LATTICE RESCORING BASED ON LARGE ENSEMBLE OF COMPLEMENTARY NEURAL LANGUAGE MODELS

Atsunori Ogawa, Naohiro Tawara, Marc Delcroix, and Shoko Araki

NTT Corporation, Japan

ABSTRACT

We investigate the effectiveness of using a large ensemble of advanced neural language models (NLMs) for lattice rescoring on automatic speech recognition (ASR) hypotheses. Previous studies have reported the effectiveness of combining a small number of NLMs. In contrast, in this study, we combine up to eight NLMs, i.e., forward/backward long short-term memory/Transformer-LMs that are trained with two different random initialization seeds. We combine these NLMs through iterative lattice generation. Since these NLMs work complementarily with each other, by combining them one by one at each rescoring iteration, language scores attached to given lattice arcs can be gradually refined. Consequently, errors of the ASR hypotheses can be gradually reduced. We also investigate the effectiveness of carrying over contextual information (previous rescoring results) across a lattice sequence of a long speech such as a lecture speech. In experiments using a lecture speech corpus, by combining the eight NLMs and using context carry-over, we obtained a 24.4% relative word error rate reduction from the ASR 1-best baseline. For further comparison, we performed simultaneous (i.e., non-iterative) NLM combination and 100-best rescoring using the large ensemble of NLMs, which confirmed the advantage of lattice rescoring with iterative NLM combination.

Index Terms— Lattice rescoring, complementary neural language models, large ensemble, iterative lattice generation, context carry-over

1. INTRODUCTION

Based on the recent introduction of state-of-the-art neural network (NN) modeling, the performance of automatic speech recognition (ASR) has been greatly improved [1, 2], and various types of ASR-based applications, including voice search services and smart speakers, have been actively developed. Despite this great progress, in some situations such as performing ASR in noisy environments or performing ASR for conversational speech, the accuracy of ASR remains at an unsatisfactory level [3–8].

A promising approach for reducing ASR errors in such severe situations involves the use of multiple ASR hypotheses (word sequences), which are represented in such forms as an N-best list or a lattice. This is because an ASR hypothesis that has a significantly lower word error rate (WER) than the 1-best hypothesis can be found in multiple hypotheses if it is appropriately rescored (reranked). Various types of rescoring methods have been developed and applied to noisy or conversational speech recognition [3–6, 9–22].

In these rescoring methods, advanced neural language models (NLMs) are used as rescoring models. They can accurately model much longer word sequences than can conventional count-based n-gram LMs [23, 24], which can model sequences of only n words (where n is typically three to five). These NLMs are used to refine language scores attached to ASR hypotheses that are calculated using the n-gram LMs. Among the NLMs, long short-term memory (LSTM)-based recurrent NLMs [25] are currently the most widely used model. A forward LSTMLM can provide good WER reduction, but the WER can be further reduced by additionally using another model. Such a model would be, for example, a forward LSTMLM that has a different model structure [12, 13] or that is trained with a different setting (e.g., a different initialization seed or a different data shuffling scheme) [5, 17] or a backward LSTMLM that is trained by using a reversed text dataset [5, 6, 10, 14, 17], since these models work complementarily with each other. In addition to the LSTMLMs, NLMs based on Transformers [26] have recently been used for rescoring. They have a non-recurrent self-attentive architecture that is completely different from that of the LSTMLMs, and they show comparable or superior rescoring performance to the LSTMLMs [19, 21, 22].

By performing ASR for a long speech such as a lecture speech, a long ASR hypothesis sequence can be obtained. In such a long speech (a series of utterances), the content of an utterance is naturally influenced by the content of previous utterances (i.e., context). Therefore, in rescoring such a long ASR hypothesis sequence, it is reasonable to use the rescoring results of the previous hypotheses as contextual information for rescoring the current hypothesis. It has been reported that, by carrying over contextual information across ASR hypotheses, the rescoring performance for such a long ASR hypothesis sequence can be improved [5, 6, 16, 18, 20–22].

