ON-THE-FLY TEXT RETRIEVAL FOR END-TO-END ASR ADAPTATION

Bolaji Yusuf[1,2,3*], *Aditya Gourav*[1], *Ankur Gandhe*[1], *Ivan Bulyko*[1]

[1] Amazon Alexa, USA
[2] Boğaziçi University, Department of Electrical and Electronics Engineering, Istanbul, Turkey
[3] Brno University of Technology, Faculty of Information Technology, Speech@FIT, Czechia

ABSTRACT

End-to-end speech recognition models are improved by incorporating external text sources, typically by fusion with an external language model. Such language models have to be retrained whenever the corpus of interest changes. Furthermore, since they store the entire corpus in their parameters, rare words can be challenging to recall. In this work, we propose augmenting a transducer-based ASR model with a retrieval language model, which directly retrieves from an external text corpus plausible completions for a partial ASR hypothesis. These completions are then integrated into subsequent predictions by an adapter, which is trained *once*, so that the corpus of interest can be switched without incurring the computational overhead of retraining. Our experiments show that the proposed model significantly improves the performance of a transducer baseline on a pair of question-answering datasets. Further, it outperforms shallow fusion on recognition of named entities by about 7% relative; when the two are combined, the relative improvement increases to 13%.

***Index Terms*—** retrieval, language model, domain adaptation, end-to-end ASR, RNN transducer, contextual biasing

1. INTRODUCTION

End-to-end (E2E) speech recognition models can be improved on a domain when they are shown text from that domain. While there have been works that do so by training parts of the model on external text [1–3], the most common method of incorporating text, unless precluded by computational constraints, is still fusion with language models (LM) [4–8] since they can be swapped at inference time.

Nevertheless, even LMs, especially neural LMs, can be unwieldy to change to match user interest. It is common for users of voice assistants and other speech technologies to use words and phrases associated with trending topics such as sporting events, album releases, pandemics etc. To contend with these surges in user interest, these ASR systems must be able to rapidly assimilate words of interest and also gracefully discard them as such ephemeral interest wanes. Although it is possible to obtain relevant text from internet forums or news articles, incorporating them into the ASR LM requires retraining the entire LM or training a separate LM for each trending topic. Moreover, LMs struggle with proper nouns and other named entities which are of the most interest because such entities, by nature rare and diffuse in training data, are assigned low likelihoods by LMs which store all information in their parameters.

This has sparked interest in biasing methods which attempt to boost the likelihoods of a catalog of entities.These include finite state transducer (FST)-based methods which compose the ASR output with an FST with negative-cost arcs carrying the entities to be boosted [9–12], and deep-biasing methods which introduce a trainable adapter into the ASR model, with an attention mechanism to select the right entity to boost [12–16]. However, both are more suited to catalogs of limited size (up to a few hundred at a time), such as contact names and song playlists, rather than the large catalogs necessary to cover multiple trending topics at the same time. FSTs for instance boost all items in the catalog with predefined weights making it hard to control what gets boosted as the catalog size increases. For deep-biasing, the limitation is due to the smearing of attention weights as the catalog size increases, as well as the increasing computational overhead of multihead attention. Therefore, it remains a challenge to have a rapidly adaptable, computationally efficient way to bias ASR towards large lists of phrases–possibly millions of tokens–at a time.

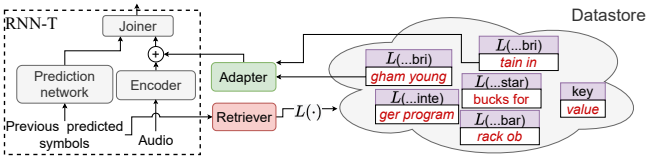

Fig. 1: RNN-T modified to use retrieval from a datastore. $L(\ldots xyz)$ denotes the retrieval LM encoding of some sentence ending in *xyz*.

Inspired by the success of retrieval mechanisms in language modeling [17–19], we propose augmenting an RNN transducer (RNN-T)-based ASR model with a retriever which searches in an external datastore for candidate continuations of a partial ASR hypothesis. The RNN-T's encoder output then attends to encodings of the retrieved continuations, and the attention output is summed to the encoder output before begin fed into the joiner.

