INTERSPEECH 2019
September 15–19, 2019, Graz, Austria

Two-Pass End-to-End Speech Recognition

Tara N. Sainath[*], *Ruoming Pang*[*], *David Rybach, Yanzhang He, Rohit Prabhavalkar, Wei Li, Mirkó Visontai, Qiao Liang, Trevor Strohman, Yonghui Wu, Ian McGraw, Chung-Cheng Chiu*

Google, Inc., USA
{tsainath,rpang}@google.com

Abstract

The requirements for many applications of state-of-the-art speech recognition systems include not only low word error rate (WER) but also low latency. Specifically, for many use-cases, the system must be able to decode utterances in a streaming fashion and faster than real-time. Recently, a streaming recurrent neural network transducer (RNN-T) end-to-end (E2E) model has shown to be a good candidate for on-device speech recognition, with improved WER and latency metrics compared to conventional on-device models [1]. However, this model still lags behind a large state-of-the-art conventional model in quality [2]. On the other hand, a non-streaming E2E Listen, Attend and Spell (LAS) model has shown comparable quality to large conventional models [3]. This work aims to bring the quality of an E2E streaming model closer to that of a conventional system by incorporating a LAS network as a second-pass component, while still abiding by latency constraints. Our proposed two-pass model achieves a 17%-22% relative reduction in WER compared to RNN-T alone and increases latency by a small fraction over RNN-T.

1. Introduction

There continues to be a growing popularity with end-to-end models (E2E) for speech recognition [1, 3, 4, 5, 6, 7, 8, 9]. These models, which fold the acoustic, pronunciation and language models (AM, PM, LMs) into a single network, have shown competitive results compared to conventional ASR systems which have separate AM, PM, and LMs. E2E models are particularly attractive for on-device ASR, as they can outperform on-device conventional models [10] of comparable size.

Running ASR on-device with direct user interaction, presents numerous challenges. First, the recognition results must be streaming. That is, words should appear on the screen as soon as they are spoken. Second, the model must have a small latency (i.e., the delay between the user speaking and the text appearing), thus running at or faster than real-time on mobile devices. Third, the model must be able to utilize user context [11] (e.g., list of contacts, song names, etc.) to improve recognition accuray. Recently, we presented a RNN-T E2E model that satisfies these constraints [1]. However, the RNN-T model's quality still lags behind that of a large conventional model [2].

Non-streaming E2E models, such as Listen, Attend and Spell (LAS) [7], have shown competitive performance to a large conventional model [3]. However, LAS models are not streaming as they must attend to the entire audio segment, making it challenging to use them in interactive applications.

In two-pass decoding, the second pass model is often used to improve the initial outputs from first-pass models by using lattice rescoring [12] or n-best reranking [13]. Keeping user-perceived latency low while obtaining the quality gains is the main challenge with applying second-pass models. Language model rescoring is commonly been used for multi-pass decoding [14, 15, 16, 17], but more recently has been used with a LAS model to rescore hypotheses from a first-pass conventional model [18]. We can think of LAS decoder, which takes acoustic information from the encoder and language model information from previous predictions, as being strictly stronger than second-pass language models. Thus, in this work, we explore using the LAS model for second-pass processing.

[*] Equal contribution.

Specifically, we explore a two-pass architecture in which an RNN-T decoder and a LAS decoder share an encoder network. Sharing the encoder allows us to reduce model size and computation cost compared with having dedicated encoders for RNN-T and LAS. During inference, the RNN-T model produces streaming predictions while in the end the LAS decoder finalizes the prediction. We explore tradeoffs by running the LAS decoder as a beam search versus rescoring hypotheses from RNN-T.

Our experiments are conducted on a $\sim$30,000 hour voice search task. We find that with LAS second-pass beam search, we can get a 15% relative improvement over first-pass RNN-T for a shorter utterance (SU) test set, but the model degrades on longer utterances (LU), a common problem for attention models [19]. In contrast, second-pass rescoring gives us a much better tradeoff for SU and LU compared to beam-search. Next, we experiment with ways to improve the rescoring model WER by changing the training objective function to more closely match rescoring. Specifically, we apply a minimum word error rate (MWER) training strategy [20] where hypotheses from RNN-T are used as inputs to the LAS decoder and the LAS decoder is trained to minimize expected word error rates. In addition, we reduce computation cost by running the first-pass RNN-T model with an adaptive beam [21] and pruning the first-pass lattice before rescoring. Overall, we find that our proposed LAS rescoring model provides 17% to 22% relative improvement in WER compared to a first-pass RNN-T model, without a degradation in biasing accuracy. In addition, the second-pass LAS decoder increases finalization latency by less than 200ms, which has been considered the limit of acceptable interactive latency [22].

