INTERSPEECH 2023
20-24 August 2023, Dublin, Ireland

Model-Internal Slot-triggered Biasing for Domain Expansion in Neural Transducer ASR Models

Yiting Lu[1,2*], *Philip Harding*[2], *Kanthashree Mysore Sathyendra*[2], *Sibo Tong*[2], *Xuandi Fu*[2], *Jing Liu*[2], *Feng-Ju Chang*[2], *Simon Wiesler*[2], *Grant P. Strimel*[2]

[1]Cambridge University, UK [2]Amazon Alexa

ytl28@cam.ac.uk, pjhard@amazon.co.uk

Abstract

Personal rare word recognition is an important yet challenging task for end-to-end speech recognition. Contextual biasing has demonstrated success in tackling this problem. Though effective in improving rare word recognition, these mechanisms can lead to errors due to false-biasing while facing further challenges when attempting to expand them to many domains. To address these limitations, in this work we propose a neural biasing design with a streaming model-internal slot classifier, trained to categorise the domain of each word piece before it is emitted. The neural biasing module can therefore be triggered in a controlled way, permitting natural scaling to many domains while reducing false-biasing and computational cost. Experiments on diverse domain slot types of application names, communications and playlist names demonstrate the proposed architecture results in 26% to 58% relative improvements on personal rare word recognition with minimal impact (0.6% rel.) on general data.

Index Terms: speech recognition, RNN-T, neural-transducer, contextual biasing, personalisation

1. Introduction

Despite recent advances in end-to-end automatic speech recognition (ASR) [1, 2], ASR systems still face challenges in accurately recognising rare words and phrases. Rare words are words that are not represented sufficiently in the training data, such as proper nouns or technical terms [3]. Personal rare word recognition [4, 5] is especially important for voice assistant applications since it allows users to interact with the system naturally, with ASR capable of catering to each user's personal context. This adaptation improves recognition for each user's customised preferences and therefore the application as a whole can execute personalised requests.

One standard approach to tackle rare word recognition is to adopt language model (LM) shallow fusion [6] at inference time. However, this method requires training of external language models and is also particularly sensitive to an LM fusion coefficient. Moreover, shallow fusion does not account for acoustic information which provides a useful signal for determining when to bias, especially when given an ambiguous textual prefix. Neural biasing (NB) [7, 8, 9, 10, 11, 12] is another commonly adopted approach for contextual biasing, where internal model states are updated using dynamic context (eg. user's playlists, devices etc.) in order to target the recognition process towards specific outcomes. NB is trained in conjunction with the main neural ASR architecture and usually outperforms shallow fusion [10, 11]. However, scaling NB to multiple domains/slots (termed '*horizontal scaling*') can pose significant challenges, due to increased catalogue sizes and inference latency. Furthermore, in most NB approaches, biasing is active for all time frames contributing to increased latency. To address these issues, Slot-Triggered Biasing (STB) [13] was proposed. STB augments the ground-truth ASR transcripts with special tags to indicate the start and end of biasing for different slots, ensuring that biasing is activated only when predicting slot content. However, STB requires retraining the ASR model from scratch due to the added special tags.

In this work, we propose a streaming model-internal slot classifier to horizontally scale contextual biasing to multiple domains. The classifier is conditioned on the internal states of the ASR model and is trained to predict the slot-type for the word piece being predicted. The neural biasing module can therefore be triggered in a controlled manner using the internal gating mechanics. Additionally, the predicted slot-types can be used to determine when biasing is required and also to narrow down the catalogues used for biasing, resulting in reduced inference latency. Our modular approach also allows for easy integration of the classifier component with a pretrained core ASR model without requiring retraining. The proposed approach typically achieves higher accuracy on personalised entities than standard attention-based biasing [10] and comparable accuracy to STB, while minimising degradation on general utterances. We also demonstrate that the proposed model-internal slot classifier is able to strike a balance between high accuracy and low latency by running a top-K/probability controlled domain selection during inference.

2. Contextual Biasing

The contextual adapter proposed in [10, 11] is used as a building block in this work to achieve contextual biasing for recurrent neural network transducer (RNN-T) ASR models [1]. The central idea of this approach is to influence the ASR model predicted posterior by adding an augmenting biasing vector to the encoder and/or decoder states according to the catalogue information. The design comprises two components: a catalogue encoder, and a biasing adapter.

