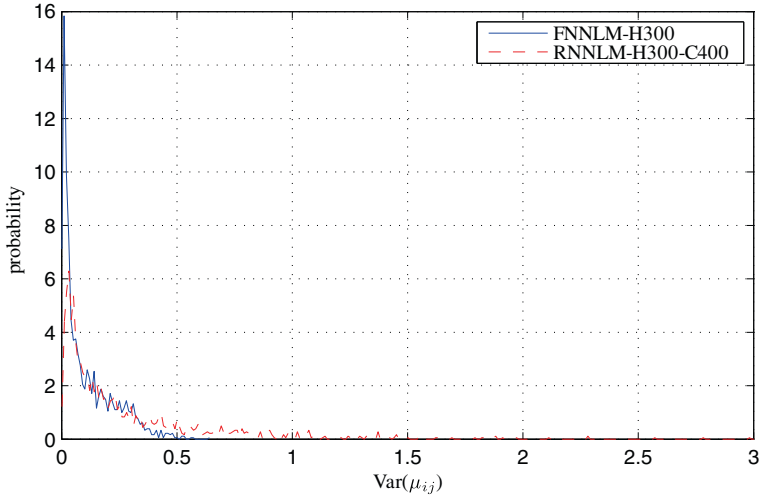

Figure 4 PDF curves of Var(μ_{ij}) for each *N*-best list on Hub5'00-SWB set.

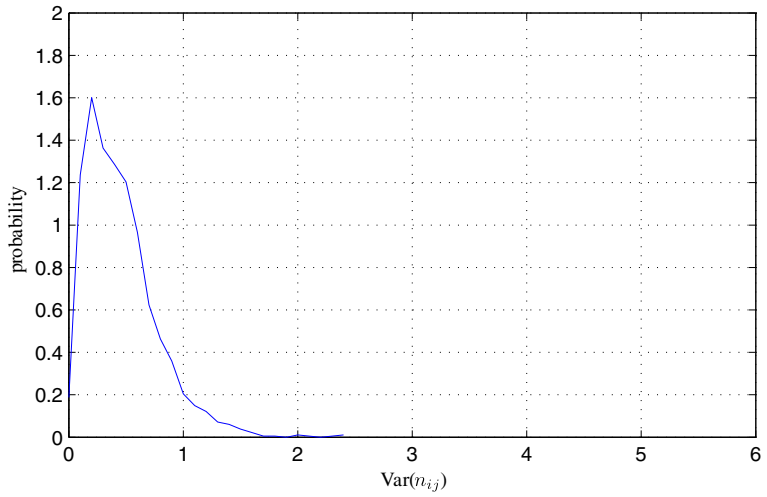

Figure 5 PDF curve of Var(n_{ij}) for each *N*-best list on Hub5'00-SWB set.

are smaller than 1.0. The difference of N-best hypotheses in length is small, and the approximation for N-best hypotheses in Equation 5 is reasonable to some extent.

3.5 Normalizing factor approximation

Based on the approximation in Equations 4, 5, and 6, the LM scores in Equation 3 can be simplified as

$$\log \mathrm{P}(W_{ij}|L) \approx \sum_{t=1}^{n_{ij}} s_{ijt} - n_{ij} \cdot \mu, \tag{8}$$

where only the first item needs to be estimated, while the second item can be estimated on validation set for rescoring. The complexity of the output layer is significantly reduced form O($|\mathbf{V}|H$) to O(H) with the constant approximation of the normalizing factor.

We also notice that the discriminative information of NNLM for N-best rescoring exists in the unnormalized probability in Equation 8 and the LM scores from back-off N-gram, especially for the KN3 in decoding, usually are available for rescoring. We will investigate the discriminative information in unnormalized NNLM (UP-NNLM), combined with back-off N-gram, to further improve the performance of speech recognizer in the next section.