In this study, we investigate the effectiveness of using a large ensemble of NLMs on lattice rescoring. As described above, previous studies [3, 4, 12–14] have reported the effectiveness of combining a small number of NLMs (up to four [13]) on lattice rescoring. In contrast, we combine up to eight NLMs, i.e., forward/backward LSTM/Transformer-LMs, which are trained with two different random initialization seeds. We combine these complementary NLMs through iterative lattice generation while introducing context carry-over (Section 2). We conducted experiments including experimental settings that have not been investigated in previous studies (Section 3) and confirmed the effectiveness of using a large ensemble of NLMs for lattice rescoring (Section 4). Our main findings can be summarized as follows.

(1) Combining six or seven NLMs can improve the performance of lattice rescoring.

(2) Lattice rescoring has an advantage over N-best rescoring when using a large ensemble of NLMs.

(3) Performing context carry-over in the backward direction is as effective as performing it in the forward direction.

(4) Iterative NLM combination has the potential to outperform simultaneous NLM combination, especially in a fast lattice rescoring setting.

2. LATTICE RESCORING METHOD

We introduce a method for combining NLMs through iterative lattice generation and a method for carrying over contextual information across lattices.

2.1. Combining NLMs through iterative lattice generation

A lattice is an efficient ASR result form of an input utterance that includes multiple ASR hypothesis candidates of the utterance. A lattice consists of nodes and arcs, where a node corresponds to a word boundary while an arc corresponds to a recognized word. An arc has an acoustic score and a language score, which are calculated during the ASR first-pass decoding. Language scores are usually calculated using a count-based n-gram LM.

We use the push-forward algorithm [27–30] for lattice rescoring. Given a lattice for an input utterance, we perform search on the lattice from its begin node to refine the language scores attached to the arcs using a rescoring model. Then, by tracing back the rescored lattice from its end node, we can obtain the final ASR hypothesis, i.e., the best word (arc) sequence, that shows the highest score.

We focus on the search processing at a lattice arc as shown in Fig. 1. Let $w_{1:t-1}$ be a partial word (arc) sequence (hypothesis) of length $t-1$. It is extended from the lattice begin node and its current score (log-likelihood) is $\log p(w_{1:t-1})$. It reaches the arc w_t, which has an acoustic score (log-likelihood) $\log p_{\texttt{acou}}(w_t)$ and a language score (log probability) $\log P_{\texttt{lang}}(w_t)$. By extending the partial hypothesis $w_{1:t-1}$ to this arc, the score of the extended partial hypothesis $w_{1:t}$ can be obtained as,

$$\log p(w_{1:t}) = \log p(w_{1:t-1}) + \log p_{\texttt{acou}}(w_t) + \alpha \left\{ \underline{(1-\beta)\log P_{\texttt{lang}}(w_t) + \beta \log P_{\texttt{resc}}(w_t \mid w_{1:t-1})} \right\}, \quad (1)$$

where $\log P_{\texttt{resc}}(w_t|w_{1:t-1})$ is the language score of w_t given $w_{1:t-1}$ calculated using a rescoring model (an NLM), β $(0<\beta<1)$ is the interpolation weight between the original language score and that calculated using the rescoring model, and α $(\alpha > 0)$ is the weight of the language score against the acoustic score. The underlined term in Eq. (1) corresponds to the refined language score attached to the arc w_t. By performing this search processing at all the arcs in the given lattice, we can generate a rescored lattice from the original lattice. Depending on the hyperparameters of search, the structure of the rescored lattice can change from that of the original lattice [28–31]. The hyperparameters include, for example, the n of n-gram approximation for merging hypotheses at a node (merging hypotheses that have the same history of n or more words into a single hypothesis) and the maximum number of hypotheses (k) stored at a node.

In the lattice generation described above, we performed processing in the forward direction using a forward NLM as the rescoring model. As described in Section 1, it has been reported that, by combining a few more NLMs that have complementarity with the above forward NLM, we can obtain a steady WER reduction [3, 4, 12–14]. In this study, to further reduce the WER, we repeat lattice generation for more iterations (up to eight iterations in our experiments as described in Section 4) by changing the NLMs, which are complementary with each other, at each iteration as shown in Fig. 1 ($I = 8$).

