NEURAL LATTICE SEARCH FOR SPEECH RECOGNITION

Rao Ma, Hao Li, Qi Liu, Lu Chen, Kai Yu

SpeechLab, Department of Computer Science and Engineering
MoE Key Lab of Artificial Intelligence
Shanghai Jiao Tong University, Shanghai, China
{rm1031, lh575526, liuq901, chenlusz, kai.yu}@sjtu.edu.cn

ABSTRACT

To improve the accuracy of automatic speech recognition, a two-pass decoding strategy is widely adopted. The first-pass model generates compact word lattices, which are utilized by the second-pass model to perform rescoring. Currently, the most popular rescoring methods are N-best rescoring and lattice rescoring with long short-term memory language models (LSTMLMs). However, these methods encounter the problem of limited search space or inconsistency between training and evaluation. In this paper, we address these problems with an end-to-end model for accurately extracting the best hypothesis from the word lattice. Our model is composed of a bidirectional LatticeLSTM encoder followed by an attentional LSTM decoder. The model takes word lattice as input and generates the single best hypothesis from the given lattice space. When combined with an LSTMLM, the proposed model yields 9.7% and 7.5% relative WER reduction compared to N-best rescoring methods and lattice rescoring methods within the same amount of decoding time.

Index Terms**—** speech recognition, word lattice, lattice-to-sequence, attention models, forward-backward algorithm

1. INTRODUCTION

Recent years have seen great progress on automatic speech recognition (ASR) based applications, ranging from personal assistants [1] to smart speakers [2]. The quality of these applications is heavily dependent on the accuracy of the 1-best hypotheses yielding from the underlying ASR systems. An ASR system typically consists of three components to recognize spoken signals: acoustic model, pronunciation and language model. The performance of the ASR system could be affected by many factors, such as the identification of multi-speakers [3], the lack of training corpus [4] or the existence of environmental noises [5]. Therefore the 1-best hypothesis may contain many errors in several situations. To address this problem, a two-pass decoding strategy is widely adopted to improve the precision of recognition. The first-pass ASR system exposes a portion of its search space by outputting multiple hypotheses, which could be represented in the compact form of word lattice. Then in the second-pass decoding, a more sophisticated model is used to search from the word lattice with *N-best rescoring* or *lattice rescoring*.

In the N-best rescoring method, ASR system calculates the N best paths from the generated word lattice [6] and forms into a sentence list for reranking. The most popular models for N-best rescoring are recurrent neuron network language models (RNNLMs) [7, 8]. In particular, long short-term memory (LSTM)-based RNNLMs are widely used since they solve the vanishing gradient problem [9]. An LSTMLM computes scores for each sentence in a given N-best hypotheses list. These language model scores are then interpolated with the ASR scores attached to each of the hypotheses to perform reranking of the N-best list. Nevertheless, N-best lists can only encode a tiny subset of all the possible hypotheses in the word lattice, which could be formulated as the limited search space problem.

For the above-mentioned problem of N-best rescoring, one possible solution is to increase the size of the candidate list. However, N-best list can only cover a relatively small portion of the overall lattice even with a large N, since most hypotheses only differ in a few word positions. Moreover, time consumption rises as N grows, which is unfavorable in real-time applications. In practice, N is set at 50 to 1000. Lattice rescoring [10, 11, 12] targets this problem by directly extracting the single best hypothesis from the lattice space. A pre-trained language model is used to decode the word lattice with the forward-backward algorithm. In addition to improving ASR performance, rescoring lattices is much faster than rescoring N-best lists. However, LSTMLM is trained with the criterion to predict the next word and is not well-suited to the task of finding the best path in the lattice. For instance, only positive examples are taken into consideration in the training, therefore it might be hard for the model to discriminate among alternative hypotheses in the evaluation.

Recently, the end-to-end (E2E) ASR system has been developed, which is a unified model blending all the components of the traditional ASR system [13]. Nevertheless, the traditional two-pass ASR system is still the mainstream of the industry due to the more robust performance. End-to-end ASR models could also benefit from the two-pass decoding strategy [14, 15]. In the first-pass, word lattices are built by merging nodes of the beam search outputs.