The rest of this paper is organized as follows. Section 2 describes the two-pass architecture and various inference strategies explored in this paper. Experiments are presented in Section 3 while results are discussed in Section 4. Finally, Section 5 concludes the paper and discusses future work.

2. Two-Pass E2E ASR

2.1. Model Architecture

The proposed two-pass architecture is shown in Figure 1. We denote the parameterized input acoustic frames as $\mathbf{x} = (\mathbf{x}_1 \dots \mathbf{x}_T)$, where $\mathbf{x}_t \in \mathbb{R}^d$ are log-mel filterbank energies in this work ($d = 80$) and T denotes the number of frames in $\mathbf{x}$. In the first pass, each acoustic frame $\mathbf{x}_t$ is passed through a shared encoder, consisting of a multi-layer LSTM, to get output $\mathbf{e}_t$, which is passed to an RNN-T decoder for producing $\mathbf{y}_r$ at each time step in a streaming fashion. In the second pass, the output of the shared encoder of all frames $\mathbf{e} = (\mathbf{e}_1 \ldots \mathbf{e}_T)$ is passed to a LAS decoder. During training, the LAS decoder computes output $\mathbf{y}_l$ according to $\mathbf{e}$. During decoding it may additionally use $\mathbf{y}_r$ as described below.

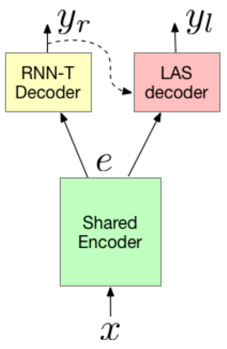

Figure 1: *Two-Pass Architecture*

2.2. Decoding

We explore using the LAS decoder in two different decoding modes in this work. Specifically,

- In the "2nd beam search" mode, it produces output $\mathbf{y}_l$ from $\mathbf{e}$ alone, ignoring $\mathbf{y}_r$, the output of the RNN-T decoder.
- In the "rescoring" mode, we first pick the top-K hypotheses from the RNN-T decoder. We then run the LAS decoder on each sequence in the teacher-forcing mode, with attention on $\mathbf{e}$, to compute a score, which combines log probability of the sequence and the attention coverage penalty [23]. The sequence with the highest LAS score is picked as the output sequence.

2.3. Training

In this section, we describe the training stategies for the two-pass model.

2.3.1. *Combined Loss*

In theory we can train a two-pass model from random initialization with the following combined loss, where $\mathbf{y}^*$ represents the ground truth transcript:

$$\mathbf{L}_{\texttt{combined}}(\mathbf{x}, \mathbf{y}^*) = \lambda \mathbf{L}_{\texttt{RNNT}}(\mathbf{x}, \mathbf{y}^*) + (1-\lambda)\mathbf{L}_{\texttt{LAS}}(\mathbf{x}, \mathbf{y}^*) \quad (1)$$

In the above equation, λ is a hyperparameter, which we set to be 0.5 in our setup to equally weight the RNN-T and LAS losses. In practice we find training directly from scratch to be unstable, mainly because the losses for RNN-T and LAS are in drastically different ranges when training from scratch. Therefore, we take a multi-step process to train the model:

1. Train an RNN-T model as in [1];
2. Take the encoder trained in step (1), freeze it, and train a LAS decoder as in [3].
3. "Deep finetuning": train the shared encoder and both decoders at the same time with the combined loss.

2.3.2. *MWER training*

One of the drawbacks of the loss in Equation 1 is that the second-pass LAS decoder is optimized independently of the RNN-T decoder. This means that there is a mismatch between the training and decoding strategies outlined in Section 2.2.