Experiments on the Squad [20] and DeepMind Question-Answering [21] datasets show that a strong RNN-T baseline can be improved by retrieving from datastores that contains related text, even when the datastores also have millions of tokens of unrelated, distracting content. Furthermore, retrieval can be complemented by shallow fusion as the latter performs better on recognition of common words while retrieval performs better for named entities.

2. MODEL

2.1. RNN Transducer

The model we propose is built on the RNN transducer [22]. The transducer is an end-to-end ASR model composed of three parts.

The encoder is a recurrent neural network which encodes a sequence of audio frames $(\mathbf{x}_1, \ldots, \mathbf{x}_N)$ into hidden states $(\mathbf{h}_1, \ldots, \mathbf{h}_N)$:

$$\mathbf{h}_n = f^{enc}(\mathbf{x}_n, \mathbf{h}_{n-1}), \quad (1)$$

*Work done as an Applied Scientist intern at Amazon Alexa.

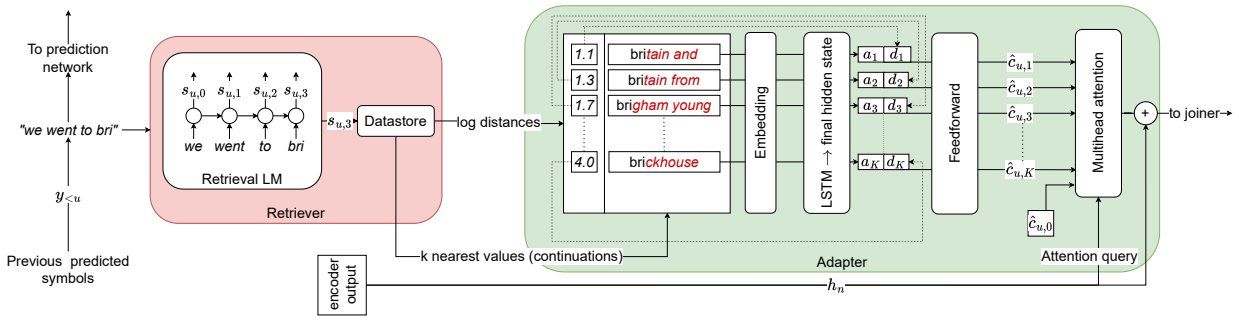

Fig. 2: The retrieval LM computes an embedding for previously predicted symbols and retrieves the k nearest neighbors of this embedding are from a datastore. The continuations (italicized in red font) and log Euclidean distances of the neighbors are transformed by an encoding network and the resulting vectors are blended by an attention mechanism whose output is used to modify the original RNN-T encoder output.

The prediction network is a recurrent network which encodes the previously output non-blank tokens $y_{<u}$:

$$\mathbf{g}_u = f^{pred}(y_{u-1}, \mathbf{g}_{u-1}), \tag{2}$$

The joiner computes a joint embedding from the two outputs:

$$\mathbf{z}_{n,u} = \phi(\mathbf{U}\mathbf{h}_n + \mathbf{V}\mathbf{g}_u + \mathbf{b_1}). \tag{3}$$

$\mathbf{U}$, $\mathbf{V}$ and $\mathbf{b}_1$ are trainable parameters and ϕ is the hyperbolic tangent function. The joint embedding is then used to compute a probability distribution over all tokens plus the blank token for alignment:

$$p(y|n,u) = \sigma(\mathbf{W}\mathbf{z}_{n,u} + \mathbf{b}_2). \tag{4}$$

$\mathbf{W}$ and $\mathbf{b}_2$ are trainable parameters and σ is the softmax function.

2.2. Retrieval-augmented RNN-T

Figure 1 depicts the modified RNN-T structure that we propose and Figure 2 illustrates the modifications in more detail. Our modifications comprise a retriever which finds potential continuations for the RNN-T's current output from external text and an adapter which uses those continuations to bias the RNN-T's subsequent outputs.