To perform iterative lattice generation (language score refinement), we need to design a way to interpolate the $(i-1)$th language score attached to a lattice arc and the ith language score calculated using the NLM (see Eq. (1) and Fig. 1). In this study, we assume that all the NLMs contribute equally to refining language scores (actually, in our experiments, they show similar dev/eval data perplexities as shown in Table 3) and define the ith interpolation weight as,

$$\beta(i) = \frac{1}{1+i}. \quad (2)$$

With this definition, at the ith iteration, language scores calculated using the first to ith NLMs can be equally combined at a lattice arc.

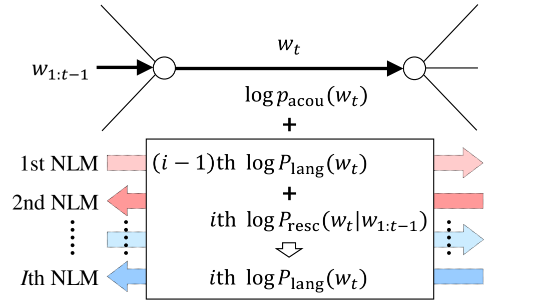

Fig. 1. Iterative lattice generation (language score refinement) at a lattice arc using I complementary NLMs.

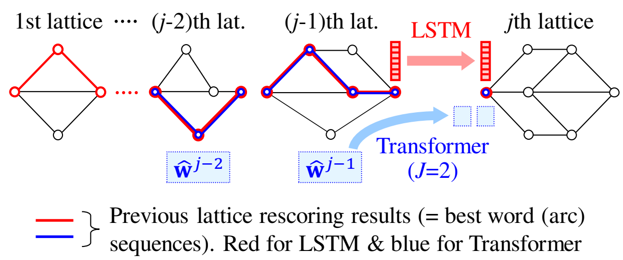

Fig. 2. Methods for carrying over contextual information (previous rescoring results) across lattices (utterances) with NLMs.

Following this definition, we combine the n-gram LM score attached to an arc in the original lattice (the 0th language score) equally with the other NLM scores. This is because an n-gram LM focuses on modeling local contexts, in contrast to the NLMs, and thus it works complementarily with the NLMs [3–6,9–22]. We can obtain the final ASR hypothesis by tracing back the Ith rescored lattice.

2.2. Carrying over contextual information across lattices

As described in Section 1, in rescoring an ASR lattice sequence of a long speech, we can exploit the previous lattice rescoring results to rescore the current lattice [5, 6, 16, 18, 20–22]. In this case, we need to develop a method to carry over the previous rescoring results (contextual information) to the begin node of the current lattice.

When we use an LSTMLM as the rescoring model, this context carry-over can be easily implemented. Thanks to its recurrent model architecture, the LSTMLM can encode rescoring results for all the previous lattices (utterances) in a single hidden state vector. As shown in Fig. 2, we can copy the single hidden state vector from the end node of the last ($(j-1)$th) lattice to the begin node of the current (jth) lattice and use it as the initial hidden state vector to start rescoring on the current lattice. Note that, when we do not perform context carry-over, we use the zero vector as the initial hidden state vector at the begin node of every lattice.

In contrast to the LSTMLM, when we use a Transformer-LM, we need to store a sequence of hidden state vectors to represent the previous rescoring results, since the Transformer-LM has a non-recurrent self-attentive model architecture [26]. This means that it linearly increases memory usage as the rescoring results get longer [21]. Therefore, in this study, as shown in Fig. 2, we keep only the last J rescoring results, i.e., $\hat{\mathbf{w}}^{j-J}, \cdots, \hat{\mathbf{w}}^{j-1}$, for the last J lattices and use them as contextual information to rescore the current (jth) lattice. In the above methods, we perform context carry-over in the forward direction (from the first utterance to the last utterance in a given long speech) using a forward NLM, however, similarly, we can also perform it in the backward direction (from the last utterance to the first utterance) using a backward NLM.

2.3. Other rescoring methods

We introduced a method for combining NLMs through iterative lattice generation in Section 2.1. However, we do not necessarily need to combine the NLMs iteratively. In fact, we can combine the NLMs simultaneously with, e.g., lattice combination [32, 33]. In this case, we individually perform lattice rescoring using each of the NLMs with the n-gram LM scores attached to arcs in the original lattice, and then we combine the rescored lattices with equal weight (since, as described in Section 2.1, we assume that all the NLMs contribute equally to refining language scores) while solving the differences in the rescored lattice structures. We experimentally compare these two NLM combination methods in Section 4.