Catalogue Encoder The catalogue encoder transforms a list of contextual entities $\boldsymbol{C} = [c_1, c_2, ..., c_K]$ into a set of embedding representations. This is accomplished with a bidirectional LSTM layer, which encodes the word-piece tokenised catalogue entities into embeddings $\boldsymbol{C}^e$.

Biasing Adapter The biasing adapter (BA) accounts for the biasing information generated from the catalogue encoder as well as the encoder and/or decoder states, and adapts the intermediate representations from the neural transducers (encoder and/or decoder states) to bias towards the word pieces of in-

*this work was carried out during an internship at Amazon Alexa

10.21437/Interspeech.2023-1010

terest. The adapter is implemented with a multi-head attention mechanism [14], which attends over the encoded list of contextual entities to produce the biasing vectors that are added to the encoder and/or decoder states. In this work, the encoder representation $\boldsymbol{h}_t$ is used as the query to generate the biasing vector $\boldsymbol{b}_t$, which is added back to the encoder representations:

$$\boldsymbol{b}_t = \text{BA}(\boldsymbol{h}_t, \boldsymbol{C}^e); \quad \hat{\boldsymbol{h}}_t = \boldsymbol{h}_t + \boldsymbol{b}_t \tag{1}$$

An additional <no_bias> token is added to all catalogue lists similar to [9, 10], allowing the BA to output a low-magnitude biasing vector when an audio frame does not require biasing.

Practical Considerations and Limitations The standard contextual adapter is trained with the core RNN-T frozen and with the biasing mechanism activated for all audio frames, similar to [10]. It relies on the <no_bias> token to implicitly suppress biasing in order to limit false-biasing. Also, the vanilla contextual adapter operates with only a single catalogue list. However, to achieve biasing towards different domains, one must account for multiple catalogues. When scaling to multiple catalogues, a naïve approach would be to concatenate all catalogues into a single one. However, this strategy makes the catalogue list become much longer, leading to significant latency implications, increased confusions and signal dilution. Our model introduced in Section 4 trains a model-internal slot classifier, which aims to – (1) allow easy scaling to multiple slot catalogues, while maximising the accuracy gains provided by the contextual signals, and (2) limit when biasing is active, thus reducing latency.

3. Related Works

STB [13] was proposed to address the limitations of the contextual adapters. STB selectively activates the biasing module by augmenting the ASR transcript with slot opening and closing tags. The tags can then be used to activate / deactivate the contextual adapter. While STB was previously shown to be an effective method of scaling contextual biasing to multiple domains, the method has some drawbacks. For instance, the base ASR model must be retrained every time a new slot type is enabled. Meanwhile, the base ASR accuracy is not guaranteed to be preserved since the slot prediction is entangled with the ASR prediction.

Another related work is on Multi-Task Spoken Language Understanding (SLU) [15] where the RNN-T model jointly predicts the ASR transcripts and slot-types associated with each word-piece using a streaming slot-classifier trained using cross-entropy loss.

4. Model-Internal Slot Classifier

In our work, we propose to train a model-internal slot classifier that determines the relevant slot type (domain) of the word piece to be generated. This is used to guide the encoder based contextual adapter to activate biasing accordingly. The slot classifier also acts as a filtering/selection mechanisms when multiple slot catalogues are used. In addition to supported slot-types, we also introduce the `other` class to our slot-classifier. When the predicted slot is the `other` class (i.e. common words), biasing can be explicitly shut down to avoid false-biasing. When the predicted slot is not `other`, biasing is activated for the slot catalogue that corresponds to the predicted slot. By limiting biasing to the selected catalogues, the contextual adapter can easily horizontally scale across multiple slot catalogues without increasing model latency. As shown in Figure 1, the model-internal