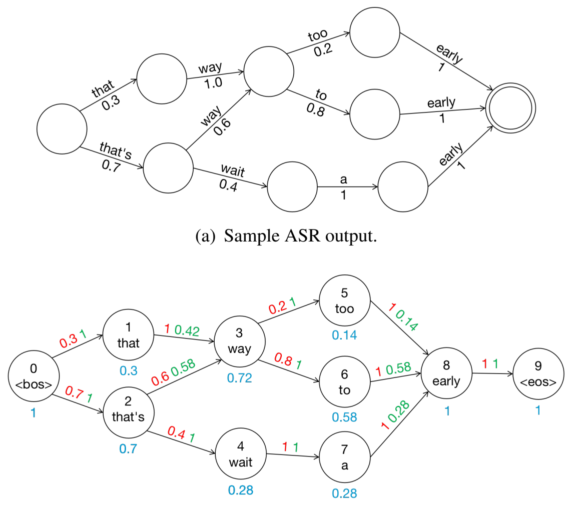
As shown above, extracting the best hypothesis from a word lattice is a crucial task in ASR systems, which is the main focus of this paper. We propose a novel lattice-to-sequence (L2S) model that could search for the best path from a word lattice in an end-to-end fashion, which addresses the limited search space problem and the inconsistency between training and evaluation. The proposed model is composed of a bidirectional lattice encoder network and a decoder with the attention mechanism. In the training phase, the encoder computes hidden vectors for each lattice node, which are utilized by the decoder to generate the speech transcription. In the evaluation, we incorporate the forward-backward algorithm to generate in the given lattice space. Experiments are conducted on the famous Swb-Fisher corpus. A consistent performance gain is obtained over the traditional N-best rescoring methods and lattice rescoring methods.

This study was supported by the National Key Research and Development Program of China (Grant No.2017YFB1002102), the Major Program of National Social Science Foundation of China (No.18ZDA293) and the Major Program of Science and Technology Commission of Shanghai Municipality (STCSM) 17JC1404104.

Despite the vast search space, we show that the proposed L2S model is also time-efficient compared to previous methods.

2. WORD LATTICES

A lattice is a compact representation of multiple alternative word sequences, which is efficient in encoding the ambiguity of upstream systems. Sub-sequences are shared among diverse hypotheses, allowing an exponential number of weighted hypotheses to be represented in a single lattice. Previous work has shown that incorporating lattices into the training of neural networks could improve task performance. Examples include encoding multiple word segmentation [16], polysemy alternatives [17] and ASR output [18, 19].



(a) Sample ASR output.

(b) Lattice after pre-processing, containing forward (red) / marginal (blue) / backward (green) probability scores.

Fig. 1. Sample word lattice for reference *"that's way too early"*.

A word lattice from the ASR system is generally represented as a directed acyclic graph $G = \langle V, E\rangle$ where edges are labeled with words and weights. Figure 1(a) shows an example output generated by Kaldi [20] toolkit. We conduct the line-graph algorithm [21] to convert the lattice into an equivalent form where word labels are assigned to nodes and weights remaining on the edges. This guarantees that each word corresponds to one hidden state in the encoder. The lattice after processing is shown in Figure 1(b). We add two special tokens BOS and EOS to represent the start and the end of the sentence. Each node is also assigned with a number in topological order such that a child node comes after all its parent nodes.

Each edge of the original word lattice is associated with an n-gram language model score and an acoustic score, which could be combined by an acoustic scale factor. The lattice scores are typically given in forward-normalized fashion, i.e. the probability on all the outgoing edges for each node sum to 1. We could further derive marginal/backward probability scores in the lattice graph. For edge $e_{i,j} \in E$, assume the forward score is $f_{i,j}$ and denote the set of node i's predecessors as P_i. Then we can compute marginal score $m_0 = 1$ and $m_i = \sum_{k\in P_i} m_k f_{k,i} (i > 0)$ on each node by using forward algorithm. The backward score is calculated as $b_{i,j} = m_i f_{i,j}/m_j$ on each edge. All three types of scores are illustrated in Figure 1(b) and will be integrated in the L2S model in the following sections.

3. NEURAL LATTICE-TO-SEQUENCE MODEL

We propose an E2E model targeting the lattice search problem. A bidirectional LatticeLSTM is adopted to encode the given lattice and compute context representations for each lattice node. An attentional LSTM decoder is utilized for hypotheses generation conditioned on the encoder outputs. In the training, the model is trained to generate the reference hypothesis with the word lattice as input. In the evaluation, the model could generate the predicted hypothesis in an autoregressive fashion or with the constraint that the generated hypothesis is in the given lattice space. The detailed model structure and the training and evaluation procedures are presented.

