HIERARCHICAL ATTENTION-BASED CONTEXTUAL BIASING FOR PERSONALIZED SPEECH RECOGNITION USING NEURAL TRANSDUCERS

Sibo Tong, Philip Harding, Simon Wiesler

Amazon Alexa

ABSTRACT

Although end-to-end (E2E) automatic speech recognition (ASR) systems excel in general tasks, they frequently struggle with accurately recognizing personal rare words. Leveraging contextual information to bias the internal states of E2E ASR model has proven to be an effective solution. However most existing work focuses on biasing for a single domain and it is still challenging to expand such contextualization mechanisms to many domains. To address this limitation, in this work we propose a hierarchical attention architecture to scale contextual biasing to a wide range of domains simultaneously. Given multiple catalogs of contextual information, the high-level attention determines which source of catalog to focus on and the low-level attention learns to attend to the most relevant entity within the focused catalog. Experiments on diverse domains demonstrate the proposed architecture results in 35% to 60% relative WER improvements on personal rare words and outperforms existing approaches.

***Index Terms*—** RNN-T, neural-transducer, contextual biasing, personalization

1. INTRODUCTION

End-to-end (E2E) automatic speech recognition (ASR) models, including sequence-to-sequence neural transducer models such as Recurrent Neural Network Transducer (RNN-T) [1] and Conformer Transducer [2], have been widely adopted due to their ability to achieve state-of-the-art performance on a variety of ASR tasks. Even though these models offer architectural simplicity and efficient optimization, they often fail to recognize words that either did not occur, or rarely occurred, in the training data [3].

The recognition of uncommon words plays a crucial role in various ASR applications, particularly in virtual assistants. Shallow language model fusion is a common approach used in E2E models to bias ASR output towards contextual content [4]. This approach can be moderately effective, but it requires training of external language models and its decoding process is sensitive to a fusion coefficient. Moreover, shallow fusion does not account for acoustic information which provides a useful signal for determining when to bias. More recently, directly biasing the internal states of E2E ASR models has been explored [5, 6, 7, 8, 9, 10]. These methods, known as contextual biasing (CB), typically consist of two components: i.) a catalog encoder used to transform contextual entities into a format consumable by the model, and ii.) a mechanism to bias internal model states using the encoded contextual entities. CB is trained in conjunction with the main neural ASR architecture and usually outperforms shallow fusion [9, 10]. While CB techniques have shown promising results in various applications, it is important to acknowledge that they can potentially degrade the performance on common content [9, 10]. Efforts have been made to address this issue and reduce such degradation, as outlined in [11, 12, 13]. However, even when biasing is effectively controlled, scaling contextual biasing to multiple domains/slots in production-scale can pose significant challenges, due to increased catalogue sizes and inference latency [14].

More recently, there have been several attempts to address the issue of scaling contextual biasing to multiple domains via slot-triggered CB [15] or a model-internal slot classifier [16]. In this paper we aim for an improved approach using a hierarchical attention architecture to horizontally scale contextual biasing to multiple domains. The architecture proposed in this work is composed of two layers of attention. In the low-level attention, domain-specific biasing vectors are obtained by attending over biasing entities within each corresponding catalog and in the high-level attention, attention is performed over the domain representation to determine which contextual catalog to focus on. Additionally, the high-level attention can also be used to narrow down the catalogs used for biasing, resulting in reduced inference latency. The proposed approach is shown to achieve higher accuracy on personalised entities than existing approaches [10, 15, 16]. We also demonstrate that the proposed hierarchical attention is able to maintain high accuracy with low latency by running a top-K controlled domain selection during inference.

2. RELATED WORKS

To date, we are aware of relatively little work on efficiently scaling contextual biasing to multiple sources of contextual information. Slot-Triggered Biasing (STB) [15] was proposed to address this by augmenting the ASR transcript with slot opening and closing tags, and selectively activating the

979-8-3503-0689-7/23/$31.00

biasing module based on the predicted slot tags. 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 the model-internal slot classifier approach [16], where a streaming model-internal slot classifier is trained to categorise the domain of each word piece before it is emitted. The biasing module can therefore be triggered in a controlled way, permitting natural scaling to many domains while reducing false-biasing and computational cost. In this paper, we illustrate that the proposed hierarchical attention architecture encompasses this approach as a special case. Furthermore, we find that other variations built upon the hierarchical attention framework outperform the model-internal classifier approach.

