INTERSPEECH 2023
20-24 August 2023, Dublin, Ireland

Selective Biasing with Trie-based Contextual Adapters for Personalised Speech Recognition using Neural Transducers

Philip Harding, Sibo Tong and Simon Wiesler

Amazon Alexa

pjhard@amazon.co.uk, tongsib@amazon.co.uk, wiesler@amazon.de

Abstract

Neural transducer ASR models achieve state of the art accuracy on many tasks, however rare word recognition poses a particular challenge as models often fail to recognise words that occur rarely, or not at all, in the training data. Methods of contextual biasing, where models are dynamically adapted to bias their outputs towards a given list of relevant words and phrases, have been shown to be effective at alleviating this issue. While such methods are effective at improving rare word recognition, over-biasing can lead to degradation on common words. In this work we propose several extensions to a recently proposed trie-based method of contextual biasing. We show how performance of the method can be improved in terms of rare word recognition, especially in the case of very large catalogues, by introducing a simple normalisation term, how the method can be trained as an adapter module, and how selective biasing can be applied to practically eliminate over-biasing on common words.
Index Terms: speech recognition, contextual biasing, personalisation

1. Introduction

Neural transducer automatic speech recognition (ASR) models such as recurrent neural network transducer (RNN-T) [1] have been widely adopted due to their ability to achieve state of the art performance on a variety of tasks. While recognition accuracy on common words is typically high, such models often fail to reliably recognise words that were seen rarely, or not at all, in the training data [2]. Rare word recognition is particularly important for applications that rely on dynamic content, including calling and messaging (contact names) and entertainment (song titles, playlist names) where target entities are often also rare words. Contextual data is sometimes available in these use-cases, with such data being either global (e.g. media catalogues) or user-specific (e.g. address books, favourite song titles). The challenge then becomes how to best make use of this data.

Previously, 'hybrid' ASR systems could be adapted to recognise new words by updating pronunciation lexicons and language models using text-only data, with grapheme-to-phoneme models used to generate pronunciations. This approach is not possible with neural transducer models trained to directly estimate text from acoustic features. Language-model adaptation can still be beneficial, but the impact is often less significant; first-pass language model fusion methods such as shallow fusion can be sensitive to combination weights [3, 4], while second pass language model rescoring relies on words existing in n-best lists generated by the first pass [5].

In the case of global catalogues, models can be updated offline by fine-tuning on synthetically generated utterances containing target words [6, 7, 8, 9]. When contextual data is more dynamic, e.g. user-specific playlist names, models need to be updated at run-time where fine-tuning is computationally infeasible. In these cases contextual biasing (CB) can be used to dynamically update internal model states at run-time such that the output distribution of the model is biased towards emitting the given words [10, 11, 12, 13, 14, 15]. As well as the given words, updates are also typically conditioned on the recognition prefix or internal model states. CB has been shown to be highly effective, enabling neural transducer models to emit previously unseen words, but challenges remain. In particular, recognition accuracy on common words can be degraded by over-biasing, especially as the number of words used for biasing grows.

In this work, we focus on trie-based CB [10, 11] and propose a number of extensions including selective biasing [16, 17] and adapter training [12, 18] to alleviate over-biasing, and a simple normalisation term to improve robustness to large biasing lists. Our proposed approach is shown to practically eliminate over-biasing while improving accuracy on rare words.

2. Contextual Biasing

Several methods of contextual biasing have been proposed in the literature, with most methods following the same basic approach [10, 11, 12, 13, 14, 15]. Given a list of variable-length contextual entities, $C = [c_0, c_1, \ldots, c_k]$, that have been tokenised to match the ASR model output vocabulary, a function, f^{emb}, is used to generate entity embeddings:

$$C^e = [c_0^e, c_1^e, \ldots, c_k^e] \; ; \;\; c_k^e = f^{emb}\left(c_k\right) . \tag{1}$$

The resulting embeddings are either fixed-dimensional vectors encoding an entire contextual entity in a single vector [12, 14], or variable length sequences of embeddings where embeddings are computed for each element of the tokenised sequence [10, 11, 13]. Given a context vector $\mathbf{z}_s$ relating to state s, a biasing vector $\mathbf{b}_s$ is then computed and added to model state vector $\mathbf{h}_s$:

$$\mathbf{b}_s = f^{bias}(C^e, \mathbf{z}_s); \quad \hat{\mathbf{h}}_s = \mathbf{h}_s + \mathbf{b}_s \, . \tag{2}$$

$\mathbf{z}_s$ is typically either the hidden state vector [12], i.e. $\mathbf{z}_s = \mathbf{h}_s$, or is the partial recognition prefix [10]. $\mathbf{h}_s$ can be the encoder, prediction network or joint state vectors.

