INTERSPEECH 2021
30 August – 3 September, 2021, Brno, Czechia

LT-LM: a novel non-autoregressive language model for single-shot lattice rescoring

Anton Mitrofanov[1,2], *Mariya Korenevskaya*[2], *Ivan Podluzhny*[1,2], *Yuri Khokhlov*[2], *Aleksandr Laptev*[1], *Andrei Andrusenko*[1], *Aleksei Ilin*[2], *Maxim Korenevsky*[2], *Ivan Medennikov*[1,2], *Aleksei Romanenko* [1,2]

[1]ITMO University, St. Petersburg, Russia
[2]STC-innovations Ltd, St. Petersburg, Russia

{mitrofanov-aa, korenevskaya, podluzhnyi, khokhlov}@speechpro.com, aalaptev@itmo.ru, {andrusenko, ilin-a, korenevsky, medennikov, romanenko}@speechpro.com

Abstract

Neural network-based language models are commonly used in rescoring approaches to improve the quality of modern automatic speech recognition (ASR) systems. Most of the existing methods are computationally expensive since they use autoregressive language models. We propose a novel rescoring approach, which processes the entire lattice in a single call to the model. The key feature of our rescoring policy is a novel non-autoregressive Lattice Transformer Language Model (LT-LM). This model takes the whole lattice as an input and predicts a new language score for each arc. Additionally, we propose the artificial lattices generation approach to incorporate a large amount of text data in the LT-LM training process. Our single-shot rescoring performs orders of magnitude faster than other rescoring methods in our experiments. It is more than 300 times faster than pruned RNNLM lattice rescoring and N-best rescoring while slightly inferior in terms of WER.

Index Terms: speech recognition, lattice transformer, language models, artificial lattices, lattice rescoring, LT-LM

1. Introduction

Language models (LMs) are a significant part of automatic speech recognition (ASR) systems. They can reduce the overall system error rate [1–4] and make the recognition results more human-readable [5]. Back-off n-gram LMs [6] are traditionally used in hybrid ASR systems [7], but the Neural Network Language Models (NNLMs) outperform n-gram ones [8–13] due to their ability to capture longer context and extract more meaningful information from text. However, NNLMs are computationally expensive, and their usage in ASR systems is challenging. The most popular way to apply them is the two-pass method [14]. First, decoding pass is performed with an n-gram LM. Then, a NNLM can be additionally used to rescore [10] the results of the first pass.

There are several widely used rescoring approaches. The simplest one is the N-best list rescoring [15]. It is applied separately to each hypothesis of the first pass, ranking them with respect to updated scores. N-best rescoring provides better Word Error Rate (WER) reduction when it operates with a larger number of hypotheses N to re-rank. Usually, hypotheses differ from each other only in a small number of words. Since N-best rescoring does not take this fact into account, it requires a lot of unnecessary computations, which makes this method very computationally expensive. This flaw of N-best rescoring is mitigated when using rescoring approaches dealing with word lattices. There are approaches such as pruned RNNLM lattice rescoring [16], fast N-best rescoring [17] and parallelizable lattice rescoring [18] which exploit the similarity of hypotheses to optimize computations. Nevertheless, these rescoring approaches are rather slow and computationally expensive as they require multiple LM calls.

In this article, we introduce a new non-autoregressive language model that processes the whole lattice in a single shot during the rescoring process. It is trained directly on ASR lattices, unlike conventional LMs trained on text sequences. We used the Transformer architecture [19] for this purpose because it can be easily adapted for training on lattices [20] and named this special model Lattice Transformer Language Model (LT-LM). This model takes a whole lattice as input and outputs a language weight for each arc. Using such a model in a rescoring process makes it possible to re-rank the lattice hypotheses in a single call to the model. This makes the rescoring process significantly faster and reduces the number of computational resources compared to rescoring with autoregressive models.

Autoregressive LMs are usually trained on large amounts of text data, which helps them to provide significant WER reduction in the rescoring task. Since LT-LM is supposed to be trained on ASR results and corresponding speech transcripts, it is challenging to obtain enough training data. To cope with the lack of training data, we developed a method of artificial lattices generation from large amounts of texts, using just a small subset of labeled audio data (about 1-2 hours). This made it possible to significantly increase the amount of training data for LT-LM and obtain rescoring results close to ones corresponding to modern autoregressive LMs.