To address this, we use an additional training step to further refine the LAS decoder to minimize errors, following the MWER training process introduced in [20]. Specifically, given input $\mathbf{x}$, groundtruth transcript $\mathbf{y}^*$, the probability computed by LAS $P(\mathbf{y}_m|\mathbf{x})$ for any given target sequence $\mathbf{y}_m$ with teacher-forcing (where $m = r$ if $\mathbf{y}_m$ is given by RNN-T and $m = l$ if $\mathbf{y}_m$ is given by LAS), we refine the pre-trained two-pass model as follows.

First, we run a beam search with one of the decoders m from the two-pass model to get a set of hypotheses $H_m = \{h_1, \ldots, h_b\}$ where b is the beam-size. To make the MWER training match decoding, the generation of H_m depends on the target decoding mode. For a LAS decoder to be used in the "2nd beam search" mode, we compute H_m by running beam search with the LAS decoder itself on $\mathbf{x}$ ($m = l$). For a LAS decoder to be used in the "rescoring" mode, on the other hand, we compute $H_m(\mathbf{x})$ by running beam search with the first-pass RNN-T decoder ($m = r$).

For each sequence $\mathbf{y}_m \in H_m$, let $W(\mathbf{y}^*, \mathbf{y}_m)$ be the number of word errors of $\mathbf{y}_m$, let $\overline{W}(\mathbf{y}^*, H_m) = \frac{1}{|H_m|}\sum_{\mathbf{y}_m \in H_m} W(\mathbf{y}^*, \mathbf{y}_m)$ be the mean number of word errors for H_m, and let $\hat{W}(\mathbf{y}^*, \mathbf{y}_m) = W(\mathbf{y}^*, \mathbf{y}_m) - \overline{W}(\mathbf{y}^*, H_m)$ be the relative word error rate of $\mathbf{y}_m$ in H_m. We also let $\hat{P}(\mathbf{y}_m|\mathbf{x}, H_m) = \frac{P(\mathbf{y}_m|\mathbf{x})}{\sum_{\mathbf{y}_i \in H_m} P(\mathbf{y}_i|\mathbf{x})}$ represent the conditional probability LAS decoder assigns to hypothesis $\mathbf{y}_m$ among all hypotheses in H_m. The MWER loss is defined as

$$\mathbf{L}_{\texttt{MWER}}(\mathbf{x}, \mathbf{y}^*) = \sum_{\mathbf{y}_m \in H_m(\mathbf{x})} \hat{P}(\mathbf{y}_m|\mathbf{x}, H_m)\hat{W}(\mathbf{y}^*, \mathbf{y}_m) \quad (2)$$

We train the LAS decoder to minimize a combination of the MWER loss and the maximum-likelihood cross-entropy loss:

$$\mathbf{L}_{\texttt{MWER}}(\mathbf{x}, \mathbf{y}^*) + \lambda_{\texttt{MLE}} \log P(\mathbf{y}^*|\mathbf{x}) \quad (3)$$

where $\lambda_{\texttt{MLE}}$ is a hyperparameter that experimentally we set to be $\lambda_{\texttt{MLE}} = 0.01$ following [20].

3. Experimental Details

3.1. Data Sets

Our experiments are conducted on a ~30,000 hour training set consisting of 43 million English utterances. The training utterances are anonymized and hand-transcribed, and are representative of Google's voice search traffic in the United States. Multi-style training (MTR) data are created by artificially corrupting the clean utterances using a room simulator, adding varying degrees of noise and reverberation with an average SNR of 12dB [24]. The noise sources are drawn from YouTube and daily life noisy environmental recordings. The main test sets we report results on include ~14K short utterances (*SU*) less than 5.5 seconds long and ~16K long utterances (*LU*) greater than 5.5 seconds, both extracted from Google traffic.

To evaluate the performance of contextual biasing, we report performance on a contacts test set, which consists of requests to call/text contacts. This set is created by mining contact names from the web, and synthesizing TTS utterances in each of these categories using a concatenative TTS approach with one voice [25]. Noise is then artificially added to the TTS data, similar to the process described above [24]. To bias model predictions towards contacts, we construct a biasing FST on a list of contact phrases and perform shallow-fusion between the biasing FST and E2E model during inference. We refer the reader to [26] for more details regarding E2E shallow-fusion biasing.