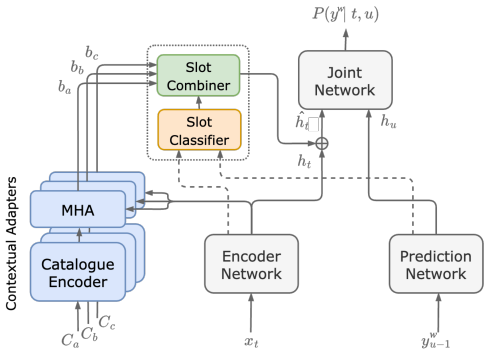

Figure 1: *Model-Internal Multi-Task Contextual Adaptor.*

slot classifier has two components: the slot classifier generates a probability distribution over slot types; and the slot combiner determines how to make use of the classifier's slot posterior to choose among the biasing vectors $\{\boldsymbol{b}_a, \boldsymbol{b}_b, \boldsymbol{b}_c\}$ corresponding to the slot catalogues $\{\boldsymbol{C}_a, \boldsymbol{C}_b, \boldsymbol{C}_c\}$. The light-weight model-internal slot classifier can be trained separately from the core RNN-T, and allows for natural scaling to multiple slot types, without re-training models from scratch.

4.1. Slot Classifier Input Features

The slot classifier can be conditioned on the encoder state $\boldsymbol{h}_t$, the decoder state $\boldsymbol{h}_u$, or a combination of the two. In this work, simple addition was adopted for the join operation in the classifier. The encoder states have length T, the decoder network states have length U, and the combination of the two leads to a grid state space of $T \times U$.

4.2. Slot Classifier Architectures

The simplest approach is to use a stateless classifier, i.e. feed-forward network. Alternatively, stateful layers such as LSTM can encode historical context which may be beneficial for slot classification. Applying recurrent layers to encoder- or decoder-conditioned models is straightfoward, but is complex when conditioning classification on joint states. To allow us to make use of both joint network states and stateful model layers we split the classifier network into decoder-only and encoder-only sub-networks, the outputs of which are passed through the joint operation and then projected to the number of slot classes.

4.3. Training Loss

The neural biasing training mainly consists of two main components: the slot classifier and the contextual adapter. The core RNN-T model (encoder, decoder and joint networks shaded in grey in Figure 1) is kept frozen throughout the biasing training. In this work, the slot classifier and the contextual adapter were jointly trained using a combination of the RNN-T loss and cross-entropy loss for slot-prediction [15].

Cross entropy (CE) Loss Each word-piece y^w_u is associated with a slot type $y^s_u \in \mathcal{S}$ where $\mathcal{S}$ is the set of supported slot types along with the `other` class. We apply one of the following CE loss formulations to supervise the slot classifier training:

$$\mathcal{L}_{ce} = -\tfrac{1}{U} \textstyle\sum_{u=0}^{U} y^s_u \log P(y^s_u|\boldsymbol{h}_u) \tag{2}$$

$$\mathcal{L}_{ce} = -\tfrac{1}{U} \textstyle\sum_{u=0}^{U} \tfrac{1}{T} \sum_{t=0}^{T} y^s_u \log P(y^s_u|\boldsymbol{h}_t) \tag{3}$$

$$\mathcal{L}_{ce} = -\tfrac{1}{U} \textstyle\sum_{u=0}^{U} \tfrac{1}{T} \sum_{t=0}^{T} y^s_u \log P(y^s_u|\boldsymbol{h}_u, \boldsymbol{h}_t) \tag{4}$$

Here, y^s_u is the slot type associated with word-piece y^w_u; $\boldsymbol{h}_t$, $\boldsymbol{h}_u$ are encoder and decoder states respectively. For the encoder

only (eqn. (3)), and the encoder plus decoder network conditioned classifiers (eqn. (4)), the CE loss is averaged over all time steps since there is no time-aligned slot annotation available.

To further enhance time alignment, we also derive a variation of the CE loss where the loss is weighted along the time axis according to an alignment coefficient $\lambda_{t,u}$.

$$\mathcal{L}_{ce\text{-}\alpha} = -\frac{1}{U}\sum_{u=0}^{U}\sum_{t=0}^{T}\lambda_{t,u}y_u^s \log P(y_u^s|\boldsymbol{h}_u, \boldsymbol{h}_t) \quad (5)$$