Very recently, a similar two-pass attention concept has been investigated in [17] to improve inference latency for contextual biasing. In the first pass, the top-K candidate biasing phrases are selected based on a fast maximum inner product search, which compares biasing-phrase-level representations with the current audio feature. In the second pass, attention is computed over the selected phrases at the word-piece level. Focusing on a different aspect, we introduce the hierarchical attention architecture to scale the existing attention-based biasing approach to multiple catalogs of contextual information. We demonstrate through experiments that the proposed hierarchical architecture can efficiently bias the base model to multiple domains simultaneously with higher accuracy on slot content over existing approaches.

3. NEURAL TRANSDUCERS AND CONTEXTUAL ADAPTERS

3.1. Recurrent Neural Network Transducers

Recurrent Neural Network Transducers (RNN-T) are a type of streaming E2E ASR model [1], typically consisting of an encoder network, a prediction network and a joint network. The encoder network produces high-level representations $\mathbf{h}_t^{enc}$ for the audio frames $\mathbf{x}_{0,t} = (\mathbf{x}_0 ... \mathbf{x}_t)$. The prediction network encodes the previously predicted word-pieces $\mathbf{y}_{0,u-1} = (\mathbf{y}_0 ... \mathbf{y}_{u-1})$ and produces the output $\mathbf{h}_u^{pre}$. The encoder and the prediction network are typically stacked RNN layers [1] or conformer blocks [2, 18]. The joint network fuses $\mathbf{h}_t^{enc}$ and $\mathbf{h}_u^{pre}$ via the join operation and models the probability distribution over word-pieces (including the blank symbol):

$$P(\mathbf{y}_u|t,u) = \text{Join}(\mathbf{h}_t^{enc}, \mathbf{h}_u^{pre}) \,. \tag{1}$$

The entire model is trained with the RNN-T loss using the forward-backward algorithm [1].

3.2. Contextual Adapters

We adopt the contextual adapters proposed in [10] to perform contextual biasing. The contextual adapters comprise two components – a catalog encoder and a biasing adapter. Given a pre-trained ASR model, they adapt this model to perform contextual biasing based on contextual catalogs.

3.2.1. Catalog Encoder

This component encodes a catalog of contextual entities, $C = [c_1, c_2, ..., c_N]$ into model-interpretable representations. Each entity c_i, is first split into word-pieces using a sub-word tokenizer [19, 20], then passed through an embedding lookup followed by bidirectional long short-term memory (BiLSTM) layers. The output of the catalog encoder is defined as the final states of the BiLSTM layers, denoted as $\boldsymbol{\mathcal{E}} = [\mathbf{e}_1 ... \mathbf{e}_N]$, where

$$\mathbf{e}_i = \text{BiLSTM}(\text{Embedding}(\mathbf{c}_i)) \,. \tag{2}$$

3.2.2. Biasing Adapters

We use cross-attention [21] based biasing adapters (BA) to generate the biasing vector. The keys and values are projected from $\boldsymbol{\mathcal{E}}$. Unlike previous work [10], we only bias the encoder representation due to its simplicity without sacrificing performance. More specifically, we use the encoder representation $\mathbf{h}_t^{enc}$ as the query to generate the biasing vector $\mathbf{b}_t$, which is then used to update the encoder representations via element-wise additions (denoted by $\oplus$):

$$\mathbf{b}_t = \text{BA}(\mathbf{h}_t^{enc}, \boldsymbol{\mathcal{E}}); \hat{\mathbf{h}}_t^{enc} = \mathbf{h}_t^{enc} \oplus \mathbf{b}_t. \tag{3}$$

The biased encoder representation $\hat{\mathbf{h}}_t^{enc}$ is then combined with the prediction network output to produce the probability distribution over word-pieces.