3.2. Model Architecture Details

All experiments use 80-dimensional log-Mel features, computed with a 25ms window and shifted every 10ms. Similar to [2], at the current frame, t, these features are stacked with 2 frames to the left and downsampled to a 30ms frame rate.

The same encoder network described in [1] is used for all experiments. It consists of 8 LSTM layers, where each layer has 2,048 hidden units followed by a 640-dimensional projection layer. We insert a time-reduction layer with the reduction factor $N = 2$ after the second LSTM layer of encoder.

The RNN-T decoder contists of a prediction network and a joint network. The prediction network has 2 LSTM layers of 2,048 hidden units and a 640-dimensional projection per layer as well as an embedding layer of 128 units. The outputs of encoder and prediction network are fed to a joint network that has 640 hidden units. The LAS decoder consists of multi-head attention [27] with four attention heads, which is fed into 2 LSTM layers of 2,048 hidden units and a 640-dimensional projection layer. It has an embedding layer of 96 units. Both decoders are trained to predict 4,096 word pieces [28], which are derived using a large corpus of text transcripts.

The total size of the RNN-T model is 114M parameters, and the additional second-pass LAS decoder is 33M parameters. All models are trained in Tensorflow [29] using the Lingvo [30] toolkit on 8×8 Tensor Processing Units (TPU) slices with a global batch size of 4,096.

3.3. Measuring Latency

As computing devices may vary, we use a simplified model of computation to estimate latency. First, we assume that the bandwidth K on CPU is 10GB/second; this number is within the range of modern mobile CPUs. We also make the non-optimal assumption that each hypothesis is computed independently, meaning that the major operations are matrix/vector multiplies, the time of which will be dominated by the speed of loading matrix parameters into the CPU.

Assuming no interrupts or batching across beam search hypotheses, the latency is calculated from Equation 4 when doing fixed beam decoding/rescoring with H hypotheses over N tokens. When using an adaptive beam, where a lattice is generated, we assume $H \cdot N$ is now replaced by the number of lattice arcs when calculating latency.

$$\text{latency} = \frac{1}{K} \cdot H \cdot N \cdot M_{\text{decoder}} \tag{4}$$

where M_{decoder} denotes the number of bytes in the decoder part of the model.

We report latency on the 90%-tile *LU* set which has longer utterances. We assume the 90%-tile contains roughly 295 audio frames and a target sequence of $N = 28$ tokens. Finally, M_{decoder} is 33MB, assuming 33M parameters of the LAS decoder which are quantized, which we have found has a negligible degradation in accuracy [1]. Our goal is to ensure that the second-pass latency on the 90%-tile is under 200ms to that user-perceived latency is minimized [22].

4. Results

4.1. 2nd Beam Search

Table 1 shows the results running the LAS decoder in the 2nd beam-search mode. For comparison, the table also shows two baselines `B0-B1`, namely an RNN-T only and a LAS-only model, trained separately from scratch. All results are obtained with fixed beam-size of $H = 8$.

Experiment `E0` indicates that when the encoder is initialized from an RNN-T model and held fixed, the LAS decoder performs worse than a LAS-only model with a dedicated encoder (`B1`), demonstrating the challenges in sharing a single encoder with different types of decoders by adapting the LAS decoder alone. When we jointly train the encoder and both decoders in a model initialized from `E0`, the model quality (`E1`) improved in both SU and LU over `E0`. Overall we find that 2nd beam search improves over RNN-T (`B0`) on SU but degrades on LU, a common issue with attention models for long utterances [19].

Table 1: *WER Results, LAS Beam Search.*

Exp-ID	Model	SU	LU
B0	RNN-T	6.9	4.5
B1	LAS-only	5.4	4.5
E0	Frozen Shared Enc	6.4	5.3
E1	Deep Finetuned	6.1	4.8

4.2. Rescoring

We noticed that the RNN-T-only model (`B0`) has much lower oracle WERs than its decoding WERs. This motivates us to explore rescoring RNN-T hypothesis with the LAS decoder. Table 2 compares the performance of running LAS with beam search (`E1`) to with rescoring (`E2`). The table shows that rescoring takes a small hit in WER on SU compared to beam search, probably because the first-pass RNN-T decoder, with a much higher SU WER of 6.9 (`B0`), generates a set of hypotheses with slightly lower quality than those generated by the LAS decoder during beam search. However, rescoring's quality on LU is much better than that of beam search, likely because RNN-T (`B0`) performs much better on longer utterances compared to LAS. Overall, LAS rescoring not only improves SU WER significantly upon the first pass RNN-T, but also improves WER for LU, demonstrating that rescoring is able to combine the strengths of RNN-T and LAS. Since rescoring gives us the best tradeoff between quality on SU and LU, we will focus only on rescoring and present further improvements in the next section.