3.1. Encoder

Given $\mathbf{x}_t, \mathbf{h}_{t-1}, \mathbf{c}_{t-1}$ as input, LSTM produces hidden vector $\mathbf{h}_t$ and cell vector $\mathbf{c}_t$ at each time step with the formula:

$$\begin{bmatrix} \mathbf{i}_t \\ \mathbf{o}_t \\ \mathbf{f}_t \\ \mathbf{g}_t \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{bmatrix} \left(\mathbf{W}^\top \begin{bmatrix} \mathbf{x}_t \\ \mathbf{h}_{t-1} \end{bmatrix} + \mathbf{b} \right)$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \mathbf{g}_t$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t),$$

where $\mathbf{W}$ is the weight matrix parameter and $\mathbf{b}$ is the bias.

The basic LSTM is applicable to tasks where words are given in a sequence linearly such as language modeling. However, there may exist several preceding words for each word in the lattice graph. In this work, we utilize a bidirectional LatticeLSTM encoder to model such dependency. The forward encoder scans word labels of the lattice in topological order and generates hidden state for each word. At time step i, we first identify all the predecessors of w_i and denote the set as P_i. The hidden vectors and cell vectors of preceding time steps are aggregated into

$$\tilde{\mathbf{h}}_i = \frac{1}{|P_i|} \sum_{k\in P_i} \mathbf{h}_k, \; \tilde{\mathbf{c}}_i = \frac{1}{|P_i|} \sum_{k\in P_i} \mathbf{c}_k.$$

For example, in Figure 1(b), node 3 has two incoming edges $e_{1,3}$ and $e_{2,3}$. Therefore, hidden states for time step 1 and 2 are compressed into $\tilde{\mathbf{h}}_3$ and $\tilde{\mathbf{c}}_3$. Assume the embedding for w_i is $\mathbf{x}_i$, then the hidden state of the i-th step is calculated with the LSTM function:

$$\mathbf{h}_i, \mathbf{c}_i = \text{LSTM}(\mathbf{x}_i, \tilde{\mathbf{h}}_i, \tilde{\mathbf{c}}_i).$$

We could incorporate the backward-normalized scores into the composition of $\tilde{\mathbf{h}}_i$ and $\tilde{\mathbf{c}}_i$, therefore biasing the aggregated state encoding towards the more likely predecessor hidden states. Note that the backward scores are selected in the forward encoder since the backward-normalized scores on each node's incoming edges sum to 1. The modified formula is given as

$$\tilde{\mathbf{h}}_i = \sum_{k\in P_i} \frac{b_{k,i}^{\mathbf{S}_h}}{\sum_{k'\in P_i} b_{k',i}^{\mathbf{S}_h}} \mathbf{h}_k, \; \tilde{\mathbf{c}}_i = \sum_{k\in P_i} \frac{b_{k,i}^{\mathbf{S}_c}}{\sum_{k'\in P_i} b_{k',i}^{\mathbf{S}_c}} \mathbf{c}_k,$$

where $\mathbf{S}_h, \mathbf{S}_c$ are parameter vectors to apply *tempered softmax*, resulting in softer probability distributions.

The backward LSTM could be computed in a similar way in the reverse topological order. Finally we concatenate the hidden vectors of the two LSTMs to form the context representation of w_i.

3.2. Decoder

The decoder is a forward LSTM that outputs token at each time step, conditioned on the context vector produced by the attention mechanism. The decoder hidden state is initialized with the final encoder hidden state, i.e. $\mathbf{s}_0 = \mathbf{h}_N, \mathbf{n}_0 = \mathbf{c}_N$. At time step j, given the previous LSTM state $\mathbf{s}_{j-1}, \mathbf{n}_{j-1}$ and the embedding $\mathbf{y}_j$ for the j-th target word, the hidden vector and cell vector are updated as

$$\mathbf{s}_j, \mathbf{n}_j = \text{LSTM}(\mathbf{y}_j, \mathbf{s}_{j-1}, \mathbf{n}_{j-1}).$$

