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

A Light-weight contextual spelling correction model for customizing transducer-based speech recognition systems

Xiaoqiang Wang[1], *Yanqing Liu*[1], *Sheng Zhao*[1], *Jinyu Li*[2]

[1]Microsoft, China
[2]Microsoft, USA
{xiaoqwa, yanqliu, szhao, jinyli}@microsoft.com

Abstract

It's challenging to customize transducer-based automatic speech recognition (ASR) system with context information which is dynamic and unavailable during model training. In this work, we introduce a light-weight contextual spelling correction model to correct context-related recognition errors in transducer-based ASR systems. We incorporate the context information into the spelling correction model with a shared context encoder and use a filtering algorithm to handle large-size context lists. Experiments show that the model improves baseline ASR model performance with about 50% relative word error rate reduction, which also significantly outperforms the baseline method such as contextual LM biasing. The model also shows excellent performance for out-of-vocabulary terms not seen during training.

Index Terms: speech recognition, contextual spelling correction, contextual biasing

1. Introduction

Recently, end-to-end (E2E) modeling becomes the most significant trend for automatic speech recognition (ASR). Because of its streaming nature, transducer models [1] including recurrent neural network (RNN) Transducer (RNN-T) [2, 3, 4, 5] and Transformer Transducer [6, 7, 8]. While achieving similar or even better recognition accuracy than hybrid models [2, 9], transducer models are facing lots of practical challenges. Customizing transducer models with dynamic context is one of the most challenging tasks. For example, in an intelligent personal assistant, "*call <PersonName>*" is customized with the user's contact list. Such contexts are dynamic, personalized, or occasion related. More examples include names, songs, playlist, location, etc. These contexts have not been observed during training, therefore transducer models usually perform poorly when recognizing these contexts.

There are several studies trying to improve the accuracy of customized transducer models. The first approach is to integrate the contextual information in the decoding process by rescoring the output probabilities with a contextual language model (LM) constructed from context phrases, referred to as shallow fusion or on-the-fly rescoring [10, 11, 12, 13]. The rescoring method itself is a one-pass process, which also relies heavily on the ASR model output distributions and the weights should be tuned carefully in different using scenarios. The second approach, contextual RNN-T [14], generally encodes contextual information into the transducer model and regards the contexts as additional model input apart from audio features. However, this method may not be scaled well with a large biasing list. Furthermore, because contextual RNN-T changes the baseline model structure, it is more expensive in training and inference, and may result in performance degradation as noted in the work from the same organization [15]. Although the deep shallow fusion [15] can achieve reasonable performance for customization, it significantly degrades the performance on general sets.

In this work, we propose a novel contextual biasing method which leverages contextual information by adding a contextual spelling correction (CSC) model on top of the transducer model. To consider contextual information during correction, a context encoder which encodes context phrases into hidden embeddings is added to the spelling correction model [16, 17], the decoder of the correction model then attends to the context encoder and text encoder by attention mechanism [18]. The proposed model is a standalone correction model which does not change the original transducer model structure, hence there is no performance degradation risk for the baseline ASR model. It's convenient for the proposed method to be applied in different domains by just changing the CSC model without retraining the original ASR model. To address the problem of general domain regression and large context list, we propose a filter mechanism to trigger the correction process and control the context list size based on ASR hypotheses. Experiments show that the proposed CSC model significantly improves the ASR accuracy of RNN-T models in domains with contextual phrases such as personal names and outperforms the baseline biasing methods. The model also shows excellent performance for out-of-vocabulary (OOV) terms not seen during training. With the help of teacher-student learning [19, 20] and quantization [21], the model can be compressed to a small size, which makes it easy to be deployed.

2. Related Work

2.1. Contextual LM

Contextual LM is adopted as one of our baseline models by following the shallow-fusion end-to-end contextual biasing method proposed by Zhao et.al [13]. Given the audio input x, shallow fusion interpolates the transducer ASR model with an external contextual LM:

$$y^* = \arg\max_y logP(y|x) + \lambda logP_c(y) \quad (1)$$

Where P and P_c are the probabilities from baseline ASR model and contextual LM, λ is a tunable parameter. The contextual LM, $min(det(S \circ G))$, is obtained by compiling the list of biasing phrases into an n-gram weighted finite state transducer (WFST) [22] G, Where G is left composed with a "speller" FST S to transduce wordpieces [23] into the corresponding word. To address the concern that biasing hurts the accuracy of utterances without contexts, activation prefix [13] is also integrated and biasing is activated only when it's preceded by prefix.

