LATTENTION: LATTICE-ATTENTION IN ASR RESCORING

Prabhat Pandey, Sergio Duarte Torres*, Ali Orkan Bayer, Ankur Gandhe, Volker Leutnant*

Amazon Alexa AI

ABSTRACT

Lattices form a compact representation of multiple hypotheses generated from an automatic speech recognition system and have been shown to improve performance of downstream tasks like spoken language understanding and speech translation, compared to using one-best hypothesis. In this work, we look into the effectiveness of lattice cues for rescoring n-best lists in second-pass. We encode lattices with a recurrent network and train an attention encoder-decoder model for n-best rescoring. The rescoring model with attention to lattices achieves 4-5% relative word error rate reduction over first-pass and 6-8% with attention to both lattices and acoustic features. We show that rescoring models with attention to lattices outperform models with attention to n-best hypotheses. We also study different ways to incorporate lattice weights in the lattice encoder and demonstrate their importance for n-best rescoring.

Index Terms— Lattice, attention, rescoring, speech recognition

1. INTRODUCTION

In a typical multi-pass automatic speech recognition (ASR) system, the first-pass system produces lattices [1] or n-best hypotheses [2] which are rescored in the second-pass. More commonly, a neural language model (NLM) trained on large amount of text data is used in the second-pass rescoring [3, 4]. Recently, stronger rescoring models utilizing acoustic information have been proposed. In [5], a listen-attend-spell [6] based model was proposed to rescore n-best lists where the encoder is shared with the first-pass recurrent neural network transducer (RNN-T) [7] model. Similarly, in [8], NLM was extended to attend to audio features generated by the acoustic model in the first-pass ASR system. In further extension to [5], a deliberation network [9] based model with additional attention to n-best hypotheses was introduced in [10]. A more compact representation of the first-pass decoding output are lattices. Lattices encode multiple hypotheses in a condensed form and carry the uncertainties from the first-pass decoding. Using a lattice encoder instead of the 1-best output has been shown to improve performance of downstream tasks like speech translation [11, 12, 13, 14] and spoken language understanding [15, 16].

There has been some previous work on rescoring lattices in second-pass [17, 18] instead of a subset of hypotheses in the n-best list, making use of the richer information in the lattices. In this work, we utilize lattice information for n-best rescoring by encoding them with a recurrent network. We train an attention based encoder-decoder model which attends to the lattice encoder and run the decoder in the teacher-forcing mode to rescore n-best lists. We experiment with different encoders: 1-best, n-best, lattice and audio features extracted from the first-pass model. We employ minimum word error rate (MWER) [19] training criterion which has been shown to improve accuracy of attention-based rescoring models [5, 8, 10].

There has already been some work on representing lattice structures in recurrent encoders [11, 12, 15] and transformers [13, 14] models. In [11], LatticeLSTM was proposed for machine translation, which extends TreeLSTM [20] to encode directed acyclic graphs with weights. We utilize LatticeLSTM with certain modifications as the lattice encoder for ASR n-best rescoring in this work. Specifically, following are the contributions of this paper: (1) We propose a simplified method for encoding lattice weights with similar performance as [11], (2) We show that lattice-attention rescoring model can provide 4-5% relative word error rate reduction (WERR) over first-pass, (3) LatticeLSTM-based lattice encoder results in more improvements compared to n-best deliberation encoder [10], even for lattices containing same hypotheses as the n-best, (4) Attending to both audio and lattice further reduces word error rate (WER), resulting in 6-8% relative WERR over first-pass, (5) We study the effect of different mechanisms to incorporate lattice weights in LatticeLSTM and show that unweighted lattice encoders (TreeLSTM) are detrimental for attention-based models and integrating lattice weights is important to mitigate confusions arising from contradictory lattice arcs.

2. LATTICE-ATTENTION MODEL

2.1. Lattice representation

We represent the lattices generated from first-pass ASR system as a weighted finite state transducer (WFST) and apply epsilon removal, determinization, minimization, weight pushing to initial states, removal of total weights and eventually, topological sorting of nodes [21]. The original lattices have labels on its arcs (we refer to them as edge-labeled lattice, an example is shown in Figure 1). In [11], node-labeled lattices are used instead of edge-labeled lattices in LatticeLSTM because of its intuitive appeal as hidden states represent a single token in LatticeLSTM in case of node-labeled lattices. We too use node-labeled lattices which are generated by applying line-graph algorithm [22] on edge-labeled lattices. Figure 2 shows the node-labeled lattice for the edge-labeled lattice depicted in Figure 1. We explain how the weights from edge-labeled-lattices are transformed to node-labeled lattices in Section 2.3.