4. HIERARCHICAL ATTENTION-BASED MULTI-SLOT CONTEXTUAL BIASING

We propose Hierarchical Contextual Adapters (HCA) which uses hierarchical attention to scale the contextual biasing to multiple catalogs, each of which contains contextual information for a specific slot or domain. In this approach, the biasing module is composed of two layers of attention. In the low-level attention, slot-specific biasing vectors $\mathbf{b}_t^s$ for each slot type s are obtained by attending over catalog embeddings within each catalog individually, using the current encoder states $\mathbf{h}_t^{enc}$ as queries. The low-level attention module is shared across all slot types. For readability we drop the time index t and label index u from future equations. If we consider the case of a single catalog, this step is equivalent to the contextual adapter approach [10]. Where this approach differs is the second step. Instead of directly biasing the encoder state using the biasing vector, a second attention

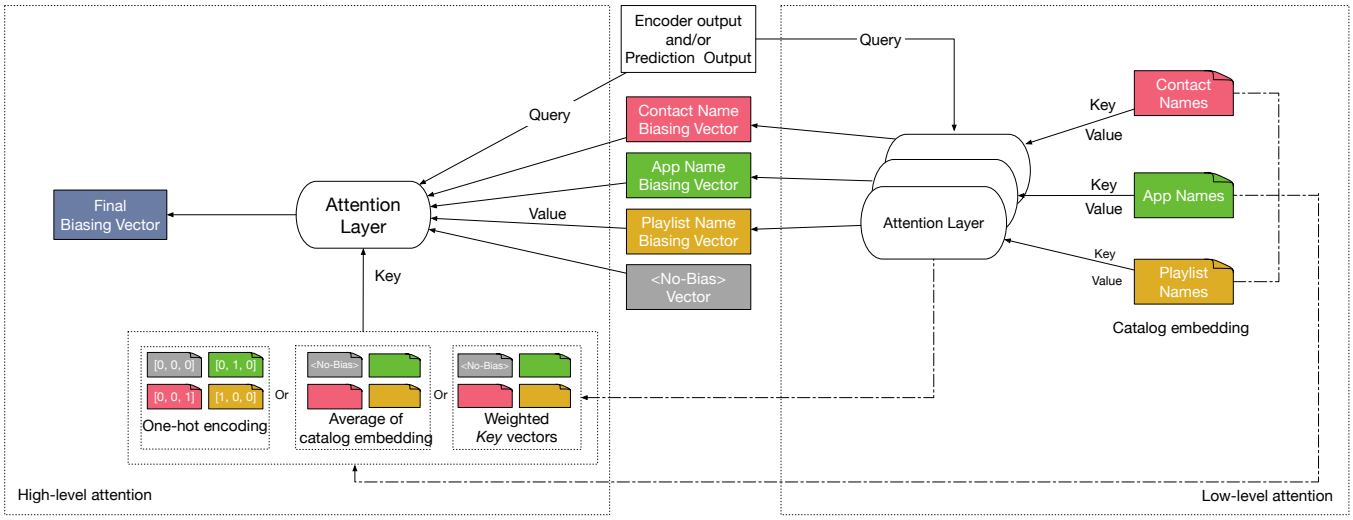

Fig. 1: An overview of the hierarchical contextual adapters.

block is used to attend over biasing vectors of all slot types. More specifically, the final biasing vector $\hat{\mathbf{b}}$ is calculated as the weighted sum of $\mathbf{b}^s$ over all S slot types:

$$\boldsymbol{\alpha} = \text{softmax}(\frac{\mathbf{Q} \cdot \mathbf{K}^\intercal}{\sqrt{d}}); \hat{\mathbf{b}} = \sum_{s=1}^{S} \alpha_s \mathbf{b}^s \tag{4}$$

where $\mathbf{b}^s$ is the biasing vector for slot type s generated from the low-level attention, $\mathbf{Q} = \mathbf{W}_q \text{Join}(\mathbf{h}^{enc}, \mathbf{h}^{pred})$ is the query, projected from joint of encoder and prediction network output with matrix $\mathbf{W}_q$. There can be multiple ways to construct the keys $\mathbf{K}$. Later in Section 4.1, we will discuss various methods considered in this paper to summarize the catalog information for each slot type. The resulting slot type embedding will be projected as keys $\mathbf{K}$. Figure 1 illustrates the overall architecture of the hierarchical attention approach. As shown in the figure, we add a special token to form the catalog of `<no-bias>` class in the high-level attention since biasing is not always needed. When attending to this class the value is an all-zero vector such that attending to this entry results in no change to the base model.