2.2. Bias encoder

There have been growing interests in incorporating contextual information into E2E ASR model with bias encoders.Pundak

http://dx.doi.org/10.21437/Interspeech.2021-379

et al. proposed CLAS model [24] which adds a context bias encoder to LAS [25] by taking context phrases as input. The decoder calculates attention with both the audio encoder and bias encoder, and the obtained attention vectors from the two encoders are then concatenated to generate the final attention. Similar idea was applied to RNN-T as contextual RNN-T [14]. However, this method may not be scaled well when the context list is large, and it also complicates the training and inference.

2.3. Spelling correction

Spelling correction [16, 17] in ASR system aims to correct recognition errors in ASR hypotheses, which is shown to be effective in model performance improvement. There is also some work targeting on proper noun recognition problem [26]. Generally, spelling correction is more like machine translation [27] which "translates" ASR hypotheses with error terms into the correct one. Different from traditional spelling correction, we additionally correct ASR hypotheses with contextual information, which takes dynamic biasing phrases for customization.

3. Contextual spelling correction

3.1. Model structure

The proposed model is a seq2seq [28] model with a text encoder, a context encoder, and a decoder, which takes ASR hypothesis as the input to text encoder, context phrase list as the input to context encoder. The context encoder encodes context phrases as context embeddings, the decoder attends to the output of both encoders, obtaining information from ASR hypotheses hidden states and context embeddings to correct contextual misspelling. All the components are transformer-based [29], composed by pre-LayerNorm [30, 31], self-attention, encoder-decoder attention and feed-forward layer (FFN). The parameters of the two encoders are shared, as shown in figure 1. In the following, we describe the key features of the proposed model.

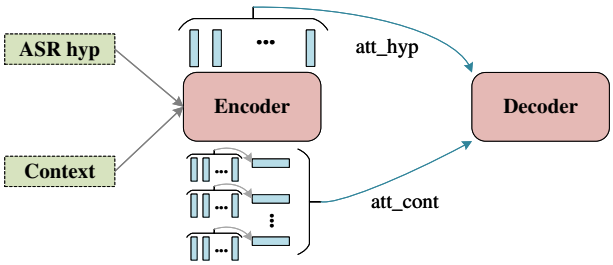

Figure 1: *Contextual spelling correction (CSC) model structure.*

3.1.1. Context encoder

Different from general spelling correction model, the contextual information is incorporated with this context encoder as embeddings, which is generated by averaging the hidden representations of the context phrase as a single vector. At each decoding step, the query vector also pays attention to context phrase embeddings, the attention of text encoder and context encoder are then added up to generate the final attention during decoding.

3.1.2. Light Weight model

To enable on-device application, we reduce the model size and improve inference efficiency with the following methods:

- Parameter sharing. Parameters are shared between the text encoder and context encoder by using a single encoder network for feature extraction of both ASR hypotheses and contexts. This design comes from: 1) ASR hypotheses and context phrases are both transcriptions that could be processed by the same network; 2) In many cases such as domain with user names, the context phrase list for training is not that large to cover all the possible word tokens or patterns, using the same network makes the context encoder benefit from text encoder; 3) Use a single encoder network makes the model smaller. Furthermore, the embeddings of encoder and decoder are also shared.
- Teacher-student learning [19, 20] and quantization [21]. Both methods are used to reduce the model size.

3.2. Training

3.2.1. Text to speech data

In most scenarios, it is time and cost consuming to get enough speech data with context phrases and label them. Generating text to speech (TTS) audio using preprocessed transcriptions with context phrases is an efficient and low-cost way to augment the training data.

We first collect a number of sentence patterns with contexts (in our case, person names), such as "*call <PersonName>*", "*do I have any mails from <PersonName> today*", etc, and combine these sentence patterns with a prepared context list randomly to get training scripts. Then we use TTS to generate audio and run through the RNN-T model to get ASR hypotheses. The input reference, context phrases and generated ASR hypotheses are finally paired for CSC model training.

3.2.2. Data augmentation

Due to the sparsity of transcripts with contexts, the collected sentence patterns is limited, which will bring about overfitting on these sentence patterns. There are two approaches to deal with this problem. The first method is to use some NER [32, 33] tools to filter out sentences with contexts from large amount of data. The second one is to randomly replace words in general sentence with ASR hypotheses-reference pairs of prepared context phrases to generate training pairs artificially, e.g, $\{x$: *you are Bob*, y: *you are Bobby*$\}$ may be generated from "*you are right*" by replacing word "*right*" with pairs $\{$*Bob*, *Bobby*$\}$. This method is simpler but may destroy the grammar to some extent. To make the data preparation process simpler, we tested the second method. From the training results we found it does not influence the model performance much.