N-best rescoring [5, 6, 10, 16–18, 20, 22] is another widely used ASR hypothesis rescoring method. As with lattice rescoring, we can perform N-best rescoring using a large ensemble of NLMs. We experimentally compare these two rescoring methods in Section 4. Note that, in contrast to lattice rescoring, with N-best rescoring, the iterative and simultaneous NLM combination methods provide the same rescoring results.

3. RELATION TO PRIOR WORK

As described in Section 1, rescoring is a promising approach for reducing ASR errors and many good studies have been conducted on rescoring techniques [3–6, 9–22]. Many studies have reported the effectiveness of combining complementary NLMs, but they investigated combinations of only a few NLMs [3–6, 10, 12–14, 17, 22]. In contrast, we combine up to eight NLMs. The previous studies did not use a backward Transformer-LM, but we use it as one of the NLMs. The combination of an LSTMLM and a Transformer-LM was investigated only in [22] with N-best rescoring. We investigate their combination with lattice rescoring.

The effectiveness of context carry-over has also been reported in many studies, but they performed it only in the forward direction [5, 6, 16, 18, 20–22]. In contrast, we perform context carry-over in both forward and backward directions (in [5, 6], the authors claim that they performed context carry-over in both directions with N-best rescoring, but they do not report the effectiveness of performing it in the backward direction).

4. EXPERIMENTS

To confirm the effectiveness of using a large ensemble of complementary NLMs on lattice rescoring, we conducted experiments using the corpus of spontaneous Japanese (CSJ) [34], which is a large-scale lecture speech corpus. We performed ASR using the Kaldi hybrid ASR system [35] and trained the NLMs using PyTorch [36, 37].

4.1. Experimental settings

Details of the CSJ training, development, and evaluation datasets are shown in Table 1 (the original Kaldi CSJ recipe has three evaluation datasets, but we merged them for simplicity). Using the training data, we trained a time delay NN-based acoustic model [38] and a trigram LM [23, 24]. The vocabulary size was set at 44k (words that appear only one time in the training data were mapped to the unknown word). Using the acoustic model and the trigram LM, we performed one-pass decoding [39] for the dev/eval data and obtained lattices for all the dev/eval utterances.

Using the training data and the PyTorch NLM training tool [37], we trained forward/backward LSTM/Transformer-LMs having the structures shown in Table 2. The final models (training epochs) were selected based on their perplexities for the development data. For the backward versions of the NLMs, we used the reversed training data. We trained two versions of each NLM by changing the random seed (i.e., 1 or 2) for parameter initialization, i.e., we trained eight NLMs in total. Perplexities obtained with the trigram LM and the four seed 1 NLMs for the dev/eval data are shown in Table 3 (the seed 2 NLMs show similar perplexities as those of the seed 1 NLMs).

Table 1. Details of the CSJ train/dev/eval datasets.

	Hours	#lecs	#utts	#words	OOV rate
Train	516	3176	403k	7.7M	0.37%
Dev	6.5	39	4000	9.6k	1.00%
Eval	5.1	30	3949	7.4k	0.86%

Table 2. Structures of an LSTMLM and a Transformer-LM.

	LSTM	Transformer
Embedding dimensions	1000	256
Positional encoding	—	Sinusoidal
Number of heads	—	8
Number of hidden nodes	1000	2000
Number of layers	2	8
Softmax (vocabulary) size	43720	43720

The training tool [37] concatenates all the training sentences and then makes a batch data filled with the batch size times the backpropagation through time (BPTT) length of words by splitting the concatenated long sentence. This batch-making strategy aims to avoid zero-padding and maximize GPU usage [21]. With this strategy, context carry-over is performed naturally in NLM training. Furthermore, the sizes of the trained NLMs are large enough, i.e., NLMs with larger sizes started to overfit the training data.