2.2.1. Retriever

The retriever, depicted in pink in Figure 2, is based on the premise that embeddings generated by a pretrained neural language model for two similar phrases are closer in Euclidean space than those of two random phrases. Therefore, to find potential continuations for any phrase (partial ASR hypotheses in our case) in a text corpus, we need to find phrases in that corpus that have similar embeddings to our phrase of interest, and return their continuations.

At the heart of our retriever is a pretrained LSTM LM which is used to generate embeddings for retrieval.[1] First we create a datastore for an adaptation text from which we intend to retrieve. To do this, each sentence in the corpus is passed through the LM; the LSTM's hidden state at each step is added as a key to the datastore, with corresponding value comprising the input tokens to the next t steps (we set $t = 2$ in this paper). By repeating this procedure for all sentences in the text, we get a key-value store, whose keys are the LM embeddings of phrases in the text, and whose values are the t-token long continuations of each key phrase.

To obtain candidate continuations of a partial ASR hypothesis during RNN-T decoding, we encode it with the retrieval LM and find the k nearest neighbors by Euclidean distance from the datastore. Note that the query to the retrieval is the same as the input to the prediction network, i.e., the sequence of non-blank tokens ($y_{<u}$). The retrieved continuations are then passed to the adapter along with the logarithms of the Euclidean distances between each key and the querying embedding.

2.2.2. Adapter

Having retrieved the k nearest neighbors, the question remains how best to integrate them back into the ASR. Adopting the framework used for contextual biasing in [15], we introduce a trainable adapter to bias the RNN-T's encoder output before feeding it into the joint network. The adapter, depicted in green in Figure 2, comprises a recurrent encoder and a multihead dot-product attention mechanism.

The adapter encoder has an embedding layer which converts the tokens in each candidate continuation into dense form. This is followed by an LSTM whose hidden state at the last step is taken as a fixed-length representation of that candidate. The corresponding log-distance from the retriever is concatenated to give the model a clue about the relevance of each candidate, and this vector is further transformed by a feedforward network to get a final representation.

The multihead attention is used to select from the representations of the candidates. The attention query is an affine projection of the encoder output $\mathbf{h}_n$[2] and its keys and values are projections of the candidates' encodings. Finally, the resulting context vector is added to the transcription output before passing the sum to the joiner.

In effect, the adapter modifies Equation 3 to:

$$\mathbf{z}_{n,u} = \phi\Big(\mathbf{U}\big(\mathbf{h}_n + \sum_{k=0}^{K} \alpha_{n,u,k}\mathbf{c}_{u,k}\big) + \mathbf{V}\mathbf{g}_u + \mathbf{b}_1\Big), \tag{5}$$

where K is the number of retrieved candidates and is a hyperparameter, $\mathbf{c}_{u,1}, \ldots, \mathbf{c}_{u,K}$ are the values of the attention mechanism computed from the retrieved candidates (by searching $y_{<u}$ in the datastore), $\mathbf{c}_{u,0} = \mathbf{c}$ is an extra trainable "no-bias" embedding intended to give the model an option when all retrieved candidates are incorrect, and $\alpha_{n,u,k}$ is the attention weight between $\mathbf{h}_n$ and $\mathbf{c}_{u,k}$.

3. EXPERIMENTS

3.1. Datasets

We experiment with two question-answering (QA) datasets: the Stanford Question Answering Dataset (Squad) v2.0 [20] and the

[1] We use an LSTM instead of the transformers used in prior works on retrieval LM due to practical latency and memory considerations. Note that the retrieval LM needs not be trained on the text from which we retrieve. In fact, in all our experiments, we use the same pretrained LSTM LM regardless of the adaptation text.

[2] We also tried using $\mathbf{g}_u$ as the attention query. We found that while using $\mathbf{g}_u$ is computationally cheaper, using $\mathbf{h}_n$ results in better ASR performance.

Table 1: Summary of the test sets. Squad-V and Squad-T refer to the Squad validation and test sets respectively, and DQA-V and DQA-T refer to the DQA validation and test sets respectively.