3.2.3. Context setup

During training, each utterance has one or two ground-truth context phrases. We pool all the context phrases from all utterances in the batch together and use this to sample the context phrase list for each utterance, a mask is used to distinguish the sampled context phrase list among these utterances. The size of context phrase list N_c for each utterance is randomly sampled from a uniform distribution $[1, N_{cmax}]$, where N_{cmax} is max context list size. To take care of the general sentences without context or cases that the ground-truth context doesn't appear in context list, we keep the proportion of training samples without ground-truth context to be $1 - P_{cont}$. What's more, similar to [24, 34, 35], a token </cont> is added after the context phrase

c in reference y. A model input example is described as:
ASR hyp x: *Who is Alyssa Friedman.*
Contexts c: *Samira, Trump, Aliza Friedman, ..., Joe Biden*
Reference y: *Who is Aliza Friedman </cont>.*

3.2.4. Teacher-student learning

After the basic CSC model is trained, teacher-student learning [19] is also adopted to further reduce the model size and improve the inference efficiency, which enables the model to be used on device. The loss function of the student model is:

$$L = \alpha L_{soft} + (1-\alpha) L_{hard} \tag{2}$$

$$L_{hard} = H(y_S, y) \tag{3}$$

$$L_{soft} = D_{\mathrm{KL}}\left(\mathrm{softmax}(\frac{y_S}{T}), \mathrm{softmax}(\frac{y_T}{T})\right) \cdot T^2 \tag{4}$$

Where L_{hard} is cross-entropy loss of student model output y_S and reference y, L_{soft} is KL-divergence of y_S and teacher model output y_T. T is a temperature parameter, and α is a weight to determine the proportion of L_{hard} and L_{soft}.

3.3. Inference

During inference, the ASR hypotheses is first filtered by a pattern classifier, and the context list is filtered by a relevance ranker and a preference ranker to address the large size context list issue in existing solutions like CLAS. Then the processed data is fed into the CSC model to correct contextual errors.

3.3.1. Pattern classifier

The pattern classifier aims to classify the input ASR hypotheses into "to be corrected" or "not to be corrected" in order to avoid the regression of general sentences that we do not want to bias. For simplicity, we adopt a rule-based classifier which is generated from training scripts and considers the sentence patterns (e.g., sentences with "*call*", "*forward ... to*", etc.) to determine when to trigger our CSC model.

3.3.2. Relevance ranker (rRanker)

To avoid performance degradation and increase inference efficiency when the context list is extremely large, a relevance ranker is used to constrain context list size based on the ASR hypotheses. For sentence s, we calculate the relevance ranker weight of a certain context phrase c_j as

$$W_r^j = -\frac{\min_i(\mathrm{editdistance}(c_j, s_i))}{\mathrm{len}(c_j)} \tag{5}$$

where s_i is a segment cut off from input text s with the same length of the context phrase c_j begin from the i-th word.

3.3.3. Preference ranker (pRanker)

In most scenarios, there is additional preference knowledge for us to determine the prior distribution of the context list. For example, in the personal assistant scenario, the context preference of each contact in the user's contact list can be determined as the query frequency of this contact. To utilize this information, we also introduce a rank weight W_p based on user's preference to preselect context together with relevance ranker weight W_r. The final selected context list can be described as:

$$c_1, c_2, ..., c_k = \mathrm{argtopk}(\alpha_p W_p + (1-\alpha_p) W_r) \tag{6}$$

Where α_p determines the proportion of relevance ranker weight W_p and preference rank weight W_r.

Table 1: *CSC Model parameters*

Model	layers	d_model	heads	d_FFN	Params(M)
Teacher	6	512	8	2048	55.4
Student	3	192	4	768	5.2

3.3.4. Decode with CSC

For each utterance, we get top N_{asr} ASR hypotheses $\{H_1, H_2, ..., H_{N_{asr}}\}$ which may contain contextual spelling errors. Each hypothesis H_i is then fed into the CSC model with the pre-selected context list and the corresponding CSC hypotheses $\{H_{i1}, H_{i2}, ..., H_{iN_{csc}}\}$ is generated by beam search mechanism. The final decoding results are obtained by ranking these $N_{asr} \times N_{csc}$ hypotheses:

$$H^* = \arg\max_H \lambda_{asr} logP_i + \lambda_{csc} logP_{ij} \tag{7}$$

where λ_{asr} and λ_{csc} are the weights for ASR and CSC scores.