The forward state used for the RNN-T loss calculation ($\alpha_{t,u} = P(y_{1:u}^w|\boldsymbol{x}_{1:t})$ as defined in [1]) roughly highlights the most probable path in the $T \times U$ grid. It can therefore be used to guide the CE loss when ground truth alignment is not available. In this work, the vanilla RNN-T logits are used to compute an approximated $\hat{\alpha}_{t,u}$, and the alignment coefficient $\lambda_{t,u}$ is a normalised version of $\hat{\alpha}_{t,u}$ over the time axis T:

$$\lambda_{t,u} = \frac{\hat{\alpha}_{t,u}}{\sum_{t=0}^{T}\hat{\alpha}_{t,u}}; \quad \text{where} \quad \hat{\alpha}_{t,u} = P(y_u^w|\boldsymbol{x}_{0:t}) \quad (6)$$

RNN-T Loss Following the contextual adapter training discussed in [10], the standard RNN-T loss was adopted. In order to use RNN-T loss ($\mathcal{L}_{rnnt}$) to implicitly constrain the slot classifier, a soft slot combination approach was adopted during training (as discussed in Section 4.4) and the RNN-T loss is expected to complement the CE loss to further improve the classifier.

4.4. Slot Combiner

Soft Combination The slot classifier generates a slot probability distribution over $\mathcal{S}$ given the encoder and/or decoder states, which can be used as weights to combine the biasing vectors:

$$\boldsymbol{b}_{u,t} = \sum_{s=1}^{|\mathcal{S}|} P(y_u^s|\boldsymbol{h}_u, \boldsymbol{h}_t)\, \boldsymbol{b}_{u,t}^s; \quad \boldsymbol{b}_{u,t}^s = \text{BA}(\boldsymbol{h}_t, \boldsymbol{C}_s^e) \quad (7)$$

where $\boldsymbol{b}_{u,t}$ is the final biasing vector at time t step u; the weight coefficient $P(y_u^s|\boldsymbol{h}_u, \boldsymbol{h}_t)$ is the slot posterior and $\boldsymbol{C}_s^e$ is the associated slot catalogue embeddings. Soft Combination is used during training since it is differentiable and allows gradients to back propagate through the classifier during training.

Top-K / Probability Cap Selection During inference, to reduce the computational complexity of cross-attention, we limit slot catalogues to only the most probable slots at each decoding step. We explore two options: a) **Top-K**: account for the top-K probable slots; b) **Probability Cap Selection (Pcap)**: account for slots which make up $x\%$ of the total probability mass. For case b), slot posteriors were first sorted from highest to lowest. The probability cap $x\%$ is then imposed by processing the probable slot types in-order until the probability cap is reached. After applying the slot selection criteria, the chosen slots were still weighted by the slot posterior to enforce consistency between training and inference.

5. Experimental Analysis

5.1. Dataset and Evaluation Metric

We used an in-house de-identified American English voice assistant dataset with each utterance consisting of audio, transcription and catalogues of contextual entities. Utterances were randomly sampled from the voice assistant traffic across more than 20 domains including Communications (Comms), SmartHome, and Music. We focused on applying contextual biasing to three domains - Comms, Music and Applications. For training contextual adapters, we split training data into 'personalised' and 'non-personalised' partitions based on whether utterances contain content from our target use-cases. Training sets contain a mixture of human- and machine- transcribed utterances. We used approximately 400 hours human-transcribed personalised data for Comms, 100 hours for Music, and 500 hours for application launch phrases. The non-personalised human-transcribed dataset contained approximately 66k hours. Slot annotations were obtained using an in-house entity tagger.

We used a mix of personalised and non-personalised data to train the contextual biasing adapters, with exact weights for each system stated in the results section. Where a model was trained to bias towards multiple slots, equal weighting was applied to data associated with each domain. We report results on a 34k utterance general dataset, and three personalised datasets - a 34k utterance Comms dataset, a 4.4k utterance Music dataset and a 22k utterance application launch dataset. Same as [11, 10], we report the relative word error rate reduction (WERR%) on the general dataset, and the relative slot word error rate reduction (WERR-S%) for named entities on the personalised datasets. In all cases the baseline model is an RNN-T model without contextual biasing.