Dataset	Contexts		Questions	
	#Paragraphs	#Tokens	#Sentences	#Words
Squad-V	19124	3997k	6510	68107
Squad-T	1204	287k	11868	123020
DQA-V	1220	1371k	3924	53494
DQA-T	1093	1172k	3198	44737

CNN portion of the DeepMind Question Answering (DQA) [21] dataset. Each has a set of questions and a set of "context" paragraphs which contain information useful for answering the questions. [3]

Our task is to perform ASR on the questions. We use the TTS system from [23] to obtain speech for the questions and construct datastores for retrieval from the contexts. This setup emulates the data available to a typical voice assistant with open-source, non-proprietary data. The synthesized questions correspond to user queries, and the contexts correspond to the knowledge base with which a downstream NLP module would resolve the queries. Note that since we do not know exactly which context applies to which question, each dataset's datastore contains the keys and continuations of all contexts from the entire dataset.

In Table 1, we report the number of context paragraphs, questions and their constituent tokens and words for each dataset. Note that for Squad, since the official test set is not public, we use the official dev-set as our test set, and split off 5% of the training questions for validation. We use the entire set of Squad training contexts for the Squad-V datastore, which makes the datastore large and retrieval that much harder. For DQA, all entities are de-anonymized before TTS and datastore construction.

To get a strong DQA baseline, we pretrain the baseline RNN-T for 100k steps on a mix of the DQA and Librispeech [24] training sets, with batches uniformly sampled from each set. The DQA training set contains 380k questions which–to avoid any acoustic mismatch–we synthesize with the same TTS system as the test sets, using speaker profiles which include those used for the test sets. Compared to a baseline trained purely on Librispeech, including the DQA training set improves the WER from 34.3% to 13.8% on the DQA-T, from 23.5% to 15.2% on the Squad-T, with only a negligible degradation on the Librispeech test sets (6.5% to 6.6% on the test-clean).

With the parameters of the pretrained RNN-T frozen, we train the adapter for 30k steps on the Squad training set, which contains 124k sentences (the official training set minus the validation sentences); we use its synthesized questions as ASR data along with a datastore of all its contexts (same datastore as Squad-V).

To mitigate the risk of adapter overfitting on retrieved continuations, we add two forms of noise and force the model to learn when to rely on the no-bias embedding. First, we mix in an equal proportion of Librispeech batches to the *adapter* training data so that the retrieved continuations from the Squad datastore would be irrelevant. In addition, with 0.1 probability we retrieve random continuations from the training datastore instead of the actual k nearest neighbors, so that the model is exposed to–and learns to deal with–incorrect continuations even when the domain matches.

[3] see https://rajpurkar.github.io/SQuAD-explorer/explore/v2.0/dev for examples of Squad contexts and associated questions.

3.2. Model configuration

The baseline RNN-T has 64 million parameters, comprising an encoder with six LSTM layers with 1024 units followed by a 640-dimensional affine projection, a prediction network which has two LSTM layers of 1024 units also followed by projection to 640 dimensions, and a joiner with intermediate dimension of 512 and output dimension of 2501 corresponding to 2500 unigram subword tokens [25] trained on Librispeech, and the extra blank token.

The adapter adds 1 million parameters, of which $2501 \times 128 \approx 320k$ are in the embedding layer. The remaining parameters are in two 128-unit LSTM layers, two feedforward layers with 128 units and multihead attention with two heads and key dimension of 128.

The retrieval LM is a two-layer LSTM with 256 units trained on Wikitext-103 [26]. We reiterate that this retrieval LM is kept fixed regardless of which datastore we retrieve from.

We implement the k-nearest-neighbor search in FAISS [27] using a CPU index with product quantization [28]. The largest index in our experiments–constructed by concatenating all datastores ("All" in Section 3.3)–occupies just under 500 megabytes of RAM. We set K to 16 for both training and inference, i.e. at each step, we retrieve 16 candidate continuations out of hundreds of thousands to millions (number context tokens in Table 1).

3.3. Test set performance

Table 2 shows the word error rates (WER) of the various test sets.

Since our retrieval mechanism involves introducing and training an adapter, some of the improvement or degradation in performance may be attributed to simply having extra parameters trained on question-answering data, essentially updating the RNN-T's internal language model (ILM), rather than being able to retrieve and use the correct continuations. To measure this effect, we train another baseline which has an adapter but no retriever. For this, we input to the adapter LSTM a single trainable embedding instead of the embeddings of retrieved continuations. This "fixed embedding" approach, when compared to the baseline without any adaptations, improves Squad and degrades DQA performance. This is expected because the baseline training includes DQA training data, while the adapter is trained with Squad and Librispeech ASR data.