2.2. Forward-weight normalization in edge-labeled lattices

The costs on the lattices are usually not in the probability space. So, before converting to node-labeled lattice, we forward-normalize costs in the edge-labeled lattice so that weights on all arcs outgoing from a node sum to 1. Let e be an edge in the edge-labeled lattice (and a node in the node-labeled lattice), $o(e)$ and $d(e)$ denote the origin and destination nodes of the edge e in the edge-labeled lattice, respectively. Let $O(j)$ denote the edge labels on the outgoing arcs from node j including a dummy edge for final state if j is a

*Equal Contribution

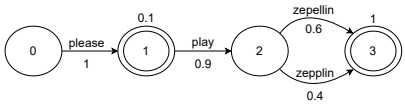

Fig. 1: Example of WFST representation of a edge-labeled-lattice with forward-normalized weights.

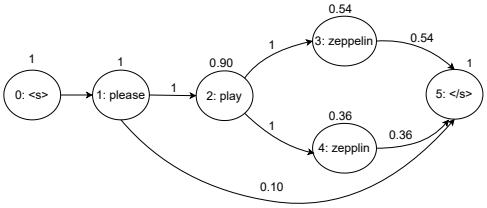

Fig. 2: Example of a node-labeled lattice. Marginal weights of lattice nodes are shown on top of each node and backward-normalized incoming arc weights on top of each arc.

final state. We define forward-normalized weight $w_e^{\mathcal{F}}$ on the edge e as $w_e^{\mathcal{F}} = \frac{\sigma(-cost_e)}{\sum_{j \in O(o(e))} \sigma(-cost_j)}$, where σ is the sigmoid function, $cost_e$ is the first-pass ASR cost on the lattice arc e. We represent the dummy edge from a node j which is a final state as $\mathbb{F}(j)$.

2.3. Weights estimation in node-labeled lattices

We now define two types of weights for node-labeled lattices which are used in LatticeLSTM. First is *marginal weights* which represent the probability of reaching a node given all the paths that contain that node. Let $I(j)$ denote the set of labels on the incoming arcs to node j in the edge-labeled lattice. Then, marginal weight for a node e in the node-labeled lattice is computed as, $w_e^{\mathcal{M}} = \sum_{k \in I(o(e))} w_k^{\mathcal{M}} \cdot w_e^{\mathcal{F}}$, using forward-backward algorithm [23], where $w_e^{\mathcal{F}}$ is the forward-normalized weight for the corresponding edge e in the edge-labeled lattice. We add a single source node, $<s>$, with outgoing arcs to labels on the outgoing arcs of start states in the edge-labeled lattice, and a single sink node, $</s>$, corresponding to the final state in the edge-labeled lattice. Both, $w_{<s>}^{\mathcal{M}}$ and $w_{</s>}^{\mathcal{M}}$ evaluate to 1.

To get the relative importance of different incoming arcs to a node, we use *backward-normalized weights*, which are normalized marginal weights for the source node of the arc. Backward-normalized weights for all incoming arcs to a node sum to 1. Formally, for edges k and e in the edge-labeled lattice, if there is an edge from k to $e, e \neq </s>$, in the node-labeled lattice, the backward-normalized weight on the arc from k to e is computed as, $w_{k,e}^{\mathcal{B}} = \frac{w_k^{\mathcal{M}}}{\sum_{j \in I(o(e))} w_j^{\mathcal{M}}}$. If the destination node of the edge k is a final state in the edge-labeled lattice, the backward-normalized weight on the arc from k to $</s>$ in the node-labeled lattice is defined as $w_{k,</s>}^{\mathcal{B}} = w_k^{\mathcal{M}} \cdot w_{\mathbb{F}(d(k))}^{\mathcal{F}}$ Figure 2 shows marginal weights for each node and backward-normalized weights for each arc of a node-labeled lattice.