The high-level attention attends over slot-specific biasing vectors by measuring the distance between the RNN-T hidden states and the slot type embedding. It can be considered as a slot classifier which produces *slot probabilities* and applies weighted catalog combination using the probabilities as the weights.

4.1. Catalog Summarization Techniques

In the next subsections, we discuss the details of different approaches to generate slot type embeddings, which are used as keys $\mathbf{K}$ in the high-level attention.

4.1.1. Using one-hot slot type encoding (HCA-OH)

We use one-hot slot type encoding as the key for the high-level attention:

$$\mathbf{K} = \mathbf{W}_k \cdot \mathbf{E}_{one\_hot} \tag{5}$$

where $\mathbf{W}_k$ is the projection matrix for the keys and $\mathbf{E}_{one\_hot}$ is stacked one-hot slot type encoding, which is an identity matrix. Since $\mathbf{E}_{one\_hot}$ is fixed, calculating $\boldsymbol{\alpha}$ from Equation 4 is conceptually the same as using a single-layer slot classifier [16] without activation functions, which uses $\text{Join}(\mathbf{h}^{enc}, \mathbf{h}^{pred})$ as input and outputs slot type probabilities. The probabilities are then used as weights to combine slot-specific biasing vectors and generate the final biasing vector. We use the abbreviation OH (one-hot) to denote this approach in the rest of the paper.

4.1.2. Summarizing catalog information using weighted key vectors from low-level attention (HCA-WK)

Using one-hot encoding of slot types is straightforward. However, catalog selection in this case is performed based on only acoustic and semantic history. Since we consider only the streaming case in this work, catalog selection can be challenging when the context is limited and ambiguous, e.g., "play ..." where multiple slot types can be explored, or cases when the target entities appear in isolation without any context. One possible way to alleviate this issue is to utilize the catalog contents as additional information. For each slot type s, low-level attention generates attention weights $\boldsymbol{\alpha}^s_{low}$ and `key` vectors projected from catalog entity embeddings $\boldsymbol{\mathcal{E}}^s$ within the corresponding catalog. The attention weights represent the weights we want attend to each biasing entity. Using $\boldsymbol{\alpha}^s_{low}$ as weights, we calculate the weighted average of the `key` vectors, denoted as $\hat{\mathbf{k}}^s_{low}$, for each slot type s and use the stacked

weighted key vectors as $\mathbf{K}$:

$$\mathbf{K} = [\hat{\mathbf{k}}^1_{low}, ..., \hat{\mathbf{k}}^s_{low}, ..., \hat{\mathbf{k}}^S_{low}] \quad (6)$$

It summarises the content of each catalog and captures how well each catalog matches the current query. If a good match has been already detected in catalog specific attention, the weighted key vector of the corresponding catalog is also supposed to be closer to the query vector in the high-level attention. This approach will be denoted as WK (weighted key).

4.1.3. Summarizing catalog information using averaged catalog embeddings (HCA-ACE)

In the run-time, the WK approach requires the low-level attention to be performed over all catalogs first in order to generate the weighted key vectors for the high-level attention. As a result, it is not possible to directly apply catalog shortlisting on the high-level attention to reduce computational complexity, which might be critical for streaming applications. Another alternative to summarise catalogs is to use arithmetic mean of catalog embeddings. It also captures the catalog information and can be pre-computed offline. More specifically, we use

$$\mathbf{K} = \mathbf{W}_k \cdot [\overline{\boldsymbol{\mathcal{E}}^1}, ..., \overline{\boldsymbol{\mathcal{E}}^s}, ..., \overline{\boldsymbol{\mathcal{E}}^S}] \quad (7)$$

as the key in the high-level attention, where $\overline{\boldsymbol{\mathcal{E}}^s}$ is arithmetic mean of the catalog embeddings for slot type s. This mean vector can be calculated in advance of the high-level attention without going through individual catalog-specific attention in the low level, thus retaining the potential to optimize runtime latency by catalog shortlisting. We denote this approach as ACE (averaged catalog embedding) in the rest of the paper.