With attention mechanism, the source annotations $\{\mathbf{h}_i\}_1^N$ could be summarized into a fixed-size context vector which is calculated as $\mathbf{q}_j = \sum_{i=1}^N \alpha_{ij}\mathbf{h}_i$. α_{ij} measures the degree that the source hidden vector $\mathbf{h}_i$ and the decoder state $\mathbf{s}_j$ match:

$$r_{ij} = \mathbf{v}_a^\top \tanh(\mathbf{W}_a\mathbf{s}_j + \mathbf{U}_a\mathbf{h}_i)$$
$$\alpha_{ij} = \frac{\exp(r_{ij})}{\sum_{i'=1}^N \exp(r_{i'j})},$$

where $\mathbf{W}_a, \mathbf{U}_a$ and $\mathbf{v}_a$ are model parameters. We could also include the marginal lattice scores into the generation of α. Therefore, words with high lattice scores may have a higher probability of being selected by the attention layer than words with lower scores. The computation of r_{ij} is replaced with:

$$r_{ij} = \mathbf{v}_a^\top \tanh(\mathbf{W}_a\mathbf{s}_j + \mathbf{U}_a\mathbf{h}_i + \mathbf{S}_a m_i).$$

Here $\mathbf{S}_a$ is a scaling parameter vector that maps the marginal score to a vector of the same dimension as $\mathbf{v}_a$.

We employ a simple concatenation layer to combine the information from the hidden vector $\mathbf{s}_j$ and the context vector $\mathbf{q}_j$. $\mathbf{o}_j$ is fed into a softmax layer to compute the predictive distribution P_j:

$$\mathbf{o}_j = \tanh(\mathbf{W}_c[\mathbf{s}_j; \mathbf{q}_j] + \mathbf{b}_c)$$
$$P_j = \text{softmax}(\mathbf{W}_o\mathbf{o}_j + \mathbf{b}_o).$$

3.3. Training and Decoding

We train the proposed L2S model by feeding it with pairs of word lattices and their corresponding reference hypotheses. With the trained L2S model, in the evaluation phase, we could generate the transcription in an unconstrained autoregressive fashion. However, the generated hypothesis might be error-prone if we impose no constraints on the transcription in the decoding stage. A superior approach would be to extract the single best hypothesis in the lattice-based search space. In this paper, we employ the adapted forward-backward algorithm with cardinality pruning as in [10]. The lattice nodes are processed in topological order starting from the BOS node. For each node, a list of partial hypotheses is retained. Each candidate hypothesis stores the hidden state and the cumulative score leading from the BOS node up to the current node. The algorithm extends the hidden states and scores by words on the successor nodes. Finally, the best scoring hypothesis of the lattice could be obtained in the EOS node. For each partial hypothesis s, we calculate the score as

$$\text{score}(s) = (1-\lambda)\cdot \text{L2S}(s) + \lambda \cdot (\text{LSTMLM}(s) + \text{AM}(s)).$$

where AM denotes the trained acoustic model. When $\lambda = 1$, the algorithm reduces to the original lattice rescoring method. When $\lambda = 0$, the hypotheses are ranked only based on the L2S score. In other cases, the scores produced by the L2S model and LSTMLM are combined for estimation. To reduce the computational cost, only the K-best hypotheses are maintained at each node.

4. EXPERIMENTS

4.1. Data Description

The experiments are conducted on Switchboard (SWBD) 300 hours and Swb-Fisher 2000 hours corpus. We follow the EESEN [22] SWBD recipe to build the baseline phone-based CTC ASR system. A 5-layer BLSTM acoustic model with the hidden size of 320 is trained on SWBD 300 hours speech, and a 3-gram language model is trained on Swb-Fisher 2000 hours transcripts. All word lattices are generated by the WFST-based approach as described in [22].

Corpus		# Utt.	Length	Node	Edge
train	FSH	1,871,512	12.1	109	189
	SWB	258,488	12.8	125	199
eval2000	CH	2,627	9.2	142	273
	SWB	1,831	12.6	122	216
rt03	FSH	3,970	10.2	131	245
	SWB	4,452	9.8	173	338

Table 1. Lattice statistics for different parts of the training corpus and two test sets. # utt. and length denote the total number and average length of the reference hypotheses. Node and edge refer to the average number of the nodes and edges of word lattices.