2. Rescoring Task

Normally, the decoding process searches for the best ASR hypothesis according to the Maximum A Posteriori (MAP) rule:

$$W^* = \underset{W}{\operatorname{argmax}}\, p(X|W)^a P(W)^l, \tag{1}$$

where W stands for a recognition hypothesis, W^* is the best hypothesis, a and l are adjustable acoustic and language weights, respectively, while $p(X|W)$ is an acoustic score, and $P(W)$ is a language score.

The process of searching for the best hypothesis requires applying the language model to all possible word sequences to estimate their language probabilities. This process is fast when n-gram LMs are used. Typically, NNLMs work much slower but significantly outperform n-gram ones in terms of WER. Thus, it is necessary to reduce the number of recognition hypotheses to apply neural network-based LMs. It can be done with a two-pass approach. First, decoding with an n-gram LM is performed, and a list of hypotheses with the best scores is saved. The result of the first pass can be compactly represented as a weighted directed acyclic graph called a lattice. Lattice arcs are usually supplied with word labels, as well as language and acoustic scores. Lattice always has a single initial state and at least one final state.

The second part of the two-pass method is the rescoring process. It updates language scores using neural network-based LM and reorders hypotheses. Generally, after such a procedure, the best hypothesis changes to the one containing fewer errors. The score of the best hypothesis after rescoring can be evaluated as follows:

$$W^* = \underset{W \in \text{lattice}}{\operatorname{argmax}} P(X|W)^a P_1(W)^{l_1} P_2(W)^{l_2}, \qquad (2)$$

where $P_1(W)$ and $P_2(W)$ are the probabilities of n-gram LM and rescoring LM, respectively, and l_1, l_2 are corresponding LM weights.

The language model estimates a probability of a given sequence of words. However, the rescoring process operates only with the most probable hypotheses of the first pass. In order to take this fact into account, we propose a special LM, which estimates the language scores directly for a set of lattice hypotheses. We named it Lattice Transformer Language Model and trained it on lattices instead of text data.

3. Lattice Transformer Language Model

While the text is a sequence of words, a lattice is a non-linear structure. This means that the order of words in a lattice is determined by lattice topology. There were several attempts to train models using lattices as their inputs [20–23]. We adapted the Transformer architecture for this purpose. Such adaptation is possible due to the self-attention component of the Transformer architecture. This module does not depend on the order of its elements because it computes a weighted sum of the input sequence. Transformer uses positional encoding to introduce the information about the sequence order into its latent representations. The following subsection describes the proposed approach of lattice topology encoding for use in Transformer.

3.1. Positional Encoding

One of the most common ways to encode positional information is combining embeddings of the input elements with the so-called positional embeddings [24]. Positional embeddings are usually obtained from the embedding matrix trained together with the model. Each embedding corresponds to a specific position in the input sequence. This encoding method can be easily adapted for training on data structures different from sequences (e.g., lattices).

We propose the following pipeline for positional encoding of lattices. The first step is representing the lattice as a set of its arcs. Then, we propose adding several auxiliary arcs to keep information about the initial and final states of a lattice. They serve the same purpose as the SOS and EOS tokens in the case of text-based learning. An auxiliary arc with the symbol <s> comes to the initial state, and auxiliary arcs with the symbol </s> come from each final state. Here symbols <s> and </s> correspond to the beginning and the end of the sentence, respectively. The next step is the topological sort of lattice to ensure the proper order of its states.

The position of the arc (x, y) in the lattice is determined by the indices of its source and destination states x and y. Thus, it is possible to encode the topological information of each arc in the lattice using the sum of positional embeddings of its source and destination states, respectively. Two embedding matrices for source and destination states are trained separately to account for arcs direction. Finally, the total arc embedding is obtained by adding its positional embedding to a corresponding word embedding.