4.2. Catalog selection and two-pass hierarchical attention

During inference, to reduce the computational complexity of cross-attention, we could limit catalogs to only the most top-K probable slots at each decoding step based on the high-level attention weights. Subsequent low-level attention will only be performed on the selected catalogs. The biasing vector for the selected catalogs are then weighted by the corresponding high-level attention weights to enforce consistency between training and inference.

However, this selection cannot be applied to HCA-WK as the high-level attention requires low-level attention to be performed over all catalogs first in order to generate the weighted key vectors. To address this issue, we also explore a two-pass hierarchical attention. In the first pass, we use one-hot encoding as the high-level attention key and apply top-K selection. Low-level attention is then performed on each of the selected catalogs which outputs biasing vector $\mathbf{b}^s$ and the weighted average of the low-level key vectors $\hat{\mathbf{k}}^s_{low}$, same as described in Section 4.1.2. In the second pass, we follow the HCA-WK variant to refine the high-level attention weights over the selected catalogs using $\hat{\mathbf{k}}^s_{low}$ as the key. The top-K selection can be trained by simple back-propagation, along with the rest of the model [22]. The attention weights for the top K catalogs have nonzero derivatives with respect to the weights of the attention layer. We also ensure the ground truth catalog is always added to the shortlist after the top-K selection during training, which was shown to help the model converge faster in our preliminary experiments. Note that it is also possible to use ACE as the high-level key in the first pass but we leave it as future work.

4.3. Auxiliary slot classification loss

The high-level attention weights can be interpreted as the probabilities of picking each catalog for biasing. It is thus reasonable to apply an auxiliary loss on the attention weights to guide the model training. We use the slot annotation obtained from in-house entity tagger as the ground truth and apply the same cross-entropy (CE) loss investigated in [16, 23] as the auxiliary loss function, weighted by a tunable parameter τ. More specifically, given the transcript word-pieces of length U, $(\mathbf{y}_1, ..., \mathbf{y}_U)$ and its corresponding slot tags (also of length U), $\mathbf{y}^s = (y^s_1, ..., y^s_U)$, the CE loss is defined as:

$$\mathcal{L}_{ce} = \frac{1}{U}\sum_{u=1}^{U}\frac{1}{T}\sum_{t=1}^{T} y^s_u \log \alpha_{t,u} \quad (8)$$

where $\alpha_{t,u}$ is the high-level attention weight at state (t, u) queried by Join$(\mathbf{h}^{enc}_t, \mathbf{h}^{pre}_u)$. The CE loss is weighted by τ before being combined with the RNN-T loss to generate the final training loss.

5. EXPERIMENTAL ANALYSIS

5.1. Experimental Setup

5.1.1. Dataset and evaluation metric

We used an in-house de-identified American English voice assistant dataset with each utterance consisting of audio, transcription and catalogs of contextual entities. The training data is not associated with identifying information, but some utterances may contain named entities. Utterances were randomly sampled from the voice assistant traffic across more than 20 domains including Communications (Comms), SmartHome, and Music. In this work we focused on applying contextual biasing to three domains - Comms, Music and Applications, where proper names (e.g. playlist names) or app launch phrases are used for biasing. The base RNN-T models were trained on the complete dataset which contains approximately 86k hours of audio. For training contextual adapters we split this training data into 'personalized' and 'non-personalized' 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 personalized data for Comms, 100 hours for Music, and 500 hours for application launch phrases. The non-personalized human-transcribed dataset contained approximately 66k hours. Slot annotations were obtained using an in-house entity tagger.

For training the Hierarchical Contextual Adapters (HCA), we used a mix of personalized and non-personalized data, ensuring that 40% utterances in each training batch are sampled from the non-personalized partition and the rest 60% were equally sampled from the personalized partition from each of the 3 target domains. We evaluate our models and report results on a 34k utterance general dataset, and three personalized datasets - a 34k utterance Comms dataset, a 4.4k utterance Music dataset and a 22k utterance application launch dataset. Utterances from personalized test sets always contain target content in the associated catalogs. Utterances from the general test set span over 20+ domains, tracking general model performance and highlighting overbiasing towards any specific domains. Same as [9, 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 target biasing content on the personalized datasets. In all cases in this paper the baseline model is the RNN-T ASR model before applying contextual biasing. Higher values indicate better performance.