Table 2: *WER results, LAS Rescoring.*

Exp-ID	Decoding	SU	LU
B0	RNN-T	6.9	4.5
B1	LAS-only	5.4	4.5
E1	Beam Search	**6.1**	4.8
E2	Rescoring	6.2	**4.1**

4.3. Further Rescoring Improvements

4.3.1. Adaptive Beam

To bridge the gap between two-pass 2nd beam search vs. rescoring on SU, we first explore increasing the diversity of rescoring candidates with a larger first-pass RNN-T beam. Table 3 shows that as beam size is increased (`E2-E4`), the WER improves, but naturally at cost of proportionally increased first-pass computation cost. To address this, we look at an adaptive beam search strategy [21]. Specifically, we prune first-pass beam candidates if they are too far in threshold from the current best candidate, where the threshold optimizes first-pass latency following [1]. The table shows that with an adaptive beam (`E5`), we can achieve similar WER to a fixed but large beam (`E3`).

Table 3: *Rescoring WER with first-pass fixed vs. adaptive beam.*

Exp-ID	First-pass Max Beam Size	SU	LU
E2	Fixed, 8	6.2	4.1
E3	Fixed, 10	6.2	4.1
E4	Fixed, 16	6.1	4.1
E5	Adaptive, 10	6.2	4.0

The adaptive-beam also has the additional benefit that it generates a lattice to use for rescoring, rather than an N-best list. Rescoring a lattice is more efficient than rescoring an N-best list, as it avoids duplicate computation on the common prefixes between candidate sequences, and thus should reduce latency. As a reminder, latency in Equation 4 is now calculated by looking at the total arcs in the lattice. Table 4 compares adaptive beam to a fixed beam with N-best rescoring, where we rescore all first-pass hypotheses. The table show that with an adaptive beam and lattice rescoring, we can reduce latency compared to a fixed beam with N-best rescoring. However, the latency is still above our budget.

Table 4: *Latency vs. Rescoring Methods.*

Strategy	Latency (ms)
1st-pass Fixed, N-best Rescoring	369.6
1st-pass Adaptive, Lattice Rescoring	**247.5**

To reduce latency further, we explore reducing the number of maximum arcs in the lattice rescored at each step. Table 5 shows we can limit the rescored hypotheses to 4, which we find does not degrade accuracy and also reduces latency. Overall, the second-pass decoder rescoring an adaptive-beam lattice fits within our 200ms latency budget.

Table 5: *Two-Pass Performance vs. Las Rescoring Beam Size.*

Beam Size	SU	LU	Contacts	Latency (ms)
2	6.2	4.0	7.5	-
4	**6.2**	**4.0**	**7.1**	**171.6**
8	6.2	4.0	7.1	247.5

4.3.2. MWER

Finally, we report two-pass results after MWER training our model. Since the LAS decoder will be used for rescoring, we use RNN-T to provide the candidate hypotheses for LAS decoder MWER training. Table 6 shows that MWER improves rescoring WER for both SU and LU by 8% relative. Overall, the two-pass rescoring model gives a 17% and 22% relative reduction in both SU and LU, respectively.