We observe significant improvements compared to either baseline on all test sets when the datastore *matches* the corresponding contexts, e.g. Squad-T datastore for Squad-T test set etc. It is noteworthy that we get relative improvements of 9% and 7% respectively on the DQA validation and test sets compared to the baseline with no adapter despite the performance drop that we incur due to the ILM shift from training the adapter on Squad+Librispeech (as evidenced by the fixed embedding results).

Next, we observe that the performance improvements brought by retrieval are generally proportional to how relevant the datastore is. For instance, when decoding the Squad test set, the WER increases when we switch from the datastore of Squad test contexts to the datastore of Squad validation contexts and increases further as we switch to the datastores of DQA contexts, at which point we get performance comparable to using the fixed embedding.

The matched results are predicated on picking the right datastore for each test utterance. We also consider retrieving from a single large datastore containing contexts from all the datasets (the "All" rows in the table). While this performs worse than picking the matching datastore, it is significantly better than using any other single datastore. This implies that even in the presence of a few million extra distracting contexts in the datastore, the retriever still does a good job of retrieving the correct ones. Thus, in practice, it would

Table 2: WER on various test sets as the datastore is varied compared to the baseline with no retrieval ("None") and a baseline with retrieval replaced by a fixed embedding ("Fixed emb."). S-F denotes the use of shallow fusion.

Datastore	S-F	Squad-V	Squad-T	DQA-V	DQA-T
None	✗	15.8	15.2	14.0	13.8
Fixed emb.	✗	14.4	13.8	14.7	14.7
Squad-V	✗	**11.9**	12.5	15.1	15.1
Squad-T	✗	13.6	**11.1**	15.5	15.7
DQA-V	✗	14.2	13.6	**12.7**	14.3
DQA-T	✗	14.4	13.8	14.6	**12.8**
All	✗	12.3	11.9	13.3	13.3
None	✓	13.7	13.1	12.5	12.4
Fixed emb.	✓	12.3	11.7	13.4	13.3
Match	✓	**11.0**	**9.9**	**11.7**	**11.8**
All	✓	11.4	10.8	12.4	12.5
Questions	✗	4.5	4.5	5.5	5.8

be a viable strategy to concatenate several data stores unless sure of which one to pick.

The second partition of the table shows the results of using shallow fusion on each system. For the DQA test sets, we use an LSTM LM trained on the DQA training contexts for shallow fusion. For Squad, we use an LSTM LM trained on Wikitext-103 (the same one used as the retrieval LM), since both Squad and Wikitext are constructed from Wikipedia data, and the Squad dataset is too small to train a robust LM. This reflects one of the advantages of retrieval: we can add any relevant corpus, no matter how small, to the datastore without danger of overfitting to it. We optimize the LM interpolation weights separately on each validation set and apply them to the corresponding test sets. Retrieval is comparable to shallow fusion on DQA test sets and outperforms it on Squad. Furthermore, the two are complementary as further significant improvements can be obtained by combining retrieval with shallow fusion.

The final partition shows the result of retrieving not from a datastore of contexts but of the questions themselves (with an adapter trained for questions). This gives us an upper-bound, however unrealistic, on performance in the "Match" setting. We observe that by retrieving from datastores of questions, we can more than halve the WER to around 5% across all test sets. This indicates that even though we only retrieve 16 continuations at a time out of hundreds of thousands (after tokenization of #Words in Table 1), the main bottleneck is not in the retrieval itself, but the simple fact that the contexts and the questions are not perfectly matched. The residual WER tells us the inherent errors due to either the retriever failing to retrieve the correct continuations or the adapter failing to bias the ASR output.