2.4. LatticeLSTM

2.4.1. Unweighted LatticeLSTM

Conventional LSTMs use a linear chain network where the hidden state of the network is propagated from the previous node to the next. Although the gates help LSTM to capture long-range dependencies, the linear chain is not well suited for representing linguistic dependencies, which can be better represented by using a tree structured network. TreeLSTMs [20] are designed considering this nature of languages, so that the information is propagated from child nodes to parent nodes. In this paper, we focus on child-sum variant of TreeLSTM that is defined by the following equations, as given in [20]. Given $\boldsymbol{x}_e$ as the word embedding and $P(e)$ as the set of predecessor nodes for a node e in the node-labeled lattice, the memory cell state ($\boldsymbol{c}_e$) and the hidden state ($\boldsymbol{h}_e$) corresponding to node e are computed as follows, where W and U are weight matrices and $\boldsymbol{b}$ is the bias vector:

$$\tilde{\boldsymbol{h}_e} = \sum_{k \in P(e)} \boldsymbol{h}_k \quad (1)$$

$$\boldsymbol{i}_e = \sigma(W^i \boldsymbol{x}_e + U^i \tilde{\boldsymbol{h}_e} + \boldsymbol{b}^i) \quad (2)$$

$$\boldsymbol{f}_{k,e} = \sigma(W^f \boldsymbol{x}_e + U^f \boldsymbol{h}_k + \boldsymbol{b}^f) \quad (3)$$

$$\boldsymbol{o}_e = \sigma(W^o \boldsymbol{x}_e + U^o \tilde{\boldsymbol{h}_e} + \boldsymbol{b}^o) \quad (4)$$

$$\boldsymbol{u}_e = tanh(W^u \boldsymbol{x}_e + U^u \tilde{\boldsymbol{h}_e} + \boldsymbol{b}^u) \quad (5)$$

$$\boldsymbol{c}_e = \boldsymbol{i}_e \odot \boldsymbol{u}_e + \sum_{k \in P(e)} \boldsymbol{f}_{k,e} \odot \boldsymbol{c}_k \quad (6)$$

$$\boldsymbol{h}_e = \boldsymbol{o}_e \odot tanh(\boldsymbol{c}_e) \quad (7)$$

The TreeLSTM cell captures dependencies of incoming arcs by summing uniformly over hidden states of all predecessor nodes. The lattices generated from ASR decoding have scores on its arcs which provide information about the likelihood of different paths. In [11], TreeLSTM was extended to define LatticeLSTM by incorporating marginal and backward-normalized weights. We explain it in detail in the next section.

2.4.2. Incorporating lattice weights

Similar to [11], we employ two mechanisms to incorporate backward-normalized weights in the LatticeLSTM cell structure. In the first approach, referred as *weighted child-sum (WCS)*, the uniform-sum of TreeLSTM in Equation 1 is modified to account for weights on the arc from predecessor nodes:

$$\tilde{\boldsymbol{h}_e} = \sum_{k \in P(e)} w_{k,e}^{\mathcal{B}} \cdot \boldsymbol{h}_k \quad (8)$$

In the second approach, referred as *biased forget gate (BFG)*, the backward-normalized weights are used to decrease the likelihood, for the cells states corresponding to the predecessor nodes with higher weights, of being attenuated in the forget gate. The forget gate computation in Equation 3 is modified as follows in LatticeLSTM:

$$\boldsymbol{f}_{k,e} = \sigma(W^f \boldsymbol{x}_e + U^f \boldsymbol{h}_k + \ln w_{k,e}^{\mathcal{B}} + \boldsymbol{b}^f) \quad (9)$$

Additional learnable coefficients, $\boldsymbol{S}_h$ and $\boldsymbol{S}_f$ of same dimension as hidden states, were introduced in [11] and $w_{k,e}^{\mathcal{B}}$ was transformed to $(w_{k,e}^{\mathcal{B}})^{\boldsymbol{S}_h}$ and $(w_{k,e}^{\mathcal{B}})^{\boldsymbol{S}_f}$ in Equations 8 and 9, respectively. We don't use these coefficients in our implementation to avoid adding any additional parameter over conventional LSTMs.