4 Combining unnormalized NNLM and back-off *N*-gram

The UP-NNLM combined with back-off N-gram in the logarithmic domain is presented in detail. Generally, the performance of STT systems can be further improved with interpolation of NNLM and back-off N-gram. Since exact probability of NNLM is unavailable in Equation 8, the linear interpolation is performed in logarithmic domain for the entire hypothesis, shown as

$$\begin{aligned} &\log\tilde{P}(W_{ij}|L) \\ &= \lambda \cdot \log \mathrm{P}(W_{ij}|L) + (1-\lambda) \cdot \log \mathrm{P}_{\mathrm{Ngram}}(W_{ij}|L) \\ &= \lambda \cdot \sum_{t=1}^{n_{ij}} (s_{ijt} - \mu) + (1-\lambda) \cdot \log \mathrm{P}_{\mathrm{Ngram}}(W_{ij}|L), \end{aligned} \tag{9}$$

where $\mathrm{P}_{\mathrm{Ngram}}(W_{ij}|L)$ is the language score of back-off N-gram for hypothesis W_{ij}. By substituting Equation 9 into Equation 1, the global score for each hypothesis is computed as

$$\begin{aligned} &g(X_i, W_{ij}|\Lambda, L) \\ &= \alpha \cdot \lambda \cdot \sum_{t=1}^{n_{ij}} s_{ijt} + \alpha \cdot (1-\lambda) \cdot \log \mathrm{P}_{\mathrm{Ngram}}(W_{ij}|L) \\ &\quad + \log \mathrm{P}(X_i|W_{ij}, \Lambda) + n_{ij} \cdot \mathrm{wdpenalty}' \\ &\quad \text{with} \quad \mathrm{wdpenalty}' = \mathrm{wdpenalty} - \alpha \cdot \lambda \cdot \mu, \end{aligned} \tag{10}$$

where the normalizing factor is absorbed into the word penalty. The unnormalized probability only needs to be computed. The computational complexity of the output layer is reduced significantly without explicit normalization.

5 Complexity analysis and speed comparisons

The complexities of NNLM and UP-NNLM are analyzed, and the evaluation speeds of NNLM and UP-NNLM are also measured, shown in Table 2 for detailed comparisons.

The class-based output layer method was based on the frequency partition [4], and the computational complexity of the output layer is given as $O((C+|\mathbf{V}|/C)H)$, shown in Table 2. This method is usually used to speed up the training of NNLM while the evaluation of NNLM is also speeded up. Compared with the class-based method, the unnormalized probability of NNLM (UP-NNLM) in our

Table 2 Complexity and speed comparisons of UP-NNLMs and NNLMs for word predictions

Model	Complexity	Speed × 10^3 (words/second)
RNNLM[a]	$O(H^2 + \|\mathbf{V}\|H)$	0.041
+Class layer	$O(H^2 + (\|\mathbf{V}\|/C + C)H)$	4.21
UP-RNNLM	$O(H^2 + H)$	11.95
FNNLM[a]	$O((N-1)DH + \|\mathbf{V}\|H)$	0.214
+Class layer	$O((N-1)DH + (\|\mathbf{V}\|/C + C)H)$	9.55
UP-FNNLM	$O((N-1)DH + H)$	17.77
fast-UP-FNNLM	$O(H)$	*240.38*

[a]The implementations of RNNLM and FNNLM are based on the open source toolkits, CSLM and RNNLM. The matrix and vector operations in CSLM toolkit are optimized via MKL Library, so that the evaluation of FNNLM is faster than RNNLM with the same size of hidden layer.

method is required with the complexity $O(H)$ in the output layer. Especially, the complexity of the hidden layer in FNNLM can be further reduced by lookup in the position-dependent projection matrix $\widehat{\boldsymbol{E}}_k \in \Re^{|\mathbf{V}|\times H}, k = 1, 2, \cdots, N-1$, where the $\widehat{\boldsymbol{E}}_k = \boldsymbol{E}\boldsymbol{W}_{\text{ih}k}$ can be computed off-line. We denote the fast version of UP-FNNLM as fast-UP-FNNLM in Table 2.

The evaluation speed is measured by the number of words processed per second on a machine with an Intel(R) Xeon(R) 8-core CPU E5520 at 2.27 GHz and 8-G RAM, shown in Table 2. The implementations are based on the open source toolkits, CSLM [17] and RNNLM [18], to ensure the repeatability. To compare clearly, the speed of NNLM without a class layer is also measured. One million words are randomly selected from training data and evaluated by the FNNLM and NNLM, where the word is fed into the FNNLM and RNNLM one by one. Experimental results show that UP-FNNLM and UP-RNNLM are about 2∼3 times faster than 'FNNLM + class layer' and 'RNNLM + class layer' for evaluation. Note that the complexity of the hidden layer in UP-FNNLM or UP-RNNLM is comparable with that of class-based output layer in FNNLM + class layer or RNNLM + class layer, so that this speedup factor is reasonable. Also, it is worthy to notice that the fast-UP-FNNLM is more than 25 times faster than FNNLM + class layer and more than 1,100 times faster than FNNLM. To clearly show the speedup, the evaluation speed is also compared with different hidden layers, shown in Table 3. The larger the hidden layer, the slower the evaluation, and the 'fast-UP-FNNLM' is the fastest of all.