Using the trained NLMs (up to eight NLMs), we performed rescoring on the lattices of all the dev/eval utterances through iterative lattice generation (Section 2.1) with and without performing context carry-over (Section 2.2). From the results of preliminary experiments, when we performed context carry-over with Transformer-LMs, we set the context length J at 1 (see Fig. 2), since we could not obtain any further performance improvement by setting $J \geq 2$. We applied 5-gram approximation for merging hypotheses at a lattice node and set the maximum number of hypotheses (k) stored at a node at 10. With this search setting, the structure of a generated lattice at an iteration can change from that of the lattice at the previous iteration [28–31]. For further comparison, we also performed lattice combination [32, 33] and 100-best rescoring (Section 2.3). The 100-best lists were extracted from the lattices. Hereafter, we refer to the forward/backward LSTMLMs trained with seed x as LFx and LBx, and similarly, we refer to the Transformer-LMs as TFx and TBx.

4.2. Effects of forward/backward NLMs and context carry-over

Table 4 shows lattice rescoring results obtained with LF1, LB1, TF1, and TB1. First, we can confirm that they (models 1 to 4) steadily reduce the WERs from the ASR 1-best (trigram LM) baseline. The LSTMLMs show slightly better performance than the Transformer-LMs. Second, we can confirm that, by using contextual information (models 5 to 8), the WERs can be further reduced. We can confirm the effect of using contextual information not only in the forward direction [5, 6, 16, 18, 20–22] but also in the backward direction. This effect is larger for the LSTMLMs than the Transformer-LMs. This is because, in contrast to the Transformer-LMs that can use only a limited length of context, the LSTMLMs can use the whole length of context as described in Section 2.2.

Third, we can confirm that, by combining the forward and backward NLMs iteratively (models 9 and 10), the WERs can be greatly reduced compared to the case of using them individually (models

Table 3. Dev/Eval data perplexities obtained with the 3g LM and the forward/backward LSTM/Transformer-LMs trained with seed 1.

Data \ Model	3-gram	LF1	LB1	TF1	TB1
Dev	71.2	31.6	31.2	30.3	29.2
Eval	70.3	34.8	34.2	33.1	32.1

Table 4. Lattice rescoring results in WER [%] obtained with the forward/backward LSTM/Transformer-LMs trained with seed 1. Asterisks $*$ indicate the experimental settings that have not been investigated in the previous studies (Section 3).

No.	Model	Context	Dev	Eval
0.	ASR 1-best (3g)	No	7.7	9.0
1.	LF1	No	6.5	7.6
2.	LB1	No	6.5	7.6
3.	TF1	No	6.6	7.7
4.	TB1*	No	6.6	7.9
5.	LF1	Yes	6.2	7.3
6.	LB1*	Yes	6.2	7.4
7.	TF1	Yes	6.5	7.6
8.	TB1*	Yes	6.6	7.8
9.	LF1 $\rightarrow$ LB1	No	6.3	7.3
10.	TF1 $\rightarrow$ TB1*	No	6.4	7.4
11.	LF1 $\rightarrow$ LB1*	Yes	**5.9**	**7.0**
12.	TF1 $\rightarrow$ TB1*	Yes	6.3	7.3

1 to 4). This is the effect of combining the complementary NLMs [5, 6, 10, 14, 17]. The effect of using both the forward and backward NLMs is slightly larger for the Transformer-LMs than for the LSTMLMs. Finally, we can confirm that, by combining the two LSTMLMs using contextual information (model 11), the WERs can be further reduced. This result indicates the complementarity of combining NLMs and using contextual information. In contrast, with the two Transformer-LMs (model 12), the effect is small. We need to further investigate a method for effectively carrying over contextual information with the Transformer-LMs [21, 22].

4.3. Effects of combining up to eight NLMs

We combined up to the eight NLMs iteratively in the order of LF1 $\rightarrow$LB1$\rightarrow$TF1$\rightarrow$TB1$\rightarrow$LF2$\rightarrow$LB2$\rightarrow$TF2$\rightarrow$TB2 with the procedure described in Section 2.1. We also performed context carry-over. With this order, we aimed to first reduce the WERs largely by using LF1$\rightarrow$LB1 (models 9 and 11 in Table 4) and then to further reduce the WERs using TF1$\rightarrow$TB1 (models 10 and 12), which are complementary with LF1$\rightarrow$LB1.