4. Experiment

Our experiment targets on correcting person name spelling errors for an English-speaking personal assistant scenario in which name contexts are occasionally triggered. To generate training scripts with contexts, we first collected 512 sentence patterns with name token that frequently appear in this scenario. Then 1 million (M) training scripts are constructed by randomly combining these sentence patterns with 1M names which are generated from 48 thousand (K) name words. These scripts are used to generate TTS audio as described in section 3.2.1. We use Fastspeech AM [36] + lpcnet vocoder [37] as the TTS model. To augment the training set, we also used another 26M in-house general scripts which are also from this scenario following the approach described in section 3.2.2.

We use two test sets to evaluate the model performance, including a test set with person names (denoted as name set) and a general set without names. The name set contains 11K utterances. The name list size (context phrase list size) K_r of each utterance is 1509 on average. This is much larger than the size in [14, 15], indicating our setup is more challenging. The general set contains 8K utterances without names.

4.1. Experimental Setup

During training, we select $batch\_size = 250$, $N_{cmax} = 100$ and $P_{cont} = 0.8$, which means the max size of context phrase list of an utterance is 100 in training process, and there are about 80% utterances with context inside each batch. Based on the above settings, we trained a teacher model, and then distilled it to a student model using the same training scripts, the distillation parameters are set to be $T = 1$, $\alpha = 0.9$. The detailed CSC model parameters are listed in Table 1. During inference, the size of filtered context list K_f after rRanker and pRanker is set to be 80 to get consistent with training. We also take $N_{asr} = 4$, $N_{csc} = 1$ and $\alpha_p = 0.3$ during CSC decoding.

4.2. Performance

Table 2 shows the CSC model performance on three RNN-T models. All these RNN-T models have 1600 LSTM [38] memory cells and the output is projected to 800. The encoder has 6 layers while the prediction network has 2 layers LSTM. The joint network outputs a vector with dimension 640. The feature

Table 2: *Model perf with ASR version change on name set*

Model	v1	v2	v3
RNN-T	28.9	26.8	30.4
RNN-T+CSC	14.5(49.8%)	13.3(50.4%)	16.1(47.0%)

Table 3: *Model performance, WER(WERR)*

	Name set	General set
RNN-T	30.4	13.4
RNN-T+contextual LM	22.2(27.0%)	13.6(−1.5%)
RNN-T+CSC (w/o pRanker)	17.4(42.8%)	13.8(−3.0%)
RNN-T+CSC	16.1(47.0%)	13.7(−2.2%)

is 80-dimension log Mel filter bank for every 10 milliseconds (ms) speech. Three of them are stacked together to form a frame of 240-dimension input acoustic feature to the encoder network. The output targets are 4 thousand word-piece units. As described in [9], all these RNN-T models were trained with 65 thousand hours of anonymized transcribed data with personally identifiable information removed, differing only at how many future frames were observed at each encoder layer.

The CSC model was trained using the v1 RNN-T model. We report both WER and relative WER reduction (WERR) from the baseline RNN-T model for the proposed RNN-T+CSC method in Table 2. We can see that final model performance does not change too much among different RNN-T models, which indicates the robustness of the proposed method.

In Table 3, we compare the performance of CSC with the baseline RNN-T model and the RNN-T model with contextual LM (on-the-fly rescoring) using the v3 model. The proposed model achieves 47.0% WERR while the model based on contextual LM achieves 27.0% WERR, with only slight regression on the general set. The context preference ranker which acts as prior knowledge improves the model performance.

Table 4 shows performance of CSC teacher model and 8-bit quantized student model on CPU (Xeon E5-2690 v4 @ 2.60GHz) with single thread. Both models are in ONNX format. The student model without rRanker and pRanker feeds the full context list into CSC model without context preselection operation. Which shows with teacher-student learning, quantization and context filtering algorithm, the overall runtime performance of the model is greatly improved.