In [11], marginal weights were integrated in the attention mechanism by *biasing attention weights (BATT)* towards lattices nodes with higher marginal weights. Specifically, at decoder step t, the attention weight α_{et} corresponding to node e is computed as:

$$\alpha_{et} = softmax_e(score(\boldsymbol{h}_e, \boldsymbol{s}_t) + \log w_e^{\mathcal{M}}) \quad (10)$$

where $score(.)$ is the alignment model and $\boldsymbol{s}_t$ is the decoder state at decoding step t.

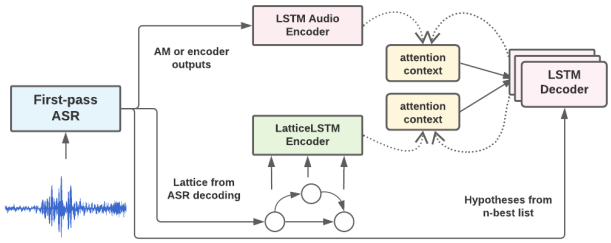

Fig. 3: Rescoring model architecture with attention to audio and lattice encoders.

Instead of modifying the usual attention mechanism, we propose an alternative approach of scaling the encoder outputs in proportion to marginal weights, which we refer as *weighted encoder output (WEO)*. We omit the biasing in the attention weights computation and update the hidden state output as follows:

$$\boldsymbol{h}'_e = w_k^{\mathcal{M}} \cdot \boldsymbol{h}_e \tag{11}$$

Note that the modified hidden states in Equation 11 are used only for attention computation and not in the summation of hidden states of predecessor nodes in Equation 1.

2.5. Rescoring Model Architecture

We use attention-encoder-decoder architecture [24] for our rescoring models and experiment with different encoder inputs: 1-best, n-best, audio and lattices. For audio input, we pass acoustic features extracted from the first-pass model to the audio encoder of the rescoring model. In case of n-best input, we encode each hypothesis separately using the same encoder and concatenate their outputs like [10]. For audio, 1-best and n-best inputs, we use a uni-directional LSTM encoder and for lattice input, we use the (uni-directional) LatticeLSTM encoder from Section 2. We also experiment with attention to multiple encoders. If there are more than one encoder in the model, we concatenate the context vectors generated from attention to different encoders. Figure 3 shows the rescoring model architecture for attention to both audio and lattice encoders.

3. EXPERIMENTAL SETUP

3.1. Datasets and Architecture Details

We used de-identified internal speech data containing a mix of command and control and free-form utterances from voice controlled far-field devices for our experiments. We experimented with two different first-pass models: a HMM-based hybrid ASR model and an end-to-end RNN-T model. The hybrid ASR system consits of a LSTM-based acoustic model trained with CTC loss criterion [25] and a 4-gram LM smoothed with Kneser-Ney [26]. Unless specified, hybrid ASR model should be assumed as the first-pass model. Lattices are generated from beam decoding [27] of the first-pass models. Apart from the original lattices (referred as *full-lattices*) produced by beam decoding, we also considered pruned *2-best* and *5-best* lattices [28].

The rescoring models contain two LSTM layers with 256 units in the decoder and one LSTM (or LatticeLSTM) layer with 256 units for 1-best/n-best/lattice encoders. For audio encoder, audio features extracted from first-pass ASR are passed to a LSTM layer of 768 units. Both encoder and decoder use an embedding layer with 256 units and vocabulary size of 50k. We use multi-head attention with 4 heads and in case of dual encoders, we use 2 attention heads for each encoder. We also trained a baseline LSTM-based LM with same set of parameters as the decoder of the attention-based rescoring models. The training data used for rescoring models is roughly a quarter of the data used to train first-pass models.

Table 1: Relative WERR over first-pass hybrid ASR model after n-best rescoring using lattice-attention models incorporating different weighting mechanisms in the lattice encoder. The models are trained and evaluated on *full-lattice* input.

Weighting Mechanism	WERR (%)
None (TreeLSTM)	-8.4
WCS	2.4
BFG	3.8
BATT	5.0
WEO	4.9
WCS + BFG + BATT	5.0
WCS + BFG + WEO	5.0

3.2. Training and Evaluation

All the rescoring models, including LSTM-LM, were first trained with MLE loss and then finetuned with MWER loss. Recent works on attention-based rescoring models [5, 8] have shown that an additional training with MWER criterion on top of maximum-likelihood training improves accuracy. For evaluation, we run the decoder in the teacher-forcing mode and apply log-linear interpolation of scores of first-pass and rescoring models. The interpolation weight is tuned on a development set. Top 5 hypotheses from the first-pass models are used for n-best rescoring. We evaluate all the experiments on an internal test set containing about 140 hours of audio. We report the results in terms of relative WERR with respect to WER of the first-pass model. Both hybrid and RNN-T baseline first-pass models have absolute WER below 10%.