Figure 3 shows experimental results for the evaluation data. We can confirm that, thanks to the complementarity of the eight NLMs, the WERs can be gradually reduced. Even at the later iterations (e.g., at the sixth and seventh iterations), the WERs can be reduced. We can confirm again the effect of using contextual information. We finally obtained a 6.8% WER, which corresponds to a 24.4% relative WER reduction from the ASR 1-best baseline of 9.0% WER. We also investigated other NLM combination orders (e.g., TF1$\rightarrow$TB1 $\rightarrow$LF1$\rightarrow$LB1$\rightarrow$TF2$\rightarrow$TB2$\rightarrow$LF2$\rightarrow$LB2), but the current order still performed slightly better than these other orders.

4.4. Comparison with other rescoring methods

We combined the eight NLMs simultaneously with lattice combination [32, 33]. With the rich (but slow) search setting described in Section 4.1 (i.e., 5-gram approximation with $k = 10$), we obtained the same WERs as those of the above-described iterative NLM combination as shown in Table 5. In contrast, the iterative combination

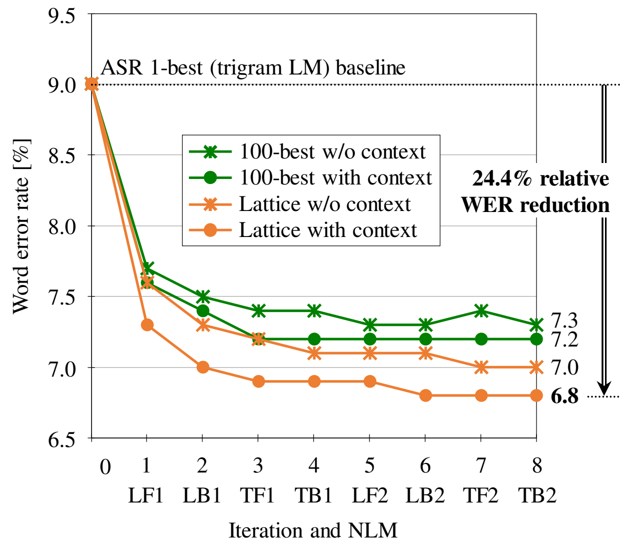

Fig. 3. Lattice/100-best rescoring results for the evaluation data obtained by combining up to the eight NLMs.

Table 5. Comparison of the iterative and simultaneous NLM combinations with the rich and fast search settings for the evaluation data.

	Iterative		Simul.	
Search setting \ Context	No	Yes	No	Yes
Rich (5-gram approx., $k = 10$)	7.0	6.8	7.0	6.8
Fast (0-gram approx., $k = 1$)	**7.3**	**7.0**	7.5	7.1

shows slightly lower WERs compared with the simultaneous combination when we perform the fast search, i.e., 0-gram approximation with $k = 1$ (in this setting, all hypotheses reaching a lattice node are merged, and thus the lattice structures are kept throughout rescoring processing [28–31]). From this result, we can confirm that the iterative (gradual) language score refinement (Section 2.1) would have an advantage in achieving stable rescoring with the fast (but unstable) search setting over the language score refinement that is always performed with the n-gram LM scores (Section 2.3).

We also performed 100-best rescoring by combining up to the eight NLMs iteratively with the order described above. The search space of the 100-best lists is greatly limited compared with that of lattices. Consequently, as shown in Fig. 3, with 100-best rescoring, the WER reduction tends to saturate at the earlier iterations (e.g., the third iteration when using the contextual information). As a result, the best WERs achieved with 100-best rescoring remain higher than those obtained with lattice rescoring. From these comparison results, we can confirm the advantage of lattice rescoring over N-best rescoring when using a large ensemble of NLMs.

5. CONCLUSION AND FUTURE WORK

We experimentally confirmed the effectiveness of using a large ensemble of complementary NLMs on lattice rescoring. The experimental results and findings obtained through this study are very informative because we conducted a variety of experiments including experimental settings that have not been investigated in the previous studies. Future work will include the use of more advanced NLMs [20, 22], an investigation into a method for effectively weighting the NLMs in the NLM combination [40, 41], and comparison/combination with system combination [42].