4.3. OOV contexts

The result of out of vocabulary (OOV) items is also important for contextual biasing, which indicates the model performance in real online scenario. We define OOV rate as the fraction of contexts in test set that are not seen during training and measure from three aspects: full name OOV rate (e.g., the full name "*Joe Biden*" not in training set), one name word OOV rate (e.g., "*Joe*" in training set while "*Biden*" is not in training set), and all name words OOV rate (e.g., neither "*Joe*" nor "*Biden*" is in training set), denoted as OOV1, OOV2, OOV3. These OOV rates for the name set are 54%, 15.9% and 5%, respectively. Table 5 shows model performance for OOV items and its complementary set, denoted as OOV_C. The CSC model improves more on OOV items than on context phrases covered in training. One of the reasons is that the baseline WER for the OOV items of the baseline RNN-T model is larger due to more contextual

Table 4: *Runtime performance for teacher/student*

Model	Size(MB)	WER	Latency (ms/utterance)
Teacher	223.0	15.6	793.6
Student	9.5	16.1	132.3
Student w/o r(p)Ranker	9.5	19.3	1846.7

Table 5: *Model performance for OOV items, WER(WERR)*

OOV	RNN-T	RNN-T+CSC
OOV1	37.2	16.5(55.6%)
OOV1_C	22.5	15.4(31.6%)
OOV2	43.3	19.1(55.9%)
OOV2_C	27.3	15.3(44.0%)
OOV3	37.1	16.9(54.4%)
OOV3_C	30.2	16.0(47.0%)

spelling errors.

4.4. Context list size

During inference, the input context list size affects the model performance. We have two context list sizes: the raw context list size K_r before filter and the filtered context list size K_f. Figure 2(a) illustrates the influence of K_r to model performance, which shows the WER increases when the raw context list size becomes larger. Figure 2(b) illustrates the influence of K_f to model performance, we can see the WER first drops and then increases when K_f becomes larger, this is because when K_f is set to small, the filter may miss the ground-truth context. There is always a best value for filtered context list size K_f, which depends on the context filter and context list distribution.

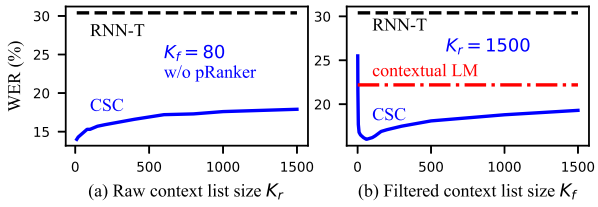

Figure 2: *Influence of context list size to model performance*

5. Conclusions

In this work, we introduce a novel light-weight CSC model for customizing the transducer-based ASR systems. The context information is integrated with a context encoder to a spelling correction model. Novel filtering algorithms are designed to deal with the large size context list. Encoder sharing, teacher-student learning, and quantization are used to achieve low runtime cost. Experiments show that the CSC model significantly outperforms the contextual LM biasing method, with around 50% relative WER reduction from the general RNN-T model on the name recognition set, while with only slight regression on the general set. The model also shows excellent performance for OOV terms not seen during training. Although the proposed method is verified with RNN-T in this study, it can also be applied to any transducer models.

6. References

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

[2] Y. He, T. N. Sainath, R. Prabhavalkar, I. McGraw, R. Alvarez, D. Zhao, D. Rybach, A. Kannan, Y. Wu, and R. Pang, "Streaming end-to-end speech recognition for mobile devices," in *Proc. ICASSP*, 2019.

[3] J. Li, R. Zhao, H. Hu, and Y. Gong, "Improving RNN transducer modeling for end-to-end speech recognition," in *Proc. ASRU*, 2019.

[4] G. Saon, Z. Tüske, and K. Audhkhasi, "Alignment-length synchronous decoding for RNN transducer," in *Proc. ICASSP*, 2020, pp. 7804–7808.

[5] A. Zeyer, A. Merboldt, R. Schlüter, and H. Ney, "A new training pipeline for an improved neural transducer," in *Proc. Interspeech*, 2020.

[6] C.-F. Yeh, J. Mahadeokar, K. Kalgaonkar, Y. Wang, D. Le, M. Jain, K. Schubert, C. Fuegen, and M. L. Seltzer, "Transformer-transducer: End-to-end speech recognition with self-attention," *arXiv preprint arXiv:1910.12977*, 2019.

[7] Q. Zhang, H. Lu, H. Sak, A. Tripathi, E. McDermott, S. Koo, and S. Kumar, "Transformer transducer: A streamable speech recognition model with transformer encoders and RNN-T loss," in *Proc. ICASSP*, 2020, pp. 7829–7833.

[8] X. Chen, Y. Wu, Z. Wang, S. Liu, and J. Li, "Developing real-time streaming transformer transducer for speech recognition on large-scale dataset," *arXiv preprint arXiv:2010.11395*, 2020.

[9] J. Li, , R. Zhao, Z. Meng *et al.*, "Developing RNN-T models surpassing high-performance hybrid models with customization capability," in *Proc. Interspeech*, 2020.