The validation and test sets are prepared by randomly choosing from the original Switchboard and Fisher transcriptions, resulting in 6,731 and 5,000 sentences respectively. Vocabulary size of 31K is used. We evaluate our models on the eval2000 and rt03 set. The word lattices are pruned with a beam size of 6.0. Details of the data are given in Table 1. The biggest word lattice in eval2000 corpus contains 1015 nodes with $1.9e^{17}$ possible paths, which remains difficult for N-best rescoring methods to process.

4.2. Experimental Settings

Our proposed L2S model contains two bidirectional LatticeLSTM layers of size 256 per encoder direction. Word Embedding size is also set to 256. The decoder contains two LSTM layers with 256 neurons each. A projection layer is inserted to map the encoder outputs from 512 dimensions to 256 dimensions. We tie the input embedding of the encoder, the input embedding of the decoder and the output embedding of the decoder since they consist of the same vocabulary. We train 4096 tokens on each batch. Adam optimizer is used for training, with the learning rate of e^{-3} for the first three epochs and e^{-4} for the last two epochs. Dropout rate is set to 0.15. The baseline LSTMLM has the same model structure as the L2S decoder and also adopts weight tying. The LSTMLM is trained for 10 epochs with a batch size of 256. All the LSTMLM and L2S models are trained independently on the same data.

4.3. Experimental Results

In Table 2, We compare our proposed L2S model to 3-gram LM and LSTMLM with different decoding strategies in terms of accuracy and time impact.[1] The first line of the table shows the baseline results produced by a statistical skip-gram model along with the acoustic model. The high WER exhibits the necessity of incorporating a second pass decoding strategy such as N-best rescoring.

[1]All the neural networks are trained and evaluated on a single NVIDIA GeForce GTX 1080Ti GPU. The N-best algorithm is conducted by the Kaldi toolkit on a Xeon E5-2670 CPU. All experiments are run single-threaded.

	Model	PPL	Decoding Method	eval2000			rt03			Avg.
				CH	SWB	sec./utt.	FSH	SWB	sec./utt.	
1	3-gram	-	first-pass output	27.3	15.2	0.0074	20.1	29.4	0.0084	23.1
2	LSTMLM	54.0	rescoring 50-best list	25.5	13.1	0.28	17.9	27.5	0.31	21.6
3			rescoring 100-best list	25.3	13.0	0.44	17.7	27.3	0.48	21.4
4			rescoring 500-best list	25.0	12.8	1.72	17.5	27.2	1.89	21.2
5			rescoring 1000-best list	25.1	12.7	3.23	17.5	27.1	3.42	21.2
6			lattice constrained ($K = 1$)	25.4	13.2	0.08	17.7	27.2	0.09	21.4
7			lattice constrained ($K = 3$)	25.1	12.7	0.21	17.4	26.9	0.25	21.1
8			lattice constrained ($K = 10$)	24.9	12.8	0.70	17.3	26.7	0.84	21.0
9	L2S	2.3	unconstrained	24.4	13.3	0.08	17.9	27.5	0.10	21.4
10			lattice constrained ($K = 1$)	23.6	12.7	0.17	16.9	26.2	0.19	20.4
11			lattice constrained ($K = 3$)	23.6	12.7	0.35	16.9	26.1	0.42	20.4
12			lattice constrained ($K = 10$)	23.6	12.7	0.97	16.9	26.1	1.17	20.4
13	L2S + LSTMLM	-	lattice constrained ($K = 1$)	**22.9**	**11.8**	0.24	**16.0**	**25.1**	0.27	**19.5**

Table 2. Comparison of various perplexity (PPL), WER [%] and timing results obtained with the trigram LM, LSTMLM, L2S model, and the combination of L2S model and LSTMLM. "first-pass" refers to the 1-best output from ASR first-pass decoding.

For LSTMLM, we test the N-best rescoring and lattice rescoring methods. The results from line 2 to 5 show that by rescoring N-best hypotheses, LSTMLM steadily reduces WER results from the ASR baseline. Rescoring on the 1000-best list gives the lowest WER of 21.2. However, when N is larger than 500, the inference comes with large latency. Most time is spent on generating the N-best list from the word lattice. Rescoring lattice with $K = 1$ (line 6) gives similar performance as rescoring the 100-best list while reducing the time effort by a factor of five. Increasing the candidate hypotheses maintained at each lattice node (K) results in better WER (line 7-8).