5.2. Model Configurations

The input audio features are 64-dimensional log filter-bank energies extracted every 10 ms with a window size of 25 ms. Three consecutive frames are stacked, after which downsampling by a factor of three is applied, resulting in 192 feature coefficients per frame. Ground truth tokens are tokenised using a word-piece tokeniser with vocabulary size of 4000 [16, 17]. The RNN-T encoder network consists of five LSTM layers, each with 1280 units, with a time-reduction layer (downsampling factor=2) at layer three. The prediction network consists of two LSTM layers with 1024 units per layer. The outputs from the encoder and prediction network are projected through a feed-forward layer to 1024 units. We use a simple addition for the join operation, followed by the `tanh` activation before further projecting to 4001 units (vocabulary size + blank label) in the output layer. Decoding is performed using the standard RNN-T beam search [1] with a beam size of seven.

The feed-forward (FF) slot classifier adopts a hidden size of 256, which then projects to a distribution over four slot classes (Comms, AppName, PlaylistName, `other`). For contextual biasing we use the attention-based adapter as in [10]. The core RNN-T model was frozen during the slot classifier and contextual adapter training. The models were trained using the Adam optimiser and a warmup-hold-decay learning rate schedule with 3k steps warm-up, 72k steps hold, and 25k steps decay. We use a maximum learning rate of 8×10^{-4} and a minimum learning rate of 6.25×10^{-5} for training the unconstrained adapters, and a maximum learning rate of 1.6×10^{-3} when training models with internal slot classifiers. The contextual adapter has 608k parameters ($< 0.5\%$ of the base RNN-T parameters) while the FF and LSTM slot classifiers have 263k and 279k parameters, respectively. The maximum catalogue size for each domain is set to 300 during training and 5000 during inference to fit within memory.

5.3. Experimental Results

Table 1 compares the performance of the proposed methods to models with contextual biasing trained on individual catalogues (ID1-3), merged catalogues (ID4) [10] and slot-triggered biasing (ID5) [13]. When evaluating with soft slot combination we

Table 1: *Results using proposed slot classifier with soft slot selection.*

ID	Inputs	Loss	Architecture	Non-personalised data weight	General WERR (%)	AppName WERR-S (%)	Comms WERR-S (%)	PlaylistName WERR-S (%)
1	-	$\mathcal{L}_{rnnt}$	1 catalogue (App)	0.70	0.6	20.1	-	-
2	-	$\mathcal{L}_{rnnt}$	1 catalogue (Comms)	0.70	-2.0	-	43.1	-
3	-	$\mathcal{L}_{rnnt}$	1 catalogue (Playlist)	0.70	0.2	-	-	39.5
4	-	$\mathcal{L}_{rnnt}$	Merged catalogues	0.70	-1.6	22.5	37.1	49.1
5	-	$\mathcal{L}_{rnnt}$	Slot-Triggered Biasing	0.00	-1.2	33.3	45.1	49.6
6	enc+dec	$\mathcal{L}_{rnnt}$	FF	0.25	-2.2	35.0	45.4	54.8
7	enc+dec	$\mathcal{L}_{rnnt}$ + $\mathcal{L}_{ce}$	FF	0.25	-2.0	33.3	45.1	54.2
8	enc+dec	$\mathcal{L}_{rnnt}$ + $\mathcal{L}_{ce-\alpha}$	FF	0.25	-2.7	34.1	45.4	55.7
9	dec	$\mathcal{L}_{rnnt}$ + $\mathcal{L}_{ce}$	FF	0.25	-2.7	30.5	44.3	54.8
10	enc	$\mathcal{L}_{rnnt}$	FF	0.25	-2.5	32.9	45.2	57.6
11	enc+dec	$\mathcal{L}_{rnnt}$ + $\mathcal{L}_{ce}$	LSTM	0.25	-1.4	34.5	45.4	59.6
12	enc+dec	$\mathcal{L}_{rnnt}$ + $\mathcal{L}_{ce}$	LSTM	0.40	0.8	30.1	43.2	61.4

find that the models can be effectively trained with RNN-T loss ($\mathcal{L}_{rnnt}$) alone (ID6), with the introduction of cross-entropy loss ($\mathcal{L}_{ce}$) resulting in comparable accuracy on slot content, with marginally reduced degradation on general utterances (ID7). Soft slot combination requires all entities in all catalogues to be processed for every emission step. In Table 2 we compare models trained with the three different loss configurations on top-1, top-2 and Pcap slot selection. We find that cross-entropy loss is essential when selecting the top-1 slot type during decoding and that introducing time alignment information by weighting the cross-entropy loss by α ($\mathcal{L}_{ce-\alpha}$) further improves performance on slot content, but results in additional degradation on general data (ID8).