5.1.2. 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 3 is applied, resulting in 192 feature coefficients per frame. Ground truth tokens are tokenized using a word-piece tokenizer with vocabulary size of 4000 [19, 20]. 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 through a feed-forward layer to 1024 units. We use a simple addition ($\oplus$) 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 7.

For the hierarchical contextual biasing, the catalog encoder and the attention-based adapter used in the low level have the same hyper-parameters as used in [10]. The dimensions of query and key in high-level attention are set to 64. The contextual adapter is trained using the Adam optimizer 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 1.6×10^{-3} and a minimum learning rate of 1.25×10^{-4}. The maximum catalog size for each domain is set to 300 during training and 5000 during inference to fit within memory.

Table 1: *Tuning weight for the auxiliary CE loss, reported in relative change in WER (%) for general data and WER-S (%) for target content on personalized data, over unbiased RNN-T.*

System	CE weight τ	General WERR	App WERR-S	Comms WERR-S	Music WERR-S
HCA-OH	0	-1.2	27.4	43.8	57.2
	0.1	-1.2	28.6	44.2	57.2
	0.5	-1.0	29.8	**44.5**	**58.9**
	1.0	**-0.8**	29.4	42.6	57.8
	2.0	-1.0	**30.7**	42.6	55.6

5.2. Experimental Results

5.2.1. Tuning weight of the auxiliary loss

We began by experimenting with hierarchical attention using one-hot slot type encoding as the key in the high-level attention (HCA-OH), and exploring various weights τ for the CE auxiliary loss. As illustrated in Table 1, HCA-OH performs reasonably well on personalized test sets without auxiliary loss. Adding auxiliary loss on high-level attention weights helps the model learn which catalog to bias towards as well as when not to bias, leading to further improvement in performance across the board. A weight of 0.5 is observed to achieve a good balance between general and personalized data. We also tuned the auxiliary loss weight for the other variants in our preliminary experiments but 0.5 is found to give the best over all performance. Therefore, we have decided to adopt a weight of 0.5 for the auxiliary loss in all subsequent experiments.

5.2.2. Comparison between different variants

We then explore the two approaches that have access to the catalog content in the high-level attention, namely WK (weighted key) and ACE (averaged catalog embedding). We apply the same auxiliary slot classification loss with weight of 0.5 given previous observations. The results are summarized in second block of Table 3. Compared with HCA-OH, both HCA-WK and HCA-ACE achieve better performance on most of the personalized test sets. This indicates that catalog information is helpful for catalog selection performed at the high-level attention. HCA-WK is particularly better than HCA-OH on Music and App Launch data where the slot classification can be ambiguous based on limited acoustic and semantic histories. Compared with HCA-WK, HCA-ACE shows slightly worse performance on personalized test sets. This can be attributed fact that HCA-WK, in the high-level attention, incorporates attention weights derived from low-level attention when generating key vectors, making it more informative than the arithmetic mean of catalog embedding.

Table 2: *Results of top-K catalog selection during inference, reported in relative change in WER (%) for general data and WER-S (%) for target content on personalized data, over unbiased RNN-T.*

System	Catalog Selection	General Data WERR	App Launch Data WERR-S	Comms Data WERR-S	Music Data WERR-S
HCA-OH	soft	-1.0	29.8	44.5	58.9
	top-1	**-0.4**	21.4	39.6	54.5
	top-2	-1.4	29.0	44.1	57.2
HCA-ACE	soft	-1.2	31.0	44.3	**60.9**
	top-1	-1.0	21.4	38.2	55.0
	top-2	-1.0	29.4	**44.6**	58.5
two pass	top-2	-1.0	**35.1**	44.4	60.0

Table 3: *Comparison between different HCA variants and previous work, reported in relative change in WER (%) for general data and WER-S (%) for target content on personalized data, over unbiased RNN-T.*