3.2. Targets and loss

The hypothesis with the least WER in the lattice is called the oracle hypothesis. The ideal rescoring would provide this hypothesis as a result. For this reason, LT-LM is trained to find the oracle hypothesis in the lattice. This is done in the following way. First, the oracle path is obtained from each lattice prepared as described above. If there are several oracle paths in lattice, the target path is chosen randomly. Then lattice arcs are split into two classes. All arcs comprising the oracle path are assigned to class 1, and the others — to class 0. LT-LM is trained to minimize the binary cross-entropy for each arc.

3.3. Single-shot Rescoring with LT-LM

The LT-LM key feature is the ability to generate probabilities for all arcs in the lattice in a single call to the model. LT-LM predicts the probability for each arc of the lattice to belong to the oracle path. These probabilities can be used in combination with original LM and AM scores to generate the final rescoring result. Then, the best hypothesis is chosen using the standard forward algorithm accordingly to (2). The described method is significantly faster than other rescoring methods requiring several calls to the model. This fact is discussed in detail in subsection 5.4.

4. Lattices generation from text

As mentioned above, recognition lattices and ground-truth word-level transcriptions are needed for LT-LM training. At the same time, classic LMs are trained in an unsupervised manner using large amounts of text data, which are much easier to collect. In order to utilize large amounts of available text data for LT-LM training, we propose an algorithm that generates realistic artificial lattices directly from the text.

It is necessary to imitate the acoustic similarity of words to generate realistic lattices from the text. We propose the following approach for generating lattices from text data as close as possible to real ones:

- simulate a per-frame alignment from text without using audio data;
- simulate a probability distribution over acoustic classes for each frame of the alignment;
- decode the resulting sequence with a beam search algorithm to generate lattices.

We used an acoustic model and a small labeled audio dataset to generate lattices. We estimated the per-frame distribution of acoustic classes of this dataset and built a per-frame alignment. For each acoustic class, we estimated the distribution of its duration in the alignment. We evaluated this distribution from the alignment and calculated the average acoustic model output for each acoustic class. Finally, we obtained a matrix of the shape (A, A), where A stands for the number of acoustic classes, and referred it to as the Fake Acoustic Model (FAM). The i-th row in this matrix contains the average distribution of acoustic classes over all frames where the alignment value is equal to i.

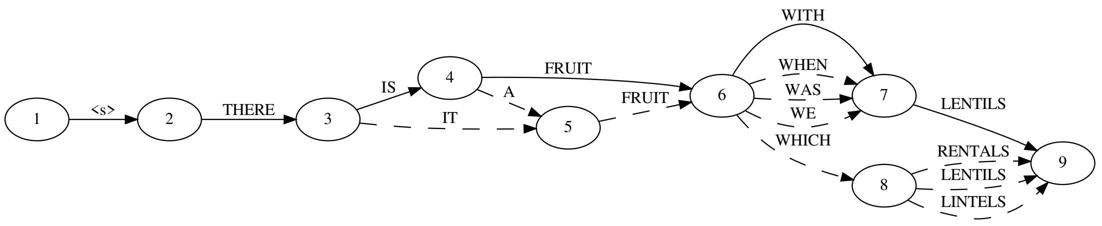

Figure 1: *A beginning part of an artificial lattice. The whole lattice contains 53 arcs, while the reference text contains 22 words.*

Alignment can be obtained from the text as follows. Similar to the forced alignment pipeline, each sentence is converted to a graph containing all possible representations of the corresponding word sequence in the form of acoustic classes. If there is an audio corresponding to the text, the most probable alignment can be obtained using the Viterbi algorithm. In contrast, we propose to sample a random path from the alignment graph. This path should not contain the same acoustic classes on adjacent arcs. Unlike the real alignment, the obtained paths are time-agnostic. Thus, it is necessary to introduce the time component and "stretch" this path to simulate realistic alignment. We sampled a random number N from the estimated distribution for each acoustic class and stretched this class by N frames. We referred to the resulting sequence of acoustic classes as the Fake Alignment (FAli).

Then, artificial lattices can be generated by performing the conventional decoding procedure on FAli. The probability distribution of acoustic classes for each frame is sampled from the FAM model. Finally, the standard decoding procedure can be applied to generate artificial lattices.