Table 2: *Comparison of feed-forward slot classification models trained with different losses in terms of Comms WERR-S (%).*

ID	Loss	Slot Selection Method			
		Soft	Top-1	Top-2	Pcap=0.8
6	$\mathcal{L}_{rnnt}$	45.6	-0.1	44.9	43.6
7	$\mathcal{L}_{rnnt}$ + $\mathcal{L}_{ce}$	45.3	42.4	45.0	44.9
8	$\mathcal{L}_{rnnt}$ + $\mathcal{L}_{ce-\alpha}$	44.9	45.0	45.2	45.1

Looking next at the features used as input to the slot classifier, we find that a model conditioned only on encoder states (ID10) performs on-par with the model trained on encoder and decoder states (ID7) in terms of WERR-S, but that the model trained using both states degrades less on general data. Conditioning the slot classifier only on decoder states is found to slightly degrade performance (ID9); the reason for this is the lack of semantic context in some utterances. Table 3 illustrates this in more detail, where WERR-S is broken down on Comms data based on whether there is any spoken context before the target entity. While decoder-conditioned slot classifiers perform strongly on utterances with spoken context, there is a clear difference on utterances without. We hypothesise that models conditioned on encoder states are able to take advantage of the tendency of the model to delay token emission [18, 19], with classification then based on some limited future acoustic context. STB is also found to perform well without context, suggesting it too benefits from acoustic lookahead.

Table 3: *Impact of availability of prior within-utterance context on Comms WERR-S ("call name" vs. "name").*

ID	Inputs	Architecture	With context	Without context
2	–	1 catalogue (Comms)	44.5	40.8
5	–	Slot-Triggered Biasing	46.5	43.9
7	enc+dec	FF	46.5	42.7
9	dec	FF	47.1	37.8
10	enc	FF	46.2	43.3

Next, we look at the architecture of the slot classifier itself. Accuracy across all three slots is improved with a stateful classifier (ID11), with degradation on general data also decreasing. Despite these improvements we still observe some degree of false biasing. We therefore tuned the weights applied to personalised and non-personalised data during training. By increasing the weight of non-personalised data from 0.25 to 0.4 (ID11 and ID12, respectively) we were able to eradicate degradation on general data at the cost of some reduced improvements on App and Comms slot content.

Finally, we evaluate the proposed model (ID12) with the different slot selection methods described in Section 4.4. Results are reported in Table 4. Top-1 slot selection is found to reduce improvements across all slot types. We found that the models sometimes assigned higher probability to the `other` class in cases where a particular slot type should have been predicted; with top-1 slot classification, a non-other slot prediction (which would result in biasing) was made in only 15% of emission steps for utterances in the App domain, i.e. an average of 0.15 catalogues were processed per emission step. For top-2 slot selection at least one catalogue will be processed per emission step, and an average of 1.01 catalogues was processed per emission step. A more dynamic method of selecting slots, expected to better generalise to larger sets of domains, is to select the top-K slots while the cumulative probability mass remains below a particular threshold (Pcap). With a threshold of 0.8, accuracy lies between Top-1 and Top-2, with 0.24 catalogues processed per emission step on average for App domain utterances.

Table 4: *Decoding strategy using LSTM-based model (ID12)*

Slot Selection	General WERR	AppName WERR-S	Comms WERR-S	Playlist WERR-S
Soft	0.8	30.1	43.2	61.4
Top-1	-0.6	23.3	39.9	55.7
Top-2	0.4	30.1	42.6	58.1
Pcap=0.8	-0.6	26.9	42.4	58.7

6. Conclusions

In this work, we proposed a model internal slot classifier which can be used to horizontally scale contextual biasing to multiple domains. The proposed method has a number of advantages compared to existing methods of horizontal scaling, achieving comparable accuracy on personal rare words while reducing the risk of degrading recognition accuracy on general data. The method is easy to train on top of an existing ASR model and is efficient at run-time; with the proposed Pcap slot selection method we demonstrated how, on average, 0.24 catalogues were processed per emission step with our proposed method when considering three domains.

7. References

[1] A. Graves, "Sequence Transduction with Recurrent Neural Networks," in *Proceedings of the 29th International Conference on Machine Learning (ICML)*, 2012.