System	General WERR	App WERR-S	Comms WERR-S	Music WERR-S
Baseline	0	0	0	0
App Launch CB	**0.6**	20.1	-	-
Comms CB	-2.0	-	43.1	-
Music CB	0.2	-	-	39.5
CB-Merged	-1.6	22.5	37.1	49.1
slot-triggered CB	-1.2	33.3	**45.1**	49.6
model-internal classifier	-1.0	27.4	43.0	53.2
HCA-OH	-1.0	29.8	44.5	58.9
HCA-WK	-0.6	**33.5**	44.5	**62.3**
HCA-ACE	-1.2	31.0	44.3	60.9

5.2.3. Comparison with existing approaches

We now compare our proposed framework with existing approaches. CB-Merged is basic contextual adapters scaled to multiple slot types by merging multiple catalogs into a single one and using a 'type embedding' to distinguish between different catalog types following [10]. In slot-triggered CB, the base RNN-T model is trained to emit slot opening and closing tags corresponding to the target domains. The contextual biasing module is then trained to selectively bias the model internal states based on the predicted slot tags following [15]. As for the model-internal classifier approach, we follow [16] and train a feedforward network to categorise the domain of each word piece for biasing. The same CE loss for domain classification is used as the auxiliary loss function during training. Soft slot combination is used in inference, which combines slot-specific biasing vectors using slot probabilities as the weights. We adopt the same configurations for the catalog encoder and biasing adapters in all these approaches. From the comparison in Table 3, the proposed method is shown to scale well to multiple domains; all the three variants outperform models with single-domain biasing by a large margin. HCA-OH yields similar or better performance than the model-internal classifier approach as expected since they are conceptually equivalent as described in 4.1.1. With HCA-WK, we achieve a 45% WERR-S over the baseline on Comms, 62% on Music and 34% on App launch phrases with negligible degradation on general data, outperforming all existing approaches.

5.2.4. Top-K catalog selection

Finally, we evaluate the hierarchical attention models with the different slot selection methods. Results are reported in Table 2. Note `soft` represents the standard attention mechanism without any selection as described in Equation 4. Similar to the results reported in [16], `top-1` catalog selection is found to reduce improvements across all domains. With `top-2` selection, we are able to maintain most of the improvement with an average of only 1.01 catalogs processed per emission step. This reduction in the number of catalogs to be processed contributes to a decrease in runtime latency. We also explore the two-pass hierarchical attention as described in Section 4.2. We select the `top-2` catalogs in the first pass with one-hot encoding as the high-level attention key. In the second pass, we refine the high-level attention weights over the selected catalogs using catalog information. This two-pass approach is shown to yield the best performance. Compared with the results in Table 3, the two-pass approach achieves high accuracy and low latency by running a `top-2` controlled domain selection during inference.

6. CONCLUSION

In this paper, we proposed the hierarchical contextual adapter to scale contextual biasing to biasing personalized content across a variety of domains using multiple catalogs. We compared different choices of the key in the high-level attention and we demonstrated through experiments that catalog information is helpful for catalog selection performed at the high level. The proposed hierarchical architecture is shown to largely outperform existing approaches. In our future work we will explore other combinations of different keys and investigate gating based on the weight of `<no-bias>` class in the high-level attention to further reduce inference latency.

7. REFERENCES

[1] Alex Graves, "Sequence transduction with recurrent neural networks," in *Proceedings of the 29th International Conference on Machine Learning (ICML)*, 2012.

[2] Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, et al., "Conformer: convolution-augmented transformer for speech recognition," in *Proceedings of Interspeech*, 2020.

[3] Cal Peyser, Sepand Mavandadi, Tara N Sainath, James Apfel, Ruoming Pang, and Shankar Kumar, "Improving tail performance of a deliberation E2E ASR model using a large text corpus," in *Proceedings of Interspeech*, 2020.

[4] Ding Zhao, Tara N Sainath, David Rybach, Pat Rondon, Deepti Bhatia, Bo Li, and Ruoming Pang, "Shallow-fusion end-to-end contextual biasing.," in *Proceedings of Interspeech*, 2019.

[5] Duc Le, Gil Keren, Julian Chan, Jay Mahadeokar, Christian Fuegen, and Michael L Seltzer, "Deep shallow fusion for RNN-T personalization," in *Proceedings of the IEEE Workshop on Spoken Language Technology*, 2021.