4. RESULTS

4.1. Weighting mechanisms in LatticeLSTM

Table 1 shows the effect of the four weighting mechanisms of LatticeLSTM introduced in Section 2.4.2. We used *full-lattice* as encoder input for these experiments. A large degradation over first-pass is observed with unweighted TreeLSTM lattice encoder (first row in Table 1). We attribute this to absence of any biasing of the most probable paths in the lattice, causing confusion in the attention module. This can be seen for an example in Figure 4, where attention weights for TreeLSTM and LatticeLSTM (WCS + BFG + WEO) models are shown for an example lattice corresponding to a confusing token in the decoder. For TreeLSTM model, attention weights are almost uniformly distributed across the homophone variants for one of the attention heads with slightly higher score for the token *"family"*, a common token in the training data. Whereas, for LatticeLSTM model, tokens associated with larger marginal and backward-normalized weights tend to have higher attention weights even if the tokens are rare.

Introducing any of the weighting mechanisms reverses the degradation observed with TreeLSTM. Weighting mechanisms based on marginal weights (i.e. WEO and BATT), have a larger impact than the mechanisms incorporating backward-normalized

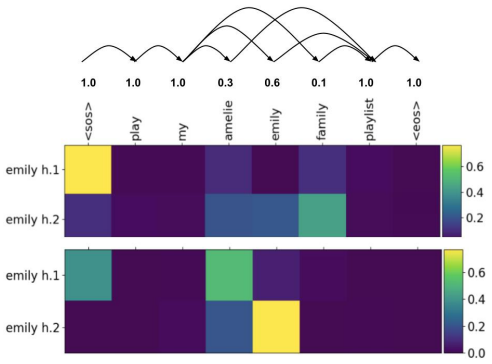

Fig. 4: This figure shows attention weights for TreeLSTM (on the top) and weighted LatticeLSTM (on the bottom) models for an example lattice with ground truth as *"play my emily playlist"*. The attention weights correspond to the decoder token *"emily"* in the 1-best hypothesis for two different attention heads in the y-axis. The x-axis shows the linearized lattice nodes fed to the encoder and marginal weights for the nodes are shown on top.

Table 2: Relative WERR (%) over first-pass hybrid ASR model after n-best rescoring using lattice-attention models trained on lattices of different depths. We report WERR on full test set and utterances with $\leq$2 and $>$2 alternate hypotheses in the original lattices.

Lattice Encoder	Full test set	Utts with $\leq$2 hyps	Utts with $>$2 hyps
2-best lattice	4.6	3.8	4.8
5-best lattice	4.8	3.8	5.1
Full-lattice	5.0	3.8	5.5

weights. From here on, lattice encoders should be assumed to have WCS, BFG and WEO weighting mechanisms.

4.2. Impact of lattice depth

Table 2 shows relative WERR over first-pass WER for rescoring models trained on lattices with varying depths. Note that for a *n-best* lattice, the original lattice is pruned to top *n* hypotheses in both training and evaluation. *2-best* lattice as the encoder input results in 4.6% WERR and *5-best* or *full* lattice inputs provide very small further improvements. This is due to most lattices being shallow as only 31% of the lattice have more than two alternate hypotheses in our data. In Table 2, we also report results on partitions of the test set with $\leq$2 and $>$2 alternate hypotheses in the original lattice. All three models have similar performance on utterances with $\leq$2 alternatives but on utterances with $>$2 alternate hypotheses, models with lattices of higher depths as encoder input perform better. This suggests that models leveraging more alternative hypotheses from the lattice can benefit from the richness of the lattice.