Figure 1 represents a part of generated lattice for an utterance from *dev_clean* subset of LibriSpeech corpus. Solid lines mark the path corresponding to the oracle hypothesis, while dashed ones indicate arcs of alternative hypotheses added during the lattice generation. The structure of a simulated lattice is similar to the structure of a real one. The words from alternative hypotheses of generated lattice sound similarly to corresponding ones from the oracle path.

5. Experiments

5.1. Baselines

Our experiments with LT-LM were performed using the LibriSpeech corpus [25], which contains 960 hours of English audiobooks for AM training and 5Gb of extra text data for LM training. The Kaldi [26] recipe for LibriSpeech[1] was used for the baseline AM training. The baseline WER results were obtained using the large 3-gram LM from this recipe. As rescoring baselines, we used a Transformer-LM[2] as well as LSTM-LM[3] from ESPNet framework [27]. Transformer-LM consists of 16 transformer blocks. Each block has 8 attention heads of dimensions 512 and 2048 units in feed-forward layers. LSTM-LM has 4 layers of 2048 neurons each. The target texts were tokenized into 5000 tokens by Byte Pair Encoding (BPE) algorithm [28] from the SentencePiece tokenization toolkit [29]. The total number of parameters of LSTM and Transformer LMs are 155M and 54M, respectively. We considered two rescoring approaches as baselines: N-best rescoring and pruned RNNLM lattice rescoring [16]. The last one was performed with 4-gram approximation and the beam of 6.

[1]kaldi/egs/librispeech/s5/local/chain/tuning/run_tdnn_1d.sh
[2]espnet/egs2/librispeech/asr1/conf/tuning/train_lm_transformer2.yaml
[3]espnet/egs2/librispeech/asr1/conf/tuning/train_lm_adam.yaml

5.2. LT-LM training

Initially, we decoded the entire LibriSpeech training set to obtain lattices for the LT-LM training. The model trained using only these lattices did not show the competitive rescoring results due to the following reasons. First, the obtained amount of data was insufficient for the LT-LM training and led to model overfitting. Second, the word diversity of the training utterances texts did not cover the vocabulary of the baseline n-gram LM trained with additional text data. We used a small subset of training data of 2 hours and additional LibriSpeech texts for the artificial lattices generation to overcome these problems. Decoding is the most computationally intensive part of the generation process. Due to the resource limitation, we accelerated the decoding by reducing the beam from 16 to 13 and the number of active hypotheses from 5000 to 3000. Finally, we pruned lattices with beam 4 to speed up the training process. LT-LM was trained on word lattices, and the overall vocabulary consisted of 200K words.

We used the Fairseq framework [30] for LT-LM training. The model had 8 transformer encoder layers, each of 8 attention heads with 816 embedded dimensions and 2048 feed-forward units. The total number of the model parameters was 211M. LT-LM was trained using Adam optimizer with a linear learning rate warmup for the first 4000 iterations and with a constant learning rate of 3e-5 after warmup. The use of gradient checkpointing [31] significantly reduced the memory usage and allowed us to increase the batch size from 4 to 64. Our best LT-LM was trained for six epochs on a combination of real and artificial lattices.

Table 1: *Rescoring results in WER on LibriSpeech development and test sets for different NNLMs and rescoring methods. "Pruned" stands for pruned RNNLM lattice rescoring.*

LM	method	dev		test	
		clean	other	clean	other
lattices pruned with beam 8					
3-gram	-	3.41	9.26	3.90	9.25
LSTM	50-best	2.78	7.51	2.84	7.67
LSTM	500-best	2.56	7.55	2.86	7.67
LSTM	Pruned	2.63	7.10	2.95	7.38
Transformer	50-best	2.47	8.10	2.82	8.18
Transformer	500-best	**2.26**	**7.06**	**2.63**	**7.23**
lattices pruned with beam 4					
LSTM	50-best	2.88	8.61	3.14	8.69
LSTM	500-best	2.74	7.89	3.05	8.05
Transformer	50-best	2.61	8.28	2.99	8.37
Transformer	500-best	**2.50**	7.51	**2.84**	**7.67**
LT-LM	Single-Shot	2.51	**7.47**	3.01	7.74