2.1. Trie-based contextual biasing

Trie-based contextual biasing was recently proposed as a method of contextualising prediction network states, with biasing conditioned on the partial recognition prefix y_s at state s [10, 11, 14]. Tokenised contextual entities, C, are encoded in a trie before recognition begins, as illustrated in Figure 1. During recognition, the trie is queried at each decoder state (s) to obtain two sets of symbols to bias the model towards: those

10.21437/Interspeech.2023-739

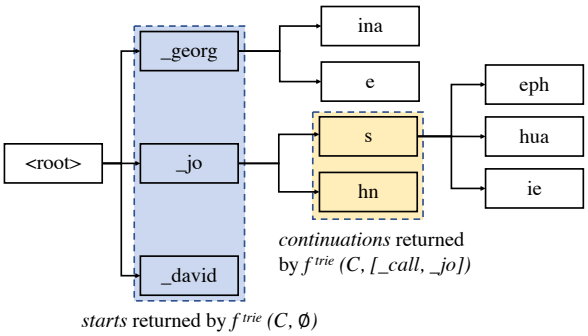


Figure 1: *Example trie built from the list of contact names* $C = [georgina, george, john, joseph, joshua, josie, david]$*, illustrating starts and continuations sets for the partial recognition prefix "call jo"*

that can start a new term in C (*starts*), and those that can extend y_s to continue a term in C (*continuations*). The corresponding embeddings for all symbols in the set of starts and continuations are then retrieved and combined to form the biasing vector used to update the prediction network output vector.

More formally, the function $f^{trie}(C, y)$ queries the trie built from C, using suffixes of y with length up to $\min(U, H)$, where U is the length of y and H is a hyper-parameter used to limit the maximum suffix length. The set of *starts* is returned when $y_s = \emptyset$. Given a symbol index, i, and embedding matrix W, $f^{emb}(i, W)$ performs a simple embedding lookup. $\mathbf{b}_s$ is then the sum of embeddings selected by f^{trie} and returned by f^{emb}:

$$f^{bias}(C, y_s) = \sum_m f^{emb}\left(f^{trie}\left(C, \emptyset\right)_m, W^{starts}\right) + \sum_m f^{emb}\left(f^{trie}\left(C, y_s\right)_m, W^{cont}\right), \quad (3)$$

where m denotes the index of a word-piece returned by f^{trie}. Embedding matrices W^{starts} and W^{cont} are learnt alongside other model parameters. Dimensions of embeddings and prediction network hidden state vectors must be the same.

We propose several modifications to this approach with the aim of improving robustness, eliminating over-biasing, and reducing the memory footprint of the method. These are described in the following sections.

2.1.1. Adapter training

In [10], joint network weights are re-trained from scratch alongside the embedding matrices W^{starts} and W^{cont} starting from a model trained without contextual biasing. In this work we propose freezing the entire baseline model and training the contextual biasing components as adapters as per [12]. Adapter training is expected to make it easier to preserve baseline model performance and reduces the number of trained parameters.

2.1.2. Projection layer

We propose decoupling the embedding dimension from the prediction network hidden state dimension by adding a fully-connected layer with swish activation function [19] after f^{bias}:

$$\mathbf{b}_s = \text{swish}\left(W^{proj} \cdot f^{bias}(C^e, \mathbf{y}_s)\right). \quad (4)$$

We do not use a bias term to ensure all elements of $\mathbf{b}_s$ remain zero when f^{bias} is also zero. This layer is expected to be beneficial in two ways: i.) the dimensionality of embeddings can be reduced, and ii.) the non-linear transformation should help the model to converge with adapter-style training.