Table 3 Testing speed comparisons of UP-NNLMs and NNLMs for different hidden layers

Model	Speed × 10^3 (words/second)					
	10	50	100	200	300	400
RNNLM[a]	0.78	0.23	0.11	0.056	0.041	0.029
+Class layer	70.87	28.5	13.94	7.39	4.21	3.10
UP-RNNLM	400.01	197.75	80.30	26.93	11.95	7.20
FNNLM[a]	2.72	1.63	0.35	0.291	0.214	0.14
+Class layer	124.89	66.83	36.02	16.16	9.55	5.75
UP-FNNLM	678.48	209.16	83.95	25.33	17.77	11.63
fast-UP-FNNLM	*746.43*	*557.18*	*406.47*	*291.35*	*240.38*	*201.75*

[a]The implementations of RNNLM and FNNLM are based on the open source toolkits, CSLM and RNNLM. The matrix and vector operations in CSLM toolkit are optimized via MKL Library, so that the evaluation of FNNLM is faster than RNNLM with the same size of hidden layer.

6 *N*-best rescoring evaluation

The NNLM and UP-NNLM are applied to N-best rescoring to demonstrate the performance of our method in this section. According to our experimental setup described in Section 2, the perplexities of our trained language models, including KN3, KN5, FNNLM, and RNNLM-C400, are presented in Table 4 for comparisons, where KN3 is used for decoding. It shows that the RNNLM-C400 interpolated with KN5 performs best of all on the Hub5'00-SWB set and the RT03S-FSH set. Also, RNNLM-C400 performs slightly better than FNNLM with the same setup in perplexity. The Hub5'00-SWB set and the RT03S-FSH set act the validation set and evaluation set, respectively. The 100-best hypotheses for these two sets are rescored and reranked by different language models, shown in Table 5, where 1-best denotes the output of HDecode with KN3. The results for 1-best hypothesis on these two sets as our baseline are comparable with other reported results [20,21].

The UP-FNNLM and UP-RNNLM, combined with back-off N-gram, are used for fast rescoring in this section. Note that the output layer of our trained RNNLM-C400 is divided into many small softmax output layers in order to speed up the training on the large corpus. Thus, the unnormalized probability comes from the activations of the class layer and the specific softmax output layer, while the entire normalizing factor is also approximated as Equation 6. The UP-NNLM is linearly interpolated with KN5 in the logarithmic domain. The

Table 4 Perplexity of Hub5'00-SWB and RT03S-FSH set for different LMs

Model	Perplexity	
	Hub5'00-SWB	RT03S-FSH
KN3	89.40	66.76
KN5	86.78	63.80
+FNNLM	71.84	53.98
+RNNLM-C400	*66.67*	*51.34*
FNNLM	76.17	56.99
RNNLM-C400	70.83	54.39

Table 5 Word error rates (WERs) of Hub5'00-SWB and RT03S-FSH for 100-best/1,000-best rescoring with NNLM and UP-NNLM