The results of the L2S model are listed from line 9 to 12. The perplexity on the test set is only 2.3, which implies that our model is as confused as if it has to choose uniformly and independently among 2.3 possibilities for each word on the test set, different from LSTMLM where the PPL is 54.0. Generating the target hypothesis with unconstrained greedy search (line 9) produces WER comparable to rescoring the 100-best list. There are chances for the model to output semantically similar words that are not in the lattice, which could be mitigated with the forward-backward algorithm. Line 10 presents an average WER of 20.4, greatly outperforming all the baseline models. Nevertheless, larger values of K have litter impacts on accuracy. Since our model is trained to discriminate among alternative sequences and achieves PPL of only 2.3, it yields more certainty at each word prediction compared to plain LSTMLM. The results imply that L2S model allows for greedy lattice search, which is a desirable property for improving run-time efficiency.

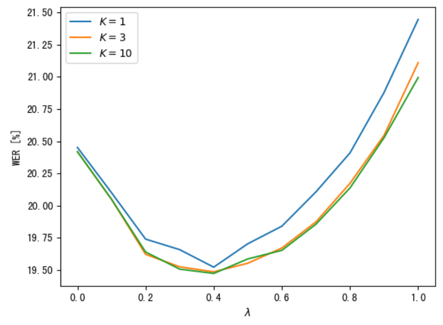


Fig. 2. WER [%] of the combination of L2S model and LSTMLM with different interpolation rate λ (weight of the LSTMLM score).

We also investigate the combination of L2S scores and LSTMLM scores in lattice rescoring. As line 13 shows, WER result drops from 21.6 to 19.5 compared to 50-best rescoring with less decoding time. To validate whether the performance gain comes from the simple ensemble technique, we also conduct lattice rescoring on the ensemble of two LSTMLMs. The average WER is 21.3 when $K = 1$. The results indicate that the L2S model and the LSTMLM have complementary abilities and our proposed model could incorporate the ability of LSTMLM for accurate generations. Figure 2 shows the average WER obtained with different beam sizes and different interpolation rates λ. The optimal values of λ fall around 0.4.

Model	PPL	eval2000		rt03		Avg.
		CH	SWB	FSH	SWB	
L2S	2.33	23.6	12.7	16.9	26.2	20.4
w/o m score	2.35	23.8	13.1	17.4	26.5	20.8
w/o f/b score	2.42	24.1	13.4	17.6	26.6	21.0
w/o m/f/b score	2.43	24.1	13.3	17.7	26.8	21.1

Table 3. Ablation over the effect of lattice scores. m/f/b refer to marginal/forward/backward score respectively.

We conduct an ablation analysis to study the impact of integrating lattice scores. Here we utilize the forward-backward algorithm with $K = 1$ as the decoding strategy. Table 3 indicates that our model takes advantage of acoustic scores and graph scores in the inference. Line 3 shows that model performance degrades drastically without using forward/backward-normalized scores, since having multiple contradicting predecessor lattice nodes may result in poor context representations. Incorporating the marginal scores into the attention mechanism provides additional WER improvements.

5. CONCLUSION AND FUTURE WORK

In this work, we introduce a novel end-to-end lattice-to-sequence model for second-pass decoding. Our model addresses the problems of limited search space and inconsistency between training and evaluation in LSTMLM rescoring approaches. Experimental results show that our method achieves 0.8% and 0.6% absolute WER reduction compared to N-best rescoring and lattice rescoring. The proposed model is able to utilize LSTMLM estimations for further WER reduction. Our framework can be easily adapted to other research fields that need to extract the 1-best path from a given word lattice, e.g. machine translation and E2E ASR system. Our future work will concentrate on applying the L2S model to other tasks.

6. REFERENCES

[1] Matthew B Hoy, “Alexa, siri, cortana, and more: an introduction to voice assistants,” *Medical reference services quarterly*, vol. 37, no. 1, pp. 81–88, 2018.

[2] Bo Li, Tara N Sainath, Arun Narayanan, Joe Caroselli, Michiel Bacchiani, Ananya Misra, Izhak Shafran, Hasim Sak, Golan Pundak, Kean K Chin, et al., “Acoustic modeling for google home,” in *Annual Conference of the International Speech Communication Association*, 2017, pp. 399–403.

[3] Yan-min Qian, Chao Weng, Xuan-kai Chang, Shuai Wang, and Dong Yu, “Past review, current progress, and challenges ahead on the cocktail party problem,” *Frontiers of Information Technology & Electronic Engineering*, vol. 19, no. 1, pp. 40–63, 2018.