6. REFERENCES

[1] G. Hinton *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, Nov. 2012.

[2] D. Yu and L. Deng, *Automatic speech recognition: A deep learning approach*. Springer-Verlag, London, 2015.

[3] G. Saon, T. Sercu, S. Rennie, and H.-K. J. Kuo, "The IBM 2016 English conversational telephone speech recognition system," in *Proc. Interspeech*, 2016, pp. 7–11.

[4] G. Saon *et al.*, "English conversational telephone speech recognition by humans and machines," in *Proc. Interspeech*, 2017, pp. 132–136.

[5] W. Xiong *et al.*, "Achieving human parity in conversational speech recognition," *arXiv:1610.05256v2 [cs.CL]*.

[6] W. Xiong, L. Wu, F. Alleva, J. Droppo, X. Huang, and A. Stolke, "The Microsoft 2017 conversational speech recognition system," in *Proc. ICASSP*, 2018, pp. 5934–5938.

[7] J. Barker, S. Watanabe, E. Vincent, and J. Trmal, "The fifth 'CHiME' speech separation and recognition challenge: Dataset, task and baselines," in *Proc. Interspeech*, 2018, pp. 1561–1565.

[8] S. Watanabe *et al.*, "CHiME-6 challenge: Tackling multispeaker speech recognition for unsegmented recordings," in *Proc. of The 6th Intl. Workshop on Speech Processing in Everyday Environments (CHiME 2020)*, 2020.

[9] J. Du *et al.*, "The USTC-iFlytek systems for CHiME-5 challenge," in *Proc. of The 5th Intl. Workshop on Speech Processing in Everyday Environments (CHiME 2018)*, 2018.

[10] N. Kanda *et al.*, "The Hitachi/JHU CHiME-5 system: Advances in speech recognition for everyday home environments using multiple microphone arrays," in *Proc. of The 5th Intl. Workshop on Speech Processing in Everyday Environments (CHiME 2018)*, 2018.

[11] Z. Zhao, J. Wu, and L. Xie, "The NWPU system for CHiME-5 challenge," in *Proc. of The 5th Intl. Workshop on Speech Processing in Everyday Environments (CHiME 2018)*, 2018.

[12] C. Li and T. Wang, "The ZTSpeech system for CHiME-5 challenge: A far-field speech recognition system with front-end and robust back-end," in *Proc. of The 5th Intl. Workshop on Speech Processing in Everyday Environments (CHiME 2018)*, 2018.

[13] I. Medennikov *et al.*, "The STC system for the CHiME-6 challenge," in *Proc. of The 6th Intl. Workshop on Speech Processing in Everyday Environments (CHiME 2020)*, 2020.

[14] A. Arora *et al.*, "The JHU multi-microphone multi-speaker ASR system for the CHiME-6 challenge," in *Proc. of The 6th Intl. Workshop on Speech Processing in Everyday Environments (CHiME 2020)*, 2020.

[15] C. Zorilă, M. Li, D. Hayakawa, M. Liu, N. Ding, and R. Doddipatla, "Toshiba's speech recognition system for the CHiME 2020 challenge," in *Proc. of The 6th Intl. Workshop on Speech Processing in Everyday Environments (CHiME 2020)*, 2020.

[16] K. Žmolíková *et al.*, "BUT system for CHiME-6 challenge," in *Proc. of The 6th Intl. Workshop on Speech Processing in Everyday Environments (CHiME 2020)*, 2020.

[17] K. Irie, Z. Lei, L. Deng, R. Schlüter, and H. Ney, "Investigation on estimation of sentence probability by combining forward, backward and bi-directional LSTM-RNNs," in *Proc. Interspeech*, 2018, pp. 392–395.

[18] W. Xiong, L. Wu, J. Zhang, and A. Stolcke, "Session-level language modeling for conversational speech," in *Proc. EMNLP*, 2018, pp. 2764–2768.

[19] K. Irie, A. Zeyer, R. Schlüter, and H. Ney, "Language modeling with deep Transformers," in *Proc. Interspeech*, 2019, pp. 3905–3909.