Model	100-best WER (%)		1,000-best WER (%)	
	Hub5'00-SWB	RT03S-FSH	Hub5'00-SWB	RT03S-FSH
1-best (KN3)	17.3	20.2	17.3	20.2
+UP-FNNLM	16.6 (−0.7)	19.0 (−1.2)	16.5 (−0.8)	19.0 (−1.2)
+UP-RNNLM-C400	16.5 (−0.8)	19.0 (−1.2)	16.4 (−0.9)	19.1 (−1.1)
+FNNLM	15.7 (−1.6)	18.4 (−1.8)	15.6 (−1.7)	18.4 (−1.8)
+RNNLM-C400	15.4 (−1.9)	18.2 (−2.0)	15.2 (−2.1)	18.4 (−1.8)
KN5	17.1 (−0.2)	19.5 (−0.7)	16.9 (−0.4)	19.4 (−0.8)
+UP-FNNLM	16.4 (−0.9)	18.6 (−1.6)	16.2 (−1.1)	18.6 (−1.6)
+UP-RNNLM-C400	*16.1* (−1.2)	*18.5* (−1.7)	*16.0* (−1.3)	*18.5* (−1.7)
+FNNLM	15.6 (−1.7)	18.3 (−1.9)	15.4 (−1.9)	18.2 (−2.0)
+RNNLM-C400	*15.3* (−2.0)	*18.1* (−2.1)	*15.2* (−2.1)	*18.1* (−2.1)
FNNLM	15.9 (−1.4)	18.7 (−1.5)	15.8 (−1.5)	18.7 (−1.5)
RNNLM-C400	15.4 (−1.9)	18.4 (−1.8)	15.5 (−1.8)	18.6 (−1.6)

weight for interpolation, the scale of LM scores, and the word penalty are all individually tuned on Hub5'00-SWB set, and then the final performance is evaluated on RT03S-FSH set, shown in Table 5. Significant reductions in WER are observed on the validation and evaluation sets. The language scores of KN3 is usually available in the lattice or N-best list, so that the UP-RNNLM combined with the KN3 reduces WER by 0.8% and 1.2% absolute on Hub5'00-SWB and RT03S-FSH sets, respectively. 'KN5 + UP-RNNLM-C400' further reduces the WER by 1.2% and 1.7% absolute on these two sets. Also, we notice that UP-RNNLM performs slightly better than UP-FNNLM, while UP-FNNLM can be evaluated much faster than UP-RNNLM. It can be seen that the 'UP-NNLM + KN5' can obtain about 1/2 to 2/3 gains of 'NNLM + KN5' with little computation. Experimental results show that the unnormalized probability of NNLMs, including FNNLM and RNNLM, is quite complementary to that of back-off N-gram, and the performance is further improved via the combination of back-off N-gram and NNLM.

7 Discussions with related work

Fast rescoring with NNLM has attracted much attention in the field of speech recognition [9,10,22-24]. Many methods [9,10,22] for factorizing the output layer were proposed to reduce the complexity of NNLM and to speed up the training and the evaluation. Other techniques [22,24] were proposed to avoid redundant computations existing in N-best or lattice rescoring. Our proposed method can be easily combined with these methods to further improve the speed of rescoring. Also, a good work on fast training of NNLM with noise-contrastive estimation (NCE) [25] was proposed in [26], where the normalizing factor for each context was treated as a parameter to learn during the training. The training of NNLM was speeded up without the explicit normalization. As a matter of fact, the normalizing factor for each context needs to be learned separately, and these normalizing factors for different contexts will be different, so that the evaluation of NNLM needs to be normalized explicitly. Interestingly, we noticed that the normalizing factors can be manually fixed to one instead of learning them during the training of NNLM, as mentioned in [26]. We believe this findings will be helpful to further improve our current work, since if the variance of the normalizing factor could be constrained in a small range the approximation will be further improved in Equation 6. In this work, the distribution of normalizing factors on the N-best list is investigated, and the normalizing factor for each hypothesis is approximated as a constant for fast rescoring without considering the variance of the normalizing factor. Based on the findings mentioned in [26], we will investigate how to constrain the variance of normalizing factors during the training to further improve our method in the next work.

Table 6 Word error rates (WERs) of Hub5'00-SWB and RT03S-FSH for lattice rescoring with UP-FNNLM

Model	WER	
	Hub5'00-SWB	RT03S-FSH
1-best (KN3)	17.3	20.2
KN5 (100-best)	17.1	19.5
KN5 (1,000-best)	16.9	19.4
KN5 (lattice)	17.0	19.4
UP-FNNLM + KN5 (100-best)	16.4	18.6
UP-FNNLM + KN5 (1,000-best)	16.2	18.6
UP-FNNLM + KN5 (lattice)	*16.2*	*18.5*