[6] Mahaveer Jain, Gil Keren, Jay Mahadeokar, Geoffrey Zweig, Florian Metze, and Yatharth Saraf, "Contextual RNN-T for open domain ASR," in *Proceedings of Interspeech*, 2020.

[7] Guangzhi Sun, Chao Zhang, and Philip 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.

[8] Golan Pundak, Tara N Sainath, Rohit Prabhavalkar, Anjuli Kannan, and Ding Zhao, "Deep context: end-to-end contextual speech recognition," in *Proceedings of the IEEE Workshop on Spoken Language Technology*, 2018.

[9] Feng-Ju Chang, Jing Liu, Martin Radfar, Athanasios Mouchtaris, Maurizio Omologo, Ariya Rastrow, and Siegfried Kunzmann, "Context-aware transformer transducer for speech recognition," in *Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding*. IEEE, 2021.

[10] Kanthashree Mysore Sathyendra, Thejaswi Muniyappa, Feng-Ju Chang, Jing Liu, Jinru Su, Grant P Strimel, Athanasios Mouchtaris, and Siegfried Kunzmann, "Contextual adapters for personalized speech recognition in neural transducers," in *Proceedings ICASSP*, 2022.

[11] Tianyi Xu, Zhanheng Yang, Kaixun Huang, Pengcheng Guo, Ao Zhang, Biao Li, Changru Chen, Chao Li, and Lei Xie, "Adaptive contextual biasing for transducer based streaming speech recognition," *arXiv preprint arXiv:2306.00804*, 2023.

[12] Anastasios Alexandridis, Kanthashree Mysore Sathyendra, Grant P Strimel, Feng-Ju Chang, Ariya Rastrow, Nathan Susanj, and Athanasios Mouchtaris, "Gated contextual adapters for selective contextual biasing in neural transducers," in *Proceedings ICASSP*, 2023.

[13] Philip Harding, Sibo Tong, and Simon Wiesler, "Selective biasing with trie-based contextual adapters for personalised speech recognition using neural transducers," in *Proceedings of Interspeech*, 2023.

[14] Tara N Sainath, Rohit Prabhavalkar, Diamantino Caseiro, Pat Rondon, and Cyril Allauzen, "Improving contextual biasing with text injection," in *Proceedings ICASSP*, 2023.

[15] Sibo Tong, Philip Harding, and Simon Wiesler, "Slot-triggered Contextual Biasing for Personalized Speech Recognition using Neural Transducers," in *Proceedings ICASSP*, 2023.

[16] Edie Lu, Philip Harding, Kanthashree Mysore Sathyendra, Sibo Tong, Xuandi Fu, Jing Liu, Feng-Ju Claire Chang, Simon Wiesler, and Grant Strimel, "Model-internal slot-triggered biasing for domain expansion in neural transducer asr models," in *Proceedings of Interspeech*, 2023.

[17] Tsendsuren Munkhdalai, Zelin Wu, Golan Pundak, Khe Chai Sim, Jiayang Li, Pat Rondon, and Tara N Sainath, "NAM+: Towards scalable end-to-end contextual biasing for adaptive ASR," in *Proceedings of the IEEE Workshop on Spoken Language Technology*, 2023.

[18] Bo Li, Anmol Gulati, Jiahui Yu, Tara N Sainath, Chung-Cheng Chiu, Arun Narayanan, Shuo-Yiin Chang, Ruoming Pang, Yanzhang He, James Qin, et al., "A better and faster end-to-end model for streaming ASR," in *Proceedings ICASSP*, 2021.

[19] Rico Sennrich, Barry Haddow, and Alexandra Birch, "Neural machine translation of rare words with subword units," in *Proceedings ACL*, 2015.

[20] Taku Kudo, "Subword regularization: Improving neural network translation models with multiple subword candidates," in *Proceedings ACL*, 2018.

[21] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, 2017.

[22] Yoshua Bengio, Nicholas Léonard, and Aaron Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," *arXiv preprint arXiv:1308.3432*, 2013.

[23] Xuandi Fu, Feng-Ju Chang, Martin Radfar, Kai Wei, Jing Liu, Grant P Strimel, and Kanthashree Mysore Sathyendra, "Multi-task RNN-T with semantic decoder for streamable spoken language understanding," in *Proceedings ICASSP*, 2022.