Table 6: *Two-pass rescoring results after MWER training.*

Exp-ID	Model	SU	LU	Contacts
B0	RNN-T only	6.9	4.5	7.0
E6	No MWER	6.2	4.0	7.1
E7	MWER	**5.7**	**3.5**	**7.0**

4.4. Comparison To Large Conventional Model

A goal of our work is to achieve in an E2E system with comparable performance to a large conventional model [2]. In this light, we compare the performance of our proposed two-pass rescoring model to a large conventional model through a "side-by-side" (SxS) evaluation with previously unseen utterances. In this experiment, each utterance is transcribed by both the conventional and two-pass models. We collect 500 utterances where the transcription differs between the two models, and send these utterances to be rated by two human transcribers. Each transcript is rated as either a win by two-pass over the conventional model (only two-pass is correct), a loss in two-pass over the conventional model (only the conventional model is correct), or neutral (both models are correct or incorrect). Unlike automatic WER evaluations, this side-by-side evaluation allows raters to decide that two different transcripts are both correct; this sometimes leads to different conclusions than an automatic evaluation would. We report the following statistics to quantitatively evaluate the SxS:

- Changed: % of utterancs in which the two models produced different hypotheses
- Wins: # of utts the two-pass hypothesis is correct and conventional model is incorrect
- Losses: # of utts the two-pass hypothesis is incorrect and conventional model is correct
- Neutral: # of utts the two-pass and conventional model are both correct or incorrect
- p-Value: Statical significance of WER change with two-pass compared to conventional model

Table 7 shows that the two-pass model changes about 13% of traffic. The two-pass model has slightly more losses (61) than wins (48) compared to the conventional model, but the majority of the hypotheses have a neutral rating (391) between the two systems. Overall, the p-Value shows the performance difference between the two models is statistically insignificant.

Table 7: *SxS results for Conventional vs. Two-pass*

Changed (%)	Win	Loss	Neutral	p-Value
13.2	48	61	391	10.0%-20.0%

A further analysis of errors is shown in Table 8. The two-pass model is trained with an order of magnitude less text-only data compared to the conventional model, and thus loses on proper nouns (PN) and also due to a weak language-model (wLM). On the contrary, since the two-pass model is trained in the written domain and learns text normalization (TN) implictly, it wins in this area compared to the conventional model which has a separate rule-based text-norm step.

Table 8: *Analysis of Errors of Conventional vs. Two-Pass Model.*

	Type	Conventional	Two-Pass
Loss	PN	alice's restaurant	**allison's restaurant**
	wLM	47-in sony plasma tv	47-in sony **pricing** tv
Win	TN	www nytimes.com	**www.nytimes.com**
	TN	john smiths office	john **smith's** office

5. Conclusions

In this paper, we present a two-pass E2E solution. Specifically, we use a second-pass LAS decoder to rescore hypotheses from a first-pass RNN-T system. We find that this approach gives a 17% to 22% reduction in WER compared to RNN-T only, and increases latency by less than 200ms.

6. References

[1] Y. He, T. N. Sainath, R. Prabhavalkar, I. McGraw, R. Alvarez, D. Zhao, D. Rybach, A. Kannan, Y. Wu, R. Pang, Q. Liang, D. Bhatia, Y. Shangguan, B. Li, G. Pundak, K. Sim, T. Bagby, S. Chang, K. Rao, and A. Gruenstein, "Streaming End-to-end Speech Recognition For Mobile Devices," in *Proc. ICASSP*, 2019.

[2] G. Pundak and T. N. Sainath, "Lower frame rate neural network acoustic models," in *Proc. Interspeech*, 2016.

[3] C. Chiu, T. N. Sainath, Y. Wu, R. Prabhavalkar, P. Nguyen, Z. Chen, A. Kannan, R. J. Weiss, K. Rao, N. Jaitly, B. Li, and J. Chorowski, "State-of-the-art speech recognition with sequence-to-sequence models," in *Proc. ICASSP*, 2018.

[4] A. Graves, "Sequence transduction with recurrent neural networks," *CoRR*, vol. abs/1211.3711, 2012.

[5] A. Graves, A.-R. Mohamed, and G. Hinton, "Speech recognition with deep neural networks," in *Proc. ICASSP*, 2012.

[6] K. Rao, H. Sak, and R. Prabhavalkar, "Exploring architectures, data and units for streaming end-to-end speech recognition with rnn-transducer," in *Proc. ASRU*, 2017, pp. 193–199.

[7] W. Chan, N. Jaitly, Q. V. Le, and O. Vinyals, "Listen, attend and spell," *CoRR*, vol. abs/1508.01211, 2015.

[8] S. Kim, T. Hori, and S. Watanabe, "Joint CTC-attention based end-to-end speech recognition using multi-task learning," in *Proc. ICASSP*, 2017, pp. 4835–4839.