3.4. Performance on named entities

Table 3 shows the results on the DQA test sets split by whether or not the reference word is a named entity. We report results only on DQA because, unlike DQA, the Squad dataset references have no named entity tags. We observe that retrieval generally improves named entities more than it does other words. Adding retrieval to the baseline RNN-T leads to relative WER improvements of 11% and 8% respectively on named entities and other words on the validation set. The respective improvements on the test set are 12% and 4%. We observe that shallow fusion improves more on regular words and less on named entities compared to retrieval. Finally, when we use shallow fusion and retrieval together, we get better named entity WER

Table 3: DQA dataset WERs for named entities and common words.

Datastore	S-F	DQA-V		DQA-T	
		Entities	Others	Entities	Others
None	✗	25.8	10.0	27.1	9.8
Fixed emb.	✗	27.1	10.6	28.5	10.4
Match	✗	**23.0**	**9.2**	**23.9**	**9.4**
All	✗	24.1	9.7	24.9	9.8
None	✓	24.6	**8.5**	25.6	**8.4**
Fixed emb.	✓	25.2	9.5	26.7	9.2
Match	✓	**21.4**	**8.5**	**22.4**	8.6
All	✓	22.7	9.0	24.6	9.0

Table 4: WER obtained by retrieving from the mixed datastore as the number of retrieved neighbors is varied.

Test set	-	1	2	4	8	16	32	64
Squad-V	15.8	16.2	15.1	13.4	12.7	12.3	12.2	**12.0**
Squad-T	15.2	16.3	14.8	13.3	12.4	11.9	11.6	**11.5**
DQA-V	14.0	17.0	15.0	14.1	13.7	13.3	13.1	**12.9**
DQA-T	13.8	16.4	14.9	13.8	13.5	13.3	13.2	**13.0**

that using either by itself, but the WER on other words does not get better than using shallow fusion by itself. This supports our thesis that the trained shallow fusion LM can do fine by itself for common words, and the utility of retrieval is most pronounced for rare words.

3.5. Impact of number of retrieved neighbors

Table 4 shows the WERs on the test sets as we vary K while retrieving from the datastore of all datasets ("All"). Note that we only vary K at inference time; during training it is still fixed to 16. We observe that the WER improves–with diminishing returns–as K increases.

In experiments whose results we omit due to space constraints, we varied K at training time and note that the higher we set K at training time, the higher we can, and have to, set it during inference. To get improved the results with lower values of K at inference, we need to train with low values of K. For instance, across test sets, the WERs obtained from setting $K = 1$ for both training and testing fall between those obtained from setting $K = 4$ and $K = 8$ in Table 4.

4. CONCLUSIONS

In this work, we have proposed augmenting an RNN-T based ASR model with a retrieval mechanism, which searches an external datastore for potential continuations of partial ASR hypotheses. We show that biasing subsequent decoding steps with these continuations significantly improves ASR performance, especially on named entities, when the datastore contains relevant text. We further show that retrieval can be complemented by a conventional shallow fusion LM, as using the two in tandem results in further improvements.

Avenues for future work include replacing the retrieval LM with a model trained explicitly for retrieval, further reducing performance degradation when the datastore and test set are unrelated, and improving efficiency by exploring ways to reduce retrieval frequency.

5. REFERENCES

[1] Peidong Wang, Tara N. Sainath, and Ron J. Weiss, "Multitask Training with Text Data for End-to-End Speech Recognition," in *Proc. Interspeech 2021*, 2021, pp. 2566–2570.

[2] Bolaji Yusuf, Ankur Gandhe, and Alex Sokolov, "USTED: Improving ASR with a Unified Speech and Text Encoder-Decoder," in *Proceedings of ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2022, pp. 8297–8301, IEEE Signal Processing Society.

[3] Samuel Thomas, Brian Kingsbury, George Saon, and Hong-Kwang J Kuo, "Integrating Text Inputs For Training and Adapting RNN Transducer ASR Models," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 8127–8131.

[4] Caglar Gulcehre et al., "On using monolingual corpora in neural machine translation," *arXiv preprint arXiv:1503.03535*, 2015.

[5] Jan Chorowski and Navdeep Jaitly, "Towards better decoding and language model integration in sequence to sequence models," in *Proc. Interspeech 2017*, 2017, pp. 523–527.