2.1.3. Normalised embeddings

In existing works, embedding vectors are not scaled before summation [10, 11, 13]. The magnitude of biasing vectors therefore increases proportionally to the number of *starts* (N^{starts}) and *continuations* (N^{cont}) returned by f^{trie}. This is in contrast to the attention-based method proposed in [12] where biasing vectors are computed as a weighted average of entity embedding vectors. Motivated by this approach, we propose normalising the summation in (3) using uniform weighting, i.e.:

$$\hat{f}^{bias}(C^e, \mathbf{y}_s) = \frac{f^{bias}(C^e, y_s)}{N^{starts} + N^{cont}}. \quad (5)$$

This approach could be further exploited if entity priors were available, e.g. frequency of played song titles.

2.1.4. Continuation-only biasing

In Eq. (3), $f^{trie}\left(C^e, \emptyset\right)$ is invariant to the active recognition prefix, y_s. For each utterance, this results in a constant factor being added to internal model states and may cause output probabilities to be biased in cases where biasing is not required. We therefore analyse the impact of dropping the term relating to *starts* from Eq. (3).

2.1.5. Shared embeddings

By decoupling the embedding dimensionality from the prediction network state dimension we can re-use prediction network input embedding weights, W^{pred}, for biasing, i.e. $W^{starts} = W^{cont} = W^{pred}$, reducing the number of trainable parameters.

2.1.6. Slot-triggered contextual biasing

Selective biasing uses cues from the model to determine when biasing is necessary so that biasing can be disabled when it is not required to reduce over-biasing. Different forms of selective biasing have been explored in literature, e.g., gated adapters [17], where an additional component is added to the model and trained to predict the probability of biasing being necessary for a particular frame. In this work, we explore the use of slot-triggered biasing [16, 20], where the neural transducer is trained to emit opening and closing tags around words or phrases that should be biased. For example, in the case of contact name biasing the model might predict `call op_ContactName phil cl_ContactName`. During decoding the contextual biasing adapter is only activated after an opening tag has been emitted by the model. In all other cases, the biasing vector $\mathbf{b}_s$ is set to zero. Biasing remains active until the corresponding closing tag is emitted by the model. When training models for use with slot-triggered biasing we apply masking to the biasing states to match conditions between training and inference as described in [16].

3. Experimental Analysis

3.1. Dataset and evaluation metric

We used an in-house American English voice assistant dataset, with each utterance consisting of audio, transcription, and a list of contextual entities. The training data is not associated with identifying information, but some utterances contain named entities. Utterances were randomly sampled from the voice assis-

Table 1: *Ablation study of proposed improvements. $\mathcal{S}$ and $\mathcal{C}$ refer to starts and continuations vectors defined in Section 2.1.*

ID	Method	Proj. layer	Embedding size	Normalise embeddings	Shared embeddings	Freeze joint	Target context dropout (P_τ)	Trainable params	General WERR (%)	Named entities WERR-S (%)
0	AttnEnc	-	64	✓	✗	✓	0.0	0.6M	-0.2	43.2
1	Trie($[\mathcal{S};\mathcal{C}]$)	✗	1024	✗	✗	✗	0.5	12.3M	-3.5	22.8
2	Trie($[\mathcal{S};\mathcal{C}]$)	✗	1024	✗	✗	✓	0.5	8.2M	-18.8	15.6
3	Trie($[\mathcal{S};\mathcal{C}]$)	✓	512	✗	✗	✓	0.5	4.6M	-2.9	15.9
4	Trie($[\mathcal{S};\mathcal{C}]$)	✓	512	✗	✗	✓	0.0	4.6M	-2.7	27.1
5	Trie($[\mathcal{S};\mathcal{C}]$)	✓	512	✓	✗	✓	0.0	4.6M	-1.6	38.8
6	Trie($\mathcal{C}$)	✓	512	✓	✗	✓	0.0	2.6M	-1.0	37.5
7	Trie($\mathcal{C}$)	✓	512	✓	✓	✓	0.0	0.5M	-1.2	36.9

tant traffic across more than 20 domains including Communications, SmartHome, and Music. The base RNN-T model was trained on the complete dataset which contains approximately 80k hours of audio. For training contextual adapters we split this training data into 'contextual' and 'non-contextual' partitions based on whether utterances contain content from the target domain, ensuring that 60% of utterances in each training batch are sampled from the non-contextual partition.

We report results on a 35k utterance *general* dataset and a 28k *named entity* dataset (NE) consisting of utterances from Communication domain, which contain an entity, which is also present in the associated biasing catalog. We report relative word error rate reduction (WERR) on the general dataset and relative slot word error rate reduction (WERR-S) for named entities on the named entity dataset. Higher values indicate better performance. The baseline is the RNN-T model without CB.