Table 2: *Rescoring GPU running time in minutes and seconds*

LM	method	dev		test	
		clean	other	clean	other
lattices pruned with beam 8					
LSTM	50-best	12m25s	13m8s	12m32s	13m21s
LSTM	500-best	48m22s	59m26s	49m5s	60m39s
LSTM	Pruned	13m47s	34m	14m23s	35m38s
Transformer	50-best	10m25s	11m48s	10m25s	11m59s
Transformer	500-best	41m27s	55m42s	39m8s	54m47s
lattices pruned with beam 4					
LSTM	50-best	9m15s	10m27s	9m6s	11m6s
LSTM	500-best	19m13s	32m	17m58s	32m34s
Transformer	50-best	8m11s	9m43s	7m59s	9m43s
Transformer	500-best	15m41s	27m55s	15m33s	28m44s
LT-LM	Single-Shot	**3.8s**	**4.5s**	**3.7s**	**4.6s**

5.3. Rescoring WER performance

The rescoring results for all models mentioned above are presented in Table 1. The first row of the table demonstrates the rescoring results of the best 3-gram LM from the LibriSpeech Kaldi recipe. The next three rows correspond to the LSTM-LM results: 50-best and 500-best rescoring and pruned RNNLM lattice rescoring [16], respectively. The next two rows correspond to 50-best and 500-best rescoring with Transformer-LM and the latest one shows the best WER results on LibriSpeech development and test sets. The NNLM weight of 0.8 was used for all the described approaches. Since LT-LM was trained on lattices pruned with a beam of 4, we used the same beam value in test time to avoid the mismatch between training and evaluation. Also, we provide the results of N-best rescoring with a beam equal to 4. These results are demonstrated in the second part of Table 1 and reflect the influence of beam value on the rescoring quality. The single-shot approach with LT-LM shows the quality comparable to N-best rescoring with LSTM-LM.

5.4. Rescoring speedup

Rescoring was performed on a single Nvidia GeForce RTX 2080 ti GPU. We measured the GPU running time of different rescoring approaches on the LibriSpeech development and test sets to compare their speed performances. The GPU running time results for all rescoring approaches except 500-best were averaged over ten launches. The results for 500-best rescoring were averaged over only two launches due to high computational cost per run. The rescoring for both LT-LM and N-best lists was performed with a batch size equal to 64. The results are presented in Table 2. It can be seen that the single-shot rescoring with LT-LM was orders of magnitude faster than the other methods. For example, LT-LM rescoring is about 350 times faster than pruned RNNLM lattice rescoring, while having comparable quality. At the same time, Transformer-LM 500-best rescoring outperforms LT-LM rescoring for a maximum of 0.5% WER on test_other, but LT-LM performs more than 630 times faster! The reasons for such acceleration can be explained as follows.

The main advantage of LT-LM is that the rescoring of the entire lattice is performed in a single-shot manner. This leads to the significant speedup of the rescoring process. Table 3 demonstrates the statistics obtained for the LibriSpeech *dev_clean* and *dev_other* sets. The left half of the table shows the average length of sequences processed for a single LM call, while the right half shows the number of such calls.

In the case of N-best rescoring, the length of the processed sequence equals the number of words in the recognition hypothesis. In Table 3 we provide two types of average sequence lengths. Micro-average sequence length for N-best rescoring is evaluated over all hypotheses. To compute macro-average, we evaluated an average hypothesis length for each lattice independently and then averaged the results. For the pruned RNNLM lattice rescoring, the length of the processed word sequence always equals one, because the algorithm processes arcs in the lattice one by one. Finally, since LT-LM processes the whole lattice, the corresponding word sequence length during the rescoring process is equal to an overall number of arcs in the lattice. Respectively, the number of LM calls is equal to the total amount of processed hypotheses for the N-best rescoring, to the total amount of arcs involved in rescoring in the composed lattice for the pruned RNNLM rescoring, and to the number of lattices for the LT-LM one.