Furthermore, an alternative method to speed up the rescoring is to use the word lattice instead of N-best list. The word lattice can compactly represent much more hypotheses than the N-best list, as the output of STT. We wonder whether our proposed method can be extended to lattice rescoring. As we all know, the N-best list is close to the lattice with the size of N-best list increased. Two experiments are designed to validate our method. On the one hand, we investigate whether the performance of N-best rescoring will be degraded with the size of N-best list increased. 1,000-best list instead of 100-best list is extracted for each utterance and rescored by our proposed method, shown in Table 5. Experimental results show that our proposed method still works well for 1,000-best list, and similar improvements are obtained for 1,000-best rescoring. On the other hand, we directly rescore the lattice with 'lattice-tool' [27] command to evaluate our proposed method. In consideration of the easy implementation and fast rescoring, the 'UP-FNNLM + KN5' is integrated into lattice-tool command, where the computation of the LM score is replaced with Equation 9 for convenience. The experimental results show that the rescoring of lattice obtains a slightly lower WER than that of N-best list in Table 6. All the results also mean that our proposed approximations based on our firsthand observations are reasonable and effective for fast N-best rescoring.

8 Conclusions

Based on the observed characteristics of N-best hypotheses, the normalizing factors of NNLM for each hypothesis are approximated as a global constant for fast evaluation. The unnormalized NNLM combined with back-off N-gram is empirically investigated and evaluated on the English-Switchboard speech-to-text task. The computation complexity is reduced significantly without explicit softmax normalization. Experimental results show that UP-NNLM is about 2∼3 times faster than 'NNLM + class layer' for evaluation. Moreover, the fast-UP-FNNLM is more than 25 times faster than FNNLM + class layer and more than 1,100 times faster than FNNLM. The N-best hypotheses from STT's output are approximately rescored and reranked by unnormalized NNLM combined with back-off N-gram model in the logarithmic domain. Experimental results show that the unnormalized probability of NNLM, including FNNLM and RNNLM, is quite complementary to that of back-off N-gram, and UP-NNLM is discriminative for N-best rescoring, even though UP-NNLM is not so accurate. The performance of STT system is improved significantly by 'KN5 + UP-NNLM' with little computational resource.

Endnotes

[a]HDecode -A -D -T 1 -s 12.0 -p -6.0 -n 32 -t 150.0 150.0 -v 105.0 95.0 -u 10000 -l lat/ -z lat -C CF -H models/HMM -w models/LM -S xa.scp -i xa.mlf models/DICT models/LIST (http://htk.eng.cam.ac.uk/extensions/).

[b]lattice-tool -nbest-decode 100 -read-htk -htk-logbase 2.718 -htk-lmscale 12.0 -htk-wdpenalty -6.0 -in-lattice-list xa.lst -out-nbest-dir nbest/ (http://www.speech.sri.com/projects/srilm/manpages/lattice-tool.1.html).

Abbreviations

ASR: automatic speech recognition; DNN: deep neural network; FNNLM: feed-forward neural network language model; KN: Kneser-Ney smoothing algorithm; KN3: back-off 3-gram based on KN smoothing; KN5: back-off 5-gram based on KN smoothing; LM: back-off *N*-gram language model; NNLM: neural network language model; PDF: probability density function; PPL: perplexity; RNNLM: recurrent neural network language model; STT: speech-to-text; WER: word error rate.

Competing interests

The authors declare that they have no competing interests.

Acknowledgements

The authors are grateful to the anonymous reviewers for their insightful and valuable comments. This work was supported by the National Natural Science Foundation of China under Grant Nos. 61273268, 61005019, and 90920302, and in part by the Beijing Natural Science Foundation Program under Grant No. KZ201110005005.