[9] C.-C. Chiu and C. Raffel, "Monotonic chunkwise alignments," in *Proc. ICLR*, 2017.

[10] I. McGraw, R. Prabhabalkar, R. Alvarez, M. Gonzalez, K. Rao, D. Rybach, O. Alsharif, H. Sak, A. Gruenstein, F. Beaufays, and C. Parada, "Personalized speech recognition on mobile devices," in *Proc. ICASSP*, 2016.

[11] P. Aleksic, M. Ghodsi, A. Michaely, C. Allauzen, K. Hall, B. Roark, D. Rybach, and P. Moreno, "Bringing contextual information to Google speech recognition," in *Proc. Interspeech*, 2015.

[12] S. Ortmanns, H. Ney, and X. Aubert, "A word graph algorithm for large vocabulary continuous speech recognition," *Computer Speech and Language*, vol. 11, no. 1, pp. 43–72, Jan. 1997.

[13] R. Schwartz and S. Austin, "A comparison of several approximate algorithms for finding multiple (N-best) sentence hypotheses," in *Proc. ICASSP*, 1991, pp. 701–704.

[14] M. Sundermeyer, H. Ney, and R. Schlüter, "From Feedforward to Recurrent LSTM Neural Networks for Language Models," *IEEE Trans. Audio, Speech and Language Processing*, vol. 23, no. 3, pp. 517–528, 2015.

[15] 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 Trans. Audio, Speech and Language Processing*, vol. 24, no. 8, pp. 1438–1449, 2016.

[16] 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.

[17] A. Kannan, Y. Wu, P. Nguyen, T. N. Sainath, Z. Chen, and R. Prabhavalkar, "An analysis of incorporating an external language model into a sequence-to-sequence model," in *Proc. ICASSP*, 2018.

[18] R. Prabhavalkar, T. N. Sainath, B. Li, K. Rao, and N. Jaitly, "An analysis of "attention" in sequence-to-sequence models," in *Proc. Interspeech*, 2017.

[19] J. K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-Based Models for Speech Recognition," in *Proc. NIPS*, 2015.

[20] R. Prabhavalkar, T. N. Sainath, Y. Wu, P. Nguyen, Z. Chen, C. C. Chiu, and A. Kannan, "Minimum Word Error Rate Training for Attention-based Sequence-to-sequence Models," in *Proc. ICASSP*, 2018.

[21] B. T. Lowerre, *The Harpy Speech Recognition System.*, Ph.D. thesis, Pittsburgh, PA, USA, 1976.

[22] R. B. Miller, "Response time in man-computer conversational transactions," in *Proceedings of the December 9-11, 1968, Fall Joint Computer Conference, Part I*, New York, NY, USA, 1968, AFIPS '68 (Fall, part I), pp. 267–277, ACM.

[23] J. K. Chorowski and N. Jaitly, "Towards Better Decoding and Language Model Integration in Sequence to Sequence Models," in *Proc. Interspeech*, 2017.

[24] C. Kim, A. Misra, K. Chin, T. Hughes, A. Narayanan, T. N. Sainath, and M. Bacchiani, "Generated of large-scale simulated utterances in virtual rooms to train deep-neural networks for far-field speech recognition in Google Home," in *Proc. Interspeech*, 2017.

[25] X. Gonzalvo, S. Tazari, C. Chan, M. Becker, A. Gutkin, and H. Silen, "Recent Advances in Google Real-time HMM-driven Unit Selection Synthesizer," in *Proc. Interspeech*, 2016.

[26] G. Pundak, T. Sainath, R. Prabhavalkar, A. Kannan, and D. Zhao, "Deep Context: End-to-End Contextual Speech Recognition," in *Proc. SLT*, 2018.

[27] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention Is All You Need," *CoRR*, vol. abs/1706.03762, 2017.

[28] M. Schuster and K. Nakajima, "Japanese and Korean voice search," in *Proc. ICASSP*, 2012.

[29] M. Abadi et al., "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," Available online: http://download.tensorflow.org/paper/whitepaper2015.pdf, 2015.

[30] Jonathan Shen, Patrick Nguyen, Yonghui Wu, Zhifeng Chen, et al., "Lingvo: a modular and scalable framework for sequence-to-sequence modeling," 2019.