3.2. Model configuration

Input features are 64-dimensional log filter-bank energies extracted using a 25 ms analysis window and 10ms frame shift. Features are downsampled by a factor of three after stacking three consecutive frames, resulting in 192 feature coefficients per frame. Ground truth transcripts are tokenised using a word-piece tokeniser with vocabulary size of 4000 [21, 22]. The RNN-T encoder network consists of five LSTM layers, each with 1280 units, with a time-reduction layer (downsampling factor of two) 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 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. The total number of parameters in the model without CB is 129.2M. Decoding uses the standard RNN-T beam search [1] with a beam width of seven. Trie-based biasing adapters are trained with a maximum catalogue size of 2500. Unless stated otherwise, catalogue size is limited to 5000 during inference and the catalogue is guaranteed to contain the target entity. Contextual adapters are trained using the Adam optimizer and a warmup-hold-decay learning rate schedule with 3k step warm-up, 72k step hold, and 25k step decay. We use a maximum learning rate of 8×10^{-4} and a minimum learning rate of 6.25×10^{-5}. 16 GPUs are used to train the biasing adapters. In some cases, we observe over-fitting when training the the contextual adapters. In these cases, we apply target context dropout [10] to regularise the model, i.e., we remove the target entity from the catalogue during training with probability P_τ.

3.3. Attention-based biasing baseline

As an additional stronger baseline, we compare the proposed trie-based biasing approach to an attention-based contextual biasing approach, which has been proposed in [12]. In this approach, entity embeddings are computed using a bLSTM network. The biasing function then employs cross-attention to compute a weighted average of entity embeddings given internal model states, where the queries are hidden encoder states and the entity embeddings are are used as keys and values. We use the same hyper-parameters for the attention-based approach as [12]. The attention-based contextual adapter adds 608k parameters to the model ($< 0.5\%$ of the base RNN-T model parameters). As the attention-based biasing approach is more memory hungry compared to the trie-based approach, the maximum catalog size is set to 300 during training to fit within memory.

3.4. Experimental results

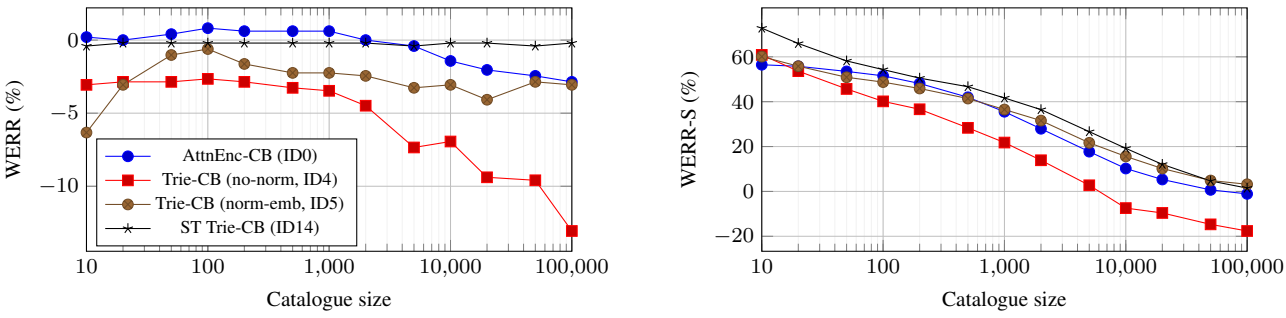
3.4.1. Adapter training and projection layer

Our experimental results are shown in Table 1. A configuration based on prior work [10] is represented as ID1. ID1 is shown to perform well on contextual entities but causes degradation on general content. We hypothesise that this is due to contextual utterances being over-sampled during training, as well as the use of embeddings relating to *starts* being included in the biasing, resulting in biasing being enabled at all times. We found that reducing the over-sampling rate significantly reduced improvements in slot error rate on contextual data, further motivating our pursuit of alternative methods of reducing over-biasing.