[20] S. Parthasarathy, W. Gale, X. Chen, G. Polovets, and S. Chang, "Long-span language modeling for speech recognition," *arXiv:1911.04571v1 [cs.CL]*.

[21] K. Irie, A. Zeyer, R. Schlüter, and H. Ney, "Training language models for long-span corss-sentence evaluation," in *Proc. ASRU*, 2019, pp. 419–426.

[22] G. Sun, C. Zhang, and P. C. Woodland, "Transformer language models with LSTM-based cross-utterance information representation," in *Proc. ICASSP*, 2021, pp. 7363–7367.

[23] F. Jelinek, *Statistical methods for speech recognition*. The MIT Press, Cambridge, MA, 1998.

[24] R. Kneser and H. Ney, "Improved backing-off for M-gram language modeling," in *Proc. ICASSP*, 1995, pp. 181–184.

[25] M. Sundermeyer, R. Schlüter, and H. Ney, "LSTM neural networks for language modeling," in *Proc. Interspeech*, 2012.

[26] A. Vaswani *et al.*, "Attention is all you need," in *Proc. NIPS*, 2017, pp. 5998–6008.

[27] M. Auli, M. Galley, C. Quirk, and G. Zweig, "Joint language and translation modeling with recurrent neural networks," in *Proc. EMNLP*, 2013, pp. 1044–1054.

[28] M. Sundermeyer, H. Ney, and R. Schlüter, "From feedforward to recurrent LSTM neural networks for language modeling," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 3, pp. 517–529, Mar. 2015.

[29] X. Liu, X. Chen, Y. Wang, M. J. F. Gales, and P. C. Woodland, "Two efficient lattice rescoring methods using recurrent neural network language models," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 8, pp. 1438–1449, Aug. 2016.

[30] S. Kumar, M. Nirschl, D. Holtmann-Rice, H. Liao, A. T. Suresh, and F. Yu, "Lattice rescoring strategies for long short term memory language models in speech recognition," in *Proc. ASRU*, 2017, pp. 165–172.

[31] H. Xu *et al.*, "A pruned RNNLM lattice-rescoring algorithm for automatic speech recognition," in *Proc. ICASSP*, 2018, pp. 5929–5933.

[32] X. Li, R. Singh, and R. M. Stern, "Combining search spaces of heterogeneous recognizers for improved speech recognition," in *Proc. Interspeech*, 2002, pp. 405–408.

[33] H. Xu, D. Povey, L. Mangu, and J. Zhu, "An improved consensus-like method for minimum Bayes risk decoding and lattice combination," in *Proc. ICASSP*, 2010, pp. 4938–4941.

[34] K. Maekawa, "Corpus of spontaneous Japanese: Its design and evaluation," in *Proc. Workshop on Spontaneous Speech Processing and Recognition (SSPR)*, 2003, pp. 7–12.

[35] D. Povey *et al.*, "The Kaldi speech recognition toolkit," in *Proc. ASRU*, 2011.

[36] A. Paszke *et al.*, "PyTorch: An imperative style, high-performance deep learning library," in *Proc. NeurIPS*, 2019, pp. 8024–8035.

[37] PyTorch Examples, https://github.com/pytorch/examples, 2021.

[38] V. Peddinti, D. Povey, and S. Khudanpur, "A time delay neural network architecture for efficient modeling of long temporal contexts," in *Proc. Interspeech*, 2015, pp. 3214–3218.

[39] C. Allauzen, M. Riley, J. Schalkwyk, W. Skut, and M. Mohri, "OpenFst: A general and efficient weighted finite-state transducer library," in *Proc. CIAA*, 2007, pp. 11–23.

[40] A. Krogh and J. Vedelsby, "Neural network ensembles, cross validation, and active learning," in *Proc. NIPS*, 1994, pp. 231–238.

[41] S. Watanabe and J. L. Roux, "Black box optimization for automatic speech recognition," in *Proc. ICASSP*, 2014, pp. 3256–3260.

[42] J. G. Fiscus, "A post-processing system to yield reduced word error rates: Recognizer Output Voting Error Reduction (ROVER)," in *Proc. ASRU*, 1997, pp. 347–354.