Table 3: *Micro-average (macro-average) word sequence lengths and numbers of model calls for different rescoring methods on LibriSpeech dev_clean and dev_other sets. For two last rows these types of average are equivalent.*

method	av. seq. len.		num. of model calls	
	clean	other	clean	other
50-best	23 (20)	19 (18)	105K	127K
500-best	27 (20)	21 (18)	795K	1.1M
Pruned	1	1	196K	456K
Single-Shot	26	33	2703	2864

At first glance, it looks surprising that the average amount of arcs in lattices is close to the micro-average N-best hypothesis length. The matter of things is as follows. The N-best list for the short utterances usually contains much less than N hypotheses. These hypotheses are mostly rather short, and the number of arcs in such lattices is only slightly larger than the hypotheses lengths. On the other hand, for long utterances, the number of arcs in lattices is usually larger than the hypotheses length by a moderate factor, but the N-best lists corresponding to such lattices contain a lot of long hypotheses to be rescored. As a result, the micro-average length of processed hypotheses is biased towards the lengths of long hypotheses from large lattices. However, an average number of arcs in lattices as well as macro-average hypothesis length are not subject to this bias.

According to Table 3, there is no significant difference in the average sequence lengths for LT-LM and N-best rescoring methods. However, the number of model calls to process for LT-LM is several orders of magnitude less than for all other rescoring methods presented in the table.

6. Conclusion and future work

We presented a novel single-shot approach to rescoring of ASR results. This approach allows to rescore the entire lattice in a single LM call. While comparable in quality to other popular rescoring approaches, the proposed one is faster by orders of magnitude. The proposed approach utilizes a special Lattice Transformer Language Model, which deals with ASR results presented as lattices.

In the future, we plan to adapt sequence-discriminative losses for LT-LM training. We believe that they can improve the LT-LM quality and help it to outperform autoregressive LMs in terms of WER. It is also interesting to apply LT-LM on far-field speech datasets, which are more challenging for ASR.

7. Acknowledgements

This work was partially financially supported by ITMO University.

8. References

[1] T. Hori, C. Hori, Y. Minami, and A. Nakamura, "Efficient WFST-based one-pass decoding with on-the-fly hypothesis rescoring in extremely large vocabulary continuous speech recognition," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, no. 4, pp. 1352–1365, 2007.

[2] K. J. Han, A. Chandrashekaran, J. Kim, and I. Lane, "The CAPIO 2017 conversational speech recognition system," *ArXiv:1801.00059*, 2018.

[3] C. Luscher, E. Beck, K. Irie, M. Kitza, W. Michel, A. Zeyer, R. Schluter, and H. Ney, "RWTH ASR systems for LibriSpeech: hybrid vs attention," in *INTERSPEECH*, 2019, pp. 231–235.

[4] S. Toshniwal, A. Kannan, C.-C. Chiu, Y. Wu, T. Sainath, and K. Livescu, "A comparison of techniques for language model integration in encoder-decoder speech recognition," in *Spoken Language Technology Workshop (SLT)*. IEEE, 2018, pp. 369–375.

[5] J. Liao, Y. Shi, M. Gong, L. Shou, S. Eskimez, L. Lu, H. Qu, and M. Zeng, "Generating human readable transcript for automatic speech recognition with pre-trained language model," *ArXiv:2102.11114*, 2021.

[6] J. Goodman, "A bit of progress in language modeling," *Computer Speech & Language*, vol. 15, pp. 403–434, 2001.

[7] M. Mohri, F. Pereira, and M. Riley, "Weighted finite-state transducers in speech recognition," *Computer Speech & Language*, vol. 16, no. 1, pp. 69–88, 2002.

[8] T. Mikolov, M. Karafiát, L. Burget, J. Cernocký, and S. Khudanpur, "Recurrent neural network based language model," in *INTERSPEECH*, 2010.

[9] I. Medennikov, A. Prudnikov, and A. Zatvornitskiy, "Improving english conversational telephone speech recognition," in *INTERSPEECH*, 2016, pp. 2–6.

[10] 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, 2016.

[11] I. Medennikov, M. Korenevsky, T. Prisyach, Y. Khokhlov, M. Korenevskaya, I. Sorokin, T. Timofeeva, A. Mitrofanov, A. Andrusenko, I. Podluzhny, A. Laptev, and A. Romanenko, "The STC system for the CHiME-6 challenge," in *CHiME 2020 Workshop on Speech Processing in Everyday Environments*, 2020.