As a first step towards training the biasing module as an adapter, we froze the joint network weights and trained only the embedding matrices (ID2). While we still see reasonable improvements on WER-S, a large degradation is observed on general data. To mitigate this degradation we introduced a projection layer in ID3 and reduced the embedding dimensionality from 1024 to 512. This is found to be moderately effective, reducing the degradation on general data to 2.9%, however WER-S improvements remain relatively low compared to ID1. After introducing the projection layer we found that target context dropout was no longer required. With $P_\tau = 0$ (ID4) we now see even lower degradation on general data and WER-S on contextual entities is now lower than ID1.

3.4.2. Normalised embeddings

Next, we applied normalisation to the biasing vector as described in Section 2.1.3 (ID5). This was found to significantly improve WER-S on the named entity dataset, with WERR-S increasing from 27.1% to 38.8%, and degradation on general data also decreasing to less than half that observed with ID1. This appears to confirm the hypothesis that it is important to control the magnitude and dynamic range of the biasing vectors. To measure the relationship between the number of entries in the catalogue C and accuracy on the general and named entities dataset, we carried out an experiment where we gen-



(a) *General utterances*

(b) *Named entities*

Figure 2: *Impact of catalogue size on general and slot word error rates of CB methods. ST = slot-triggered selective biasing.*

Table 2: *Impact of reducing embedding dim. ($\mathcal{C}$ inputs only)*

ID	Embedding size	Shared embeddings	Trainable params	General WERR	NE WERR-S
7	512	✓	524k	-1.2	36.9
8	256	✗	1.3M	-1.0	36.9
9	64	✗	323k	-1.2	35.3
10	16	✗	81k	-1.2	29.3
11	8	✗	40k	-0.4	23.4
12	4	✗	20k	-0.4	16.9

erated synthetic catalogues with 10 to 100k domain-relevant terms per catalogue. Results are shown in Figure 2. Without normalisation, WER on general data begins to degrade further beyond 1000 entries per catalogue, with degradation accelerating beyond 2000 entries per catalogue. This corresponds to an increasing mismatch with training conditions where a cap of 2500 entries per catalogue was imposed. General WER is more stable with normalised catalogues (ID5) which suggests the model fails to generalise to larger catalogues due to the increasing magnitude of biasing vectors. In terms of WER-S on the named entity dataset, normalisation is found to be particularly beneficial for very large catalogues, with performance exceeding even the attention-based method when catalogue sizes increase beyond 1000 entities.

3.4.3. *Continuation-only biasing*

To further reduce degradation on general data we evaluated the impact of removing *starts* embeddings from the biasing vector (ID6). This was found to be beneficial, with degradation on general data reducing to 1% rel., with only a relatively small impact on WER-S. We found that while some entities can no longer be biased as they are represented as single word-pieces, these appear in the training data sufficiently often that the model is able to recognise them without CB. The minor degradation therefore is mostly comprised of cases where the first word-piece of less common entities is mis-recognised. Most of these confusions were phonetically related, e.g. *juan* → *one*.

3.4.4. *Reduced embedding footprint*

We investigated two methods to reduce the memory footprint of trie-based contextual adapter: i.) sharing embeddings with the prediction network input layer (ID7), and ii.) reducing embedding size (ID8-12). For ID7, the embedding matrix is frozen when training the biasing adapter. Accuracy was found to degrade slightly, however the number of additional parameters of the model is reduced five-fold over ID6. Table 2 shows the effect of reducing embedding size. Improvements on named entities remain relatively stable as embedding vectors are reduced from 512 dimensions to 64. Performance of the baseline trie-based model, ID1, is exceeded by a model with around 40k parameters (ID11): 200x fewer.

Table 3: *Impact of applying selective biasing to trie-based CB. All systems use shared embeddings.*

ID	Inputs	Selective biasing	Trainable Params	General WERR	NE WERR-S
7	$\mathcal{C}$	✗	524k	-1.2	36.9
13	$\mathcal{C}$	✓	524k	-0.2	37.2
14	$[\mathcal{S};\mathcal{C}]$	✓	1.0M	-0.4	40.5

3.4.5. *Selective biasing*

Slot-triggered biasing is found to be an effective method for reducing the degradation on general data to as little as 0.2% (ID13) while maintaining the improvements on the named entities dataset. Furthermore, we are able to re-introduce the *starts* embeddings as there is reduced risk of these degrading general data as found in Section 3.4.3 (ID14).