[10] K. Hall, E. Cho, C. Allauzen, F. Beaufays, N. Coccaro, K. Naka-jima, M. Riley, B. Roark, D. Rybach, and L. Zhang, "Composition-based on-the-fly rescoring for salient n-gram biasing," in *Proc. Interspeech*, 2015.

[11] I. Williams, A. Kannan, P. Aleksic, D. Rybach, and T. Sainath, "Contextual speech recognition in end-to-end neural network systems using beam search," in *Proc. Interspeech*, 2018.

[12] Y. He, T. N. Sainath, R. Prabhavalkar, I. McGraw, R. Alvarez, D. Zhao, D. Rybach, A. Kannan, Y. Wu, R. Pang *et al.*, "Streaming end-to-end speech recognition for mobile devices," in *Proc. ICASSP*, 2019, pp. 6381–6385.

[13] D. Zhao, T. N. Sainath, D. Rybach, D. Bhatia, B. Li, and R. Pang, "Shallow-fusion end-to-end contextual biasing," in *Proc. Interspeech*, 2019.

[14] M. Jain, G. Keren, J. Mahadeokar, and Y. Saraf, "Contextual RNN-T for open domain ASR," in *Proc. Interspeech*, 2020.

[15] D. Le, G. Keren, J. Chan, J. Mahadeokar, C. Fuegen, and M. L. Seltzer, "Deep shallow fusion for RNN-T personalization," in *Proc. SLT*, 2020.

[16] J. Guo, T. N. Sainath, and R. J. Weiss, "A spelling correction model for end-to-end speech recognition," in *Proc. ICASSP*, 2019, pp. 5651–5655.

[17] S. Zhang, M. Lei, and Z. Yan, "Investigation of transformer based spelling correction model for CTC-based end-to-end Mandarin speech recognition," in *Proc. Interspeech*, 2019.

[18] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[19] J. Li, R. Zhao, J.-T. Huang, and Y. Gong, "Learning small-size DNN with output-distribution-based criteria." in *Proc. Interspeech*, 2014, pp. 1910–1914.

[20] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[21] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[22] M. Mehryar, F. Pereira, and M. Riley, "Weighted finite-state transducers in speech recognition," *Computer Speech Language: 16.1*, pp. 69–88, 2002.

[23] M. Schuster and N. Kaisuke, "Japanese and korean voice search," in *Proc. ICASSP*, 2012.

[24] G. Pundak, T. N. Sainath, R. Prabhavalkar, A. Kannan, and D. Zhao, "Deep context: end-to-end contextual speech recognition," in *Proc. SLT*, 2018, pp. 418–425.

[25] W. Chan, N. Jaitly, Q. Le, and O. Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *Proc. ICASSP*. IEEE, 2016, pp. 4960–4964.

[26] D. Le, T. Koehler, C. Fliegen, and M. Seltzer, "G2g: Tts-driven pronunciation learning for graphemic hybrid asr," in *Proc. ICASSP*, 2020, pp. 6869–6873.

[27] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[28] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *arXiv preprint arXiv:1409.3215*, 2014.

[29] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, 2017, pp. 6000–6010.

[30] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[31] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," *OpenAI blog 1.8*, 2019.

[32] D. Nadeau and S. Sekine, "A survey of named entity recognition and classification," *Lingvisticae Investigationes: 30.1*, pp. 3–26, 2007.

[33] L. Jing, A. Sun, J. Han, and C. Li, "A survey on deep learning for named entity recognition," *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[34] A. Bruguier, R. Prabhavalkar, G. Pundak, and T. N. Sainath, "Phoebe: Pronunciation-aware contextualization for end-to-end speech recognition," in *Proc. ICASSP*, 2019, pp. 6171–6175.

[35] Y. He, P. Rohit, R. Kanishka, L. Wei, B. Anton, and M. Ian, "Streaming small-footprint keyword spotting using sequence-to-sequence models," in *Proc. ASRU*, 2017.

[36] Y. Ren, Y. Ruan, X. Tan, T. Qin, S. Zhao, Z. Zhao, and T.-Y. Liu, "Fastspeech: Fast, robust and controllable text to speech," in *Advances in Neural Information Processing Systems*, 2019, pp. 3137–3181.

[37] J.-M. Valin and J. Skoglund, "LPCNet: Improving neural speech synthesis through linear prediction," in *Proc. ICASSP*, 2019, pp. 5891–5895.

[38] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.