Received: 7 December 2013 Accepted: 4 April 2014
Published: 28 April 2014

References

1. Y Bengio, R Ducharme, P Vincent, C Jauvin, A neural probabilistic language model. Mach. Learn. Res. (JMLR), 1137–1155 (2003)
2. E Arisoy, TN Sainath, B Kingsbury, B Ramabhadran, Deep neural network language models, in *Proceedings of NAACL-HLT Workshop* (Montreal, 2012), pp. 20–28. http://www.aclweb.org/anthology/W12-2703
3. T Mikolov, M Karafiat, L Burget, JH Cernocky, S Khudanpur, Recurrent neural network based language model, in *Proceedings of InterSpeech* (Makuhari, 2010), pp. 1045–1048
4. T Mikolov, S Kombrink, L Burget, JH Cernocky, S Khudanpur, Extensions of recurrent neural network language model, in *Proceedings of ICASSP* (Prague, 2011)
5. M Sundermeyer, R Schluter, H Ney, LSTM neural networks for language modeling, in *Proceedings of InterSpeech* (Portland, 2012)
6. T Mikolov, A Deoras, S Kombrink, L Burget, JH Cernocky, Empirical evaluation and combination of advanced language modeling techniques, in *Proceedings of InterSpeech* (Florence, 2011)
7. S Kombrink, T Mikolov, M Karafiat, L Burget, Recurrent neural network based language modeling in meeting recognition, in *Proceedings of InterSpeech* (Florence, 2011)
8. T Mikolov, Statistical language models based on neural networks. PhD thesis, Brno University of Technology (BUT), 2012. http://www.fit.vutbr.cz/imikolov/rnnlm/thesis.pdf.
9. HS Le, I Oparin, A Allauzen, JL Gauvain, F Yvon, Structured output layer neural network language models for speech recognition. IEEE Trans. Audio Speech Lang. Process. **21**, 197–206 (2013)
10. Y Shi, WQ Zhang, J Liu, MT Johnson, RNN language model with word clustering and class-based output layer. EURASIP J. Audio Speech Music Process. **22** (2013). doi:10.1186/1687-4722-2013-22
11. F Morin, Y Bengio, Hierarchical probabilistic neural network language model, in *Proceedings of AISTATS* (Barbados, 2005), pp. 246–252
12. A Mnih, G Hinton, A scalable hierarchical distributed language model. Adv. Neural Inf. Process. Syst. **21**, 1081–1088 (2008)
13. C Fellbaum, *WordNet: an Electronic Lexical Database* (MIT, Cambridge, 1998)
14. PF Brown, PV deSouza, RL Mercer, VJD Pietra, JC Lai, Class-based N-gram models for natural language. Comput. Linguist. **18**(4), 467–479 (1992)
15. J Godfrey, E Holliman, *Switchboard-1 Release 2* (Linguistic Data Consortium, Philadelphia, 1997)

16. The CMU pronouncing dictionary release 0.7a (2007). http://www.speech.cs.cmu.edu/cgi-bin/cmudict.
17. H Schwenk, CSLM: Continuous space language model toolkit (2010). http://www-lium.univ-lemans.fr/cslm/
18. T Mikolov, A Deoras, S Kombrink, L Burget, JH Cernocky, RNNLM - Recurrent Neural Network Language Modeling Toolkit, in *Proceedings of ASRU* (Hawaii, 2011). http://www.fit.vutbr.cz/imikolov/rnnlm/
19. DE Rumelhart, GE Hinton, RJ Williams, Learning representations by back-propagating errors. Nature **323**, 533–536 (1986)
20. F Seide, G Li, X Chen, D Yu, Feature engineering in context-dependent deep neural networks for conversational speech transcription, in *Proceedings of ASRU* (Hawaii, 2011)
21. M Cai, Y Shi, J Liu, Deep maxout neural networks for speech recognition, in *Proceedings of ASRU* (Olomouc, 2013)
22. H Schwenk, Continuous space language models. Comput. Speech Lang. **21**(3), 592–518 (2007)
23. M Auli, M Galley, C Quirk, G Zweig, Joint language and translation modeling with recurrent neural networks, in *Proceedings of EMNLP* (Seattle, 2013), pp. 1044–1054
24. Y Si, Q Zhang, T Li, J Pan, Y Yan, Prefix tree based n-best list re-scoring for recurrent neural network language model used in speech recognition system, in *Proceedings of InterSpeech* (Lyon, 2013), pp. 3419–3423
25. M Gutmann, A Hyvarinen, Noise-contrastive estimation: a new estimation principle for unnormalized statistical models, in *Proc. of AISTATS* (Sardinia, 2010), pp. 297–304
26. A Mnih, YW Teh, A fast and simple algorithm for training neural probabilistic language models, in *Proceedings of ICML* (Edinburgh, 2012)
27. A Stolcke, SRILM – an extensible language modeling toolkit, in *Proceedings of ICSLP*, (2002), pp. 901–904

doi:10.1186/1687-4722-2014-19
Cite this article as: Shi *et al.*: Empirically combining unnormalized NNLM and back-off *N*-gram for fast *N*-best rescoring in speech recognition. *EURASIP Journal on Audio, Speech, and Music Processing* 2014 **2014**:19.