3.4.6. *Comparison to attention-based neural biasing*

Our final trie-based contextual biasing system (ID14) provides the best trade-off between accuracy improvements on contextual data and degradations on general data. Comparing this system with the attention-based neural biasing system, we see that both systems perform on par, however the trie-based method is computationally more efficient. In the setup with real contextual catalogues, the attention-based neural biasing slightly outperforms the proposed trie-based biasing approach (43.2% rel. WERR-S vs. 40.5% rel. WERR-S). In the experiments with synthetic catalogs, trie-based biasing slightly outperforms the attention-based method across all catalogue sizes, see Fig. 2. Noteworthy here is that trie-based biasing is invariant to catalogue size in terms of accuracy on general data. This is in contrast to attention-based neural biasing, which exhibits degradations for very large catalogue sizes.

4. Conclusions

We have demonstrated how over-biasing caused by trie-based CB can be practically eliminated through the use of adapter training and selective biasing, and how a simple normalisation term can improve robustness to large catalogues of biasing terms. We reduced memory footprint of the method by sharing existing model components with the biasing adapter. With the proposed improvements, the trie-based method was shown to achieve on-par performance with a computationally more expensive attention-based method of contextual biasing.

5. References

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

[2] 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.

[3] 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.

[4] S. Kim, Y. Shangguan, J. Mahadeokar, A. Bruguier, C. Fuegen, M. L. Seltzer, and D. Le, "Improved neural language model fusion for streaming recurrent neural network transducer," in *Proceedings of ICASSP*, 2021, pp. 7333–7337.

[5] A. Gourav, L. Liu, A. Gandhe, Y. Gu, G. Lan, X. Huang, S. Kalmane, G. Tiwari, D. Filimonov, A. Rastrow *et al.*, "Personalization strategies for end-to-end speech recognition systems," in *Proceedings of ICASSP*, 2021, pp. 7348–7352.

[6] X. Zheng, Y. Liu, D. Gunceler, and D. Willett, "Using Synthetic Audio to Improve The Recognition of Out-Of-Vocabulary Words in End-To-End ASR Systems," in *Proceedings of ICASSP*, 2021, pp. 5674–5678.

[7] Z. Chen, Y. Zhang, A. Rosenberg, B. Ramabhadran, P. Moreno, and G. Wang, "Tts4pretrain 2.0: Advancing the use of Text and Speech in ASR Pretraining with Consistency and Contrastive Losses," in *Proceedings of ICASSP*, 2022, pp. 7677–7681.

[8] G. Wang, A. Rosenberg, Z. Chen, Y. Zhang, B. Ramabhadran, Y. Wu, and P. Moreno, "Improving Speech Recognition Using Consistent Predictions on Synthesized Speech," in *Proceedings of ICASSP*, 2020, pp. 7029–7033.

[9] D. Baby, P. D'Alterio, and V. Mendelev, "Incremental learning for RNN-Transducer based speech recognition models," in *Proceedings of Interspeech*, 2022.

[10] 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, pp. 251–257.

[11] D. Le, M. Jain, G. Keren, S. Kim, Y. Shi, J. Mahadeokar, J. Chan, Y. Shangguan, C. Fuegen, O. Kalinli *et al.*, "Contextualized Streaming End-to-End Speech Recognition with Trie-Based Deep Biasing and Shallow Fusion," in *Proceedings of Interspeech*, 2021.

[12] 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.

[13] G. Sun, C. Zhang, and P. C. Woodland, "Tree-Constrained Pointer Generator for End-to-End Contextual Speech Recognition," in *Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding*, 2021, pp. 780–787.

[14] 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.

[15] 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.

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

[17] A. Alexandridis, K. Mysore Sathyendra, G. P. Strimel, F.-J. Chang, A. Rastrow, N. Susanj, and A. Mouchtaris, "Gated Contextual Adapters for Selective Contextual Biasing in Neural Transducers," in *Proceedings of ICASSP*, 2023.

[18] S.-A. Rebuffi, H. Bilen, and A. Vedaldi, "Learning multiple visual domains with residual adapters," *Advances in neural information processing systems*, vol. 30, 2017.

[19] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for Activation Functions," in *Proceedings of ICLR*, 2018.

[20] J. Andrés-Ferrer, D. Albesano, P. Zhan, and P. Vozila, "Contextual Density Ratio for Language Model Biasing of Sequence to Sequence ASR Systems," in *Proceedings of Interspeech*, 2021.

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

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