[6] Ehsan Variani, David Rybach, Cyril Allauzen, and Michael Riley, "Hybrid Autoregressive Transducer (HAT)," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 6139–6143.

[7] Erik McDermott, Hasim Sak, and Ehsan Variani, "A Density Ratio Approach to Language Model Fusion in End-to-End Automatic Speech Recognition," in *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 2019, pp. 434–441.

[8] Zhong Meng et al., "Internal language model training for domain-adaptive end-to-end speech recognition," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 7338–7342.

[9] Ding Zhao et al., "Shallow-Fusion End-to-End Contextual Biasing," in *Interspeech*, 2019, pp. 1418–1422.

[10] Aditya Gourav et al., "Personalization strategies for end-to-end speech recognition systems," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 7348–7352.

[11] Martin Kocour et al., "Boosting of Contextual Information in ASR for Air-Traffic Call-Sign Recognition," in *Proceedings Interspeech 2021*. 2021, vol. 2021, pp. 3301–3305, International Speech Communication Association.

[12] Golan Pundak, Tara N Sainath, Rohit Prabhavalkar, Anjuli Kannan, and Ding Zhao, "Deep context: end-to-end contextual speech recognition," in *2018 IEEE spoken language technology workshop (SLT)*. IEEE, 2018, pp. 418–425.

[13] Mahaveer Jain, Gil Keren, Jay Mahadeokar, Geoffrey Zweig, Florian Metze, and Yatharth Saraf, "Contextual RNN-T for Open Domain ASR," in *Proc. Interspeech 2020*, 2020, pp. 11–15.

[14] Feng-Ju Chang et al., "Context-Aware Transformer Transducer for Speech Recognition," in *2021 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 2021, pp. 503–510.

[15] Kanthashree Mysore Sathyendra et al., "Contextual adapters for personalized speech recognition in neural transducers," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 8537–8541.

[16] Saket Dingliwal, Monica Sunkara, Srikanth Ronanki, Jeff Farris, Katrin Kirchhoff, and Sravan Bodapati, "Personalization of ctc speech recognition models," in *IEEE 2022 Workshop on Spoken Language Technology*, 2022.

[17] Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis, "Generalization through Memorization: Nearest Neighbor Language Models," in *International Conference on Learning Representations (ICLR)*, 2020.

[18] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang, "Retrieval augmented language model pre-training," in *International Conference on Machine Learning*. PMLR, 2020, pp. 3929–3938.

[19] Junxian He, Graham Neubig, and Taylor Berg-Kirkpatrick, "Efficient nearest neighbor language models," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 2021, pp. 5703–5714.

[20] Pranav Rajpurkar, Robin Jia, and Percy Liang, "Know what you don't know: Unanswerable questions for SQuAD," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, Melbourne, Australia, July 2018, pp. 784–789, Association for Computational Linguistics.

[21] Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom, "Teaching machines to read and comprehend," *Advances in neural information processing systems*, vol. 28, 2015.

[22] Alex Graves, "Sequence transduction with recurrent neural networks," *arXiv preprint arXiv:1211.3711*, 2012.

[23] Amin Fazel, Wei Yang, Yulan Liu, Roberto Barra-Chicote, Yixiong Meng, Roland Maas, and Jasha Droppo, "SynthASR: Unlocking Synthetic Data for Speech Recognition," in *Proc. Interspeech 2021*, 2021, pp. 896–900.

[24] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur, "Librispeech: An ASR corpus based on public domain audio books," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015, pp. 5206–5210.

[25] Taku Kudo, "Subword regularization: Improving neural network translation models with multiple subword candidates," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2018, pp. 66–75.

[26] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher, "Pointer sentinel mixture models," in *ICLR*, 2017.

[27] Jeff Johnson, Matthijs Douze, and Hervé Jégou, "Billion-scale similarity search with GPUs," *IEEE Transactions on Big Data*, vol. 7, no. 3, pp. 535–547, 2019.

[28] Herve Jegou, Matthijs Douze, and Cordelia Schmid, "Product quantization for nearest neighbor search," *IEEE transactions on pattern analysis and machine intelligence*, vol. 33, no. 1, pp. 117–128, 2010.