[2] A. Gulati, J. Qin, C.-C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu *et al.*, "Conformer: Convolution-augmented Transformer for Speech Recognition," in *Proceedings of Interspeech*, 2020.

[3] C. Peyser, S. Mavandadi, T. N. Sainath, J. Apfel, R. Pang, and S. Kumar, "Improving Tail Performance of a Deliberation E2E ASR Model Using a Large Text Corpus," in *Proceedings of Interspeech*, 2020.

[4] T. N. Sainath, R. Prabhavalkar, S. Kumar, S. Lee, A. Kannan, D. Rybach, V. Schogol, P. Nguyen, B. Li, Y. Wu *et al.*, "No Need for a Lexicon? Evaluating the value of the Pronunciation Lexica in End-to-End Models," in *Proceedings of ICASSP*. IEEE, 2018, pp. 5859–5863.

[5] A. Bruguier, F. Peng, and F. Beaufays, "Learning Personalized Pronunciations for Contact Name Recognition," in *Proceedings of Interspeech*, 2016, pp. 3096–3100.

[6] D. Zhao, T. N. Sainath, D. Rybach, P. Rondon, D. Bhatia, B. Li, and R. Pang, "Shallow-Fusion End-to-End Contextual Biasing," in *Proceedings of Interspeech*, 2019.

[7] D. Le, G. Keren, J. Chan, J. Mahadeokar, C. Fuegen, and M. L. Seltzer, "Deep Shallow Fusion for RNN-T Personalization," in *Proceedings of the IEEE Workshop on Spoken Language Technology*, 2021.

[8] M. Jain, G. Keren, J. Mahadeokar, G. Zweig, F. Metze, and Y. Saraf, "Contextual RNN-T for Open Domain ASR," in *Proceedings of Interspeech*, 2020.

[9] G. Pundak, T. N. Sainath, R. Prabhavalkar, A. Kannan, and D. Zhao, "Deep context: end-to-end contextual speech recognition," in *Proceedings of the IEEE Workshop on Spoken Language Technology*, 2018.

[10] K. M. Sathyendra, T. Muniyappa, F.-J. Chang, J. Liu, J. Su, G. P. Strimel, A. Mouchtaris, and S. Kunzmann, "Contextual Adapters for Personalized Speech Recognition in Neural Transducers," in *Proceedings of ICASSP*, 2022.

[11] F.-J. Chang, J. Liu, M. Radfar, A. Mouchtaris, M. Omologo, A. Rastrow, and S. Kunzmann, "Context-Aware Transformer Transducer for Speech Recognition," in *2021 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 2021, pp. 503–510.

[12] G. Sun, C. Zhang, and P. C. Woodland, "Minimising Biasing Word Errors for Contextual ASR With the Tree-Constrained Pointer Generator," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 31, pp. 345–354, 2023.

[13] S. Tong, P. Harding, and S. Wiesler, "Slot-triggered Contextual Biasing for Personalized Speech Recognition using Neural Transducers," in *Proceedings of ICASSP*, 2023.

[14] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is All You Need," *Advances in neural information processing systems*, 2017.

[15] X. Fu, F.-J. Chang, M. Radfar, K. Wei, J. Liu, G. P. Strimel, and K. M. Sathyendra, "Multi-Task RNN-T with Semantic Decoder for Streamable Spoken Language Understanding," in *Proceedings of ICASSP*. IEEE, 2022, pp. 7507–7511.

[16] R. Sennrich, B. Haddow, and A. Birch, "Neural Machine Translation of Rare Words with Subword Units," in *Proceedings ACL*, 2015.

[17] T. Kudo, "Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates," in *Proceedings ACL*, 2018.

[18] A. Tripathi, H. Lu, H. Sak, and H. Soltau, "Monotonic Recurrent Neural Network Transducer and Decoding Strategies," in *Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding*, 2019, pp. 944–948.

[19] J. Mahadeokar, Y. Shangguan, D. Le, G. Keren, H. Su, T. Le, C.-F. Yeh, C. Fuegen, and M. L. Seltzer, "Alignment Restricted Streaming Recurrent Neural Network Transducer," in *Proceedings of the IEEE Workshop on Spoken Language Technology*, 2021, pp. 52–59.