[12] I. Medennikov, Y. Y. Khokhlov, A. Romanenko, I. Sorokin, A. Mitrofanov, V. Bataev, A. Andrusenko, T. Prisyach, M. Korenevskaya, O. Petrov, and A. Zatvornitsky, "The STC ASR system for the VOiCES from a distance challenge 2019," in *INTERSPEECH*, 2019, pp. 2453–2457.

[13] G. Saon, G. Kurata, T. Sercu, K. Audhkhasi, S. Thomas, D. Dimitriadis, X. Cui, B. Ramabhadran, M. Picheny, L.-L. Lim, B. Roomi, and P. Hall, "English conversational telephone speech recognition by humans and machines," in *INTERSPEECH*, 2017, pp. 132–136.

[14] M. Sundermeyer, Z. Tüske, R. Schlüter, and H. Ney, "Lattice decoding and rescoring with long-span neural network language models," in *INTERSPEECH*, 2014.

[15] A. Deoras, T. Mikolov, and K. Church, "A fast re-scoring strategy to capture long-distance dependencies," in *Empirical Methods in Natural Language Processing (EMNLP)*, Jul. 2011, pp. 1116–1127.

[16] X. Hainan, C. Tongfei, G. Dongji, W. Yiming, K. Li, N. Goel, Y. Carmiel, D. Povey, and S. Khudanpur, "A pruned RNNLM lattice-rescoring algorithm for automatic speech recognition," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 5929–5933.

[17] E. Beck, R. Schluter, and H. Ney, "LVCSR with transformer language models," in *INTERSPEECH*, 2020, pp. 1798–1802.

[18] K. Li, D. Povey, and S. Khudanpur, "A parallelizable lattice rescoring strategy with neural language models," *arXiv:2103.05081*, 2021.

[19] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems (NIPS)*, vol. 30, 2017.

[20] M. Sperber, G. Neubig, N.-Q. Pham, and A. Waibel, "Self-attentional models for lattice inputs," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2019, pp. 1185–1197.

[21] V. L. Shiv and C. Quirk, "Novel positional encodings to enable tree-based transformers," in *Advances in Neural Information Processing Systems (NIPS)*, December 2019.

[22] X. Li, H. Yan, X. Qiu, and X. Huang, "FLAT: Chinese NER using flat-lattice transformer," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, Jul. 2020, pp. 6836–6842.

[23] R. Ma, H. Li, Q. Liu, L. Chen, and K. Yu, "Neural lattice search for speech recognition," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 7794–7798.

[24] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin, "Convolutional sequence to sequence learning," in *Proceedings of the 34th International Conference on Machine Learning (ICML)*, vol. 70, 2017, pp. 1243–1252.

[25] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "LibriSpeech: an ASR corpus based on public domain audio books," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015, pp. 5206–5210.

[26] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, J. Silovsky, G. Stemmer, and K. Vesely, "The kaldi speech recognition toolkit," in *Workshop on Automatic Speech Recognition and Understanding*. IEEE, 2011.

[27] S. Watanabe, T. Hori, S. Karita, T. Hayashi, J. Nishitoba, Y. Unno, N. Enrique Yalta Soplin, J. Heymann, M. Wiesner, N. Chen, A. Renduchintala, and T. Ochiai, "ESPnet: end-to-end speech processing toolkit," in *INTERSPEECH*, 2018, pp. 2207–2211.

[28] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, Aug. 2016, pp. 1715–1725.

[29] T. Kudo and J. Richardson, "SentencePiece: a simple and language independent subword tokenizer and detokenizer for neural text processing," in *Empirical Methods in Natural Language Processing (EMNLP): System Demonstrations*, Nov. 2018, pp. 66–71.

[30] M. Ott, S. Edunov, A. Baevski, A. Fan, S. Gross, N. Ng, D. Grangier, and M. Auli, "fairseq: a fast, extensible toolkit for sequence modeling," in *Proceedings of NAACL-HLT 2019: Demonstrations*, 2019.

[31] T. Chen, B. Xu, C. Zhang, and C. Guestrin, "Training deep nets with sublinear memory cost," *ArXiv:1604.06174*, 2016.