4.3. Encoder type

Table 3 captures relative WERR of rescoring models with different encoder types over first-pass WER. Except LSTM-LM, all the models are attention-based and trained spearately on audio features/1-best/n-best/lattice outputs of hybrid and RNN-T first-pass models. The LSTM-LM rescoring model is same for both hybrid and RNN-T as it is trained only on transcriptions. Our results show that attention to any of the encoder provides more improvement compared to discriminatively trained LSTM-LM. The consistently smaller improvement for RNN-T system compared to hybrid is due to stronger first-pass model, evident from 1.2% WERR for RNN-T compared to 2.9% WERR for hybrid when rescored with same LSTM-LM model. The lattice encoder performs better compared to 1-best or n-best encoders and provides similar WERR compared to audio encoder, agnostic of the first-pass model. In situations where the audio is not available for second-pass rescoring, lattice encoders could be a good proxy. The audio-attention model can be seen as LAS rescoring proposed in [5] and audio & n-best attention can be seen as deliberation network rescoring of [10] without the joint training of first-pass and second-pass models. The LatticeLSTM-based *5-best* lattice encoder outperforms deliberation-style *5-best* encoder of [10] for both single and dual encoder setups. Also, due to compactness of lattice representation, there are 52% fewer forward-passes on average in LatticeLSTM encoder compated to deliberation network. Adding attention to additional 1-best, n-best or lattice encoder provides further WERR over audio-only attention model. The model with attention to both, audio and *full* lattice, achieves the best result with 7.8% relative WERR over first-pass for hybrid and 5.7% for RNN-T. The small difference between *5-best* or *full* lattice encoders can be attributed to very small number of lattices with more than five alternatives as discussed in Section 4.2.

Table 3: Relative WERR (%) over first-pass models (Hybrid and RNN-T) after n-best rescoring using different models.

Rescoring Model	Hybrid	RNN-T
No encoder (LSTM-LM)	2.9	1.2
1-best encoder	3.8	2.7
5-best deliberation encoder	4.1	3.1
5-best lattice encoder	4.8	3.6
Full-lattice encoder	5.0	3.8
Audio encoder (LAS)	5.2	3.7
Audio & 1-best encoders	6.5	4.4
Audio & 5-best deliberation encoders	6.9	4.8
Audio & 5-best lattice encoders	7.6	5.5
Audio & Full-lattice encoders	7.8	5.7
Oracle at 5-best	42.9	36.1

5. CONCLUSIONS

We proposed an attention encoder-decoder model with attention to lattices for rescoring n-best hypotheses generated by an ASR model. The lattice-attention rescoring model achieves 4-5% relative WERR over hybrid or RNN-T first-pass models, which is comparable to performance of audio-only attention model. Attention to both, audio and lattices, brings further improvement, resulting in 6-8% WERR. We showed that LatticeLSTM-based lattice encoder excels over n-best encoder respresentation of deliberation network. Further, deeper lattices can benefit from richness of the lattice. As opposed to the attention biasing method, we proposed a simpler alternative with similar performance, in which encoder outputs are scaled in proportion to lattice weights while keeping the usual attention mechanism unchanged. We also looked into different weighting mechanisms in the lattice encoder and showed that incorporating weights in the lattice encoder is essentital for attention-based models to avoid inherent confusions in the lattices arising from conflicting arcs.

6. REFERENCES

[1] Hermann Ney and Xavier Aubert, "A word graph algorithm for large vocabulary, continuous speech recognition," in *CSLP*, 1994.

[2] Richard Schwartz and Steve Austin, "A comparison of several approximate algorithms for finding multiple (n-best) sentence hypotheses," in *ICASSP*, 1991, pp. 701–704.

[3] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur, "Recurrent neural network based language model," in *Eleventh annual conference of the international speech communication association*, 2010.

[4] Martin Sundermeyer, Ralf Schlüter, and Hermann Ney, "Lstm neural networks for language modeling," in *Thirteenth annual conference of the international speech communication association*, 2012.

[5] Tara N Sainath, Ruoming Pang, David Rybach, Yanzhang He, Rohit Prabhavalkar, Wei Li, Mirkó Visontai, Qiao Liang, Trevor Strohman, Yonghui Wu, et al., "Two-pass end-to-end speech recognition," *Proc. Interspeech 2019*, pp. 2773–2777, 2019.

[6] William Chan, Navdeep Jaitly, Quoc Le, and Oriol Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *ICASSP*, 2016, pp. 4960–4964.

[7] Alex Graves, "Sequence transduction with recurrent neural networks," *ICML 2012*, 2012.