[4] Yajie Miao, Florian Metze, and Shourabh Rawat, “Deep maxout networks for low-resource speech recognition,” in *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*. IEEE, 2013, pp. 398–403.

[5] Tian Tan, Yanmin Qian, Hu Hu, Ying Zhou, Wen Ding, and Kai Yu, “Adaptive very deep convolutional residual network for noise robust speech recognition,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 8, pp. 1393–1405, 2018.

[6] Yen-Lu Chow and Richard Schwartz, “The n-best algorithm: An efficient procedure for finding top n sentence hypotheses,” in *Proceedings of the workshop on Speech and Natural Language*. Association for Computational Linguistics, 1989, pp. 199–202.

[7] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur, “Recurrent neural network based language model,” in *Annual Conference of the International Speech Communication Association*, 2010.

[8] Tomáš Mikolov, Stefan Kombrink, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur, “Extensions of recurrent neural network language model,” in *2011 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2011, pp. 5528–5531.

[9] Martin Sundermeyer, Ralf Schlüter, and Hermann Ney, “Lstm neural networks for language modeling,” in *Annual Conference of the International Speech Communication Association*, 2012.

[10] Michael Auli, Michel Galley, Chris Quirk, and Geoffrey Zweig, “Joint language and translation modeling with recurrent neural networks,” in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 2013, pp. 1044–1054.

[11] Martin Sundermeyer, Zoltán Tüske, Ralf Schlüter, and Hermann Ney, “Lattice decoding and rescoring with long-span neural network language models,” in *Annual Conference of the International Speech Communication Association*, 2014.

[12] Shankar Kumar, Michael Nirschl, Daniel Holtmann-Rice, Hank Liao, Ananda Theertha Suresh, and Felix Yu, “Lattice rescoring strategies for long short term memory language models in speech recognition,” in *2017 IEEE Workshop on Automatic Speech Recognition and Understanding*. IEEE, 2017, pp. 165–172.

[13] Zhehuai Chen, Mahaveer Jain, Yongqiang Wang, Michael L Seltzer, and Christian Fuegen, “Joint grapheme and phoneme embeddings for contextual end-to-end asr,” in *Annual Conference of the International Speech Communication Association*, 2019, pp. 3490–3494.

[14] Chung-Cheng Chiu, Tara N Sainath, Yonghui Wu, Rohit Prabhavalkar, Patrick Nguyen, Zhifeng Chen, Anjuli Kannan, Ron J Weiss, Kanishka Rao, Ekaterina Gonina, et al., “State-of-the-art speech recognition with sequence-to-sequence models,” in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2018, pp. 4774–4778.

[15] Michał Zapotoczny, Piotr Pietrzak, Adrian Łańcucki, and Jan Chorowski, “Lattice generation in attention-based speech recognition models,” in *Annual Conference of the International Speech Communication Association*, 2019, pp. 2225–2229.

[16] Jinsong Su, Zhixing Tan, Deyi Xiong, Rongrong Ji, Xiaodong Shi, and Yang Liu, “Lattice-based recurrent neural network encoders for neural machine translation,” in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[17] Jacob Buckman and Graham Neubig, “Neural lattice language models,” *Transactions of the Association for Computational Linguistics*, vol. 6, pp. 529–541, 2018.

[18] Faisal Ladhak, Ankur Gandhe, Markus Dreyer, Lambert Mathias, Ariya Rastrow, and Björn Hoffmeister, “Latticernn: Recurrent neural networks over lattices,” in *Annual Conference of the International Speech Communication Association*, 2016, pp. 695–699.

[19] Matthias Sperber, Graham Neubig, Jan Niehues, and Alex Waibel, “Neural lattice-to-sequence models for uncertain inputs,” in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2017, pp. 1380–1389.

[20] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, et al., “The kaldi speech recognition toolkit,” in *2011 IEEE Workshop on Automatic Speech Recognition and Understanding*. IEEE, 2011.

[21] Robert L Hemminger, “Line graphs and line digraphs,” *Selected topics in graph theory*, 1983.

[22] Yajie Miao, Mohammad Gowayyed, and Florian Metze, “Eesen: End-to-end speech recognition using deep rnn models and wfst-based decoding,” in *2015 IEEE Workshop on Automatic Speech Recognition and Understanding*. IEEE, 2015, pp. 167–174.