[8] Ankur Gandhe and Ariya Rastrow, "Audio-attention discriminative language model for asr rescoring," in *ICASSP*, 2020, pp. 7944–7948.

[9] Yingce Xia, Fei Tian, Lijun Wu, Jianxin Lin, Tao Qin, Nenghai Yu, and Tie-Yan Liu, "Deliberation networks: Sequence generation beyond one-pass decoding," in *NeurIPS*, 2017, pp. 1782–1792.

[10] Ke Hu, Tara N Sainath, Ruoming Pang, and Rohit Prabhavalkar, "Deliberation model based two-pass end-to-end speech recognition," in *ICASSP*, 2020, pp. 7799–7803.

[11] Matthias Sperber, Graham Neubig, Jan Niehues, and Alex Waibel, "Neural lattice-to-sequence models for uncertain inputs," in *EMNLP*, 2017, pp. 1380–1389.

[12] Jinsong Su, Zhixing Tan, Deyi Xiong, Rongrong Ji, Xiaodong Shi, and Yang Liu, "Lattice-based recurrent neural network encoders for neural machine translation," in *AAAI*, 2017, vol. 31.

[13] Matthias Sperber, Graham Neubig, Ngoc-Quan Pham, and Alex Waibel, "Self-attentional models for lattice inputs," in *ACL*, 2019, pp. 1185–1197.

[14] Fengshun Xiao, Jiangtong Li, Hai Zhao, Rui Wang, and Kehai Chen, "Lattice-based transformer encoder for neural machine translation," in *ACL*, 2019, pp. 3090–3097.

[15] Faisal Ladhak, Ankur Gandhe, Markus Dreyer, Lambert Mathias, Ariya Rastrow, and Björn Hoffmeister, "Latticernn: Recurrent neural networks over lattices.," in *Interspeech*, 2016, pp. 695–699.

[16] Yue Zhang and Jie Yang, "Chinese ner using lattice lstm," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2018, pp. 1554–1564.

[17] Xunying Liu, Yongqiang Wang, Xie Chen, Mark JF Gales, and Philip C Woodland, "Efficient lattice rescoring using recurrent neural network language models," in *ICASSP*, 2014, pp. 4908–4912.

[18] Hainan Xu, Tongfei Chen, Dongji Gao, Yiming Wang, Ke Li, Nagendra Goel, Yishay Carmiel, Daniel Povey, and Sanjeev Khudanpur, "A pruned rnnlm lattice-rescoring algorithm for automatic speech recognition," in *ICASSP*, 2018, pp. 5929–5933.

[19] Takaaki Hori, Chiori Hori, Shinji Watanabe, and John R Hershey, "Minimum word error training of long short-term memory recurrent neural network language models for speech recognition," in *ICASSP*, 2016, pp. 5990–5994.

[20] Kai Sheng Tai, Richard Socher, and Christopher D Manning, "Improved semantic representations from tree-structured long short-term memory networks," in *ACL*, 2015, pp. 1556–1566.

[21] Cyril Allauzen, Michael Riley, Johan Schalkwyk, Wojciech Skut, and Mehryar Mohri, "Openfst: A general and efficient weighted finite-state transducer library," in *International Conference on Implementation and Application of Automata*. Springer, 2007, pp. 11–23.

[22] Robert L Hemminger, "Line graphs and line digraphs," *Selected topics in graph theory*, 1983.

[23] Lawrence R Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[24] Dzmitry Bahdanau, Kyung Hyun Cho, and Yoshua Bengio, "Neural machine translation by jointly learning to align and translate," in *ICLR*, 2015.

[25] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *ICML*, 2006, pp. 369–376.

[26] R. Kneser and H. Ney, "Improved backing-off for m-gram language modeling," in *ICASSP*, 1995, vol. 1, pp. 181–184 vol.1.

[27] Daniel Povey, Mirko Hannemann, Gilles Boulianne, Lukáš Burget, Arnab Ghoshal, Miloš Janda, Martin Karafiát, Stefan Kombrink, Petr Motlíček, Yanmin Qian, Korbinian Riedhammer, Karel Veselý, and Ngoc Thang Vu, "Generating exact lattices in the wfst framework," in *ICASSP*, 2012, pp. 4213–4216.

[28] 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," `https://kaldi-asr.org/doc/nbest-to-lattice_8cc.html`, 2011.