INTEGRATING TEXT INPUTS FOR TRAINING AND ADAPTING RNN TRANSDUCER ASR MODELS

Samuel Thomas, Brian Kingsbury, George Saon, Hong-Kwang J. Kuo

IBM Research AI

ABSTRACT

Compared to hybrid automatic speech recognition (ASR) systems that use a modular architecture in which each component can be independently adapted to a new domain, recent end-to-end (E2E) ASR system are harder to customize due to their all-neural monolithic construction. In this paper, we propose a novel text representation and training framework for E2E ASR models. With this approach, we show that a trained RNN Transducer (RNN-T) model's internal LM component can be effectively adapted with text-only data. An RNN-T model trained using both speech and text inputs improves over a baseline model trained on just speech with close to 13% word error rate (WER) reduction on the Switchboard and CallHome test sets of the NIST Hub5 2000 evaluation. The usefulness of the proposed approach is further demonstrated by customizing this general purpose RNN-T model to three separate datasets. We observe 20-45% relative word error rate (WER) reduction in these settings with this novel LM style customization technique using only unpaired text data from the new domains.

***Index Terms*—** Automatic speech recognition, end-to-end models, RNN Transducers, adaptation, language model customization.

1. INTRODUCTION

With their remarkable performance and simplified training pipeline, end-to-end (E2E) models have become the de-facto approach to automatic speech recognition, replacing traditional hybrid ASR models that separately model different knowledge sources such as the language model, acoustic model, and lexicon. One advantage to the modular architecture of hybrid models, however, is that each component can be trained or adapted on separate, independent data sets. On the other hand, current all-neural E2E systems require transcribed training sets with paired speech and text transcripts. This limitation becomes profound, especially when ASR models need to be customized for new domains. With hybrid models, domain adaptation could be performed by adapting the language model on task- or domain-specific text-only data and updating the lexicon to cover any new words. We address this shortcoming of E2E models using a novel representation that effectively integrates text inputs into model training and adaptation.

To demonstrate the usefulness of our method we conduct a study on RNN-Transducer (RNN-T) models, a class of end-to-end, streamable, all-neural models that have been widely adopted for speech recognition [1–5]. RNN-T models typically consist of three different sub-networks: an encoder network, a prediction network, and a joint network [6] as shown in Figure 1. The encoder network or transcription network produces acoustic embeddings, while the prediction network resembles a language model in that it is conditioned on previous non-blank symbols produced by the model. The joint network combines the two embedding outputs to produce a posterior distribution over the output symbols. This architecture elegantly replaces a conventional hybrid ASR system composed of separate acoustic model, language model, pronunciation lexicon, and decoder components.

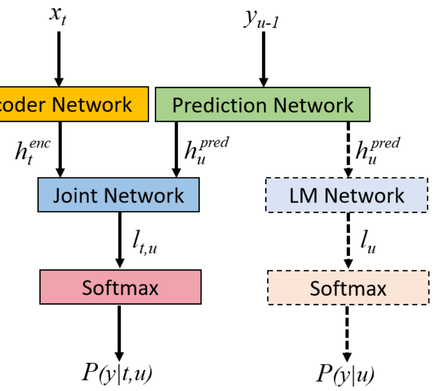

Fig. 1. Architecture of an RNN-T transducer ASR model with additional layers for NN-LM customization

Several methods have been proposed to adapt or customize these models to new domains. An external language model can be trained separately on data specific to the new task and then integrated with the existing RNN-T model via shallow fusion, density ratio fusion [7], or internal language model estimation [8]. Because scores from the external language model are combined with scores from the RNN-T model during decoding, a more complex decoder is necessary. Other methods attempt to directly update one or more parts of the RNN-T model, and thus do not require changes to decoding. A text-to-speech (TTS) model can be used to synthesize audio from text, and then the paired synthetic speech and text can be used to adapt the prediction network [9]. A more recent approach avoids synthesizing audio from text. This method first inserts a temporary LM layer to the prediction network as shown in Figure 1. With this new layer, the prediction network is then trained as a neural LM on external text data [10].

2. TEXTOGRAMS

In this work, we propose a novel approach to adapting RNN-T models to domains and tasks unseen during training. Like [10], our method uses only text data, does not require TTS, and does not require a more complex decoder that combines scores from multiple models. Unlike [10], our approach optimizes the RNN-T loss during adaptation. The main idea we use is to train the transcription network to work with two different input modalities: the stan-

978-1-6654-0540-9/22/$31.00 ©2022 IEEE

dard acoustic features (log-Mel spectrograms augmented with Δ and Δ^2 features in this work) and also a representation we call a "textogram". A textogram is simply a concatenation of the one-hot encodings of the symbols making up the reference text, with the encoding of each symbol being repeated some number of times such that the textograms have durations similar to spectrograms. For example, given a reference text *ideas* and an RNN-T model that operates on graphemes, textogram features are constructed by first splitting the word into its constituent graphemes, *i*, *d*, *e*, *a*, and *s*. Each symbol is assigned a duration, four frames in this case, to create a 2-dimensional representation as shown in Figure 2. This representation is very similar to posteriograms, which are symbol posterior probability estimates from a trained neural network acoustic model.

Once constructed in this fashion, these representations are used along with traditional speech based log-mel features to train RNN-T models. In this work, we simply increase the dimensionality of the input to the first layer of the transcription network to be the sum of the dimensionality of the acoustic features and the textogram features, and then fix the unused modality's inputs to 0.0 during training, but other designs are certainly possible.

To allow the model to learn robustly from the textogram representations, various variabilities can be added to this representation. These choices include:

(a) Label masking: To allow the model to learn useful n-gram sequences instead of blindly memorizing the text, active entries of the textogram representation can be randomly dropped. The rate of label masking is a parameter that can be empirically selected.

(b) Label confusions: The acoustic confusion among various speech sounds can be introduced into the textogram by substituting various labels with their confusable sounds e.g., *p* and *b*

(c) Variable label duration: The length of each linguistic unit can be varied to model real durations in the speech signal. In Figure 2 we use four frames per symbol.

(d) Modeling pronunciations: The input textogram may include different "sounds-like" sequences for a given target output. For example, the target *Miami*, may be associated with textogram sequences, *Miami*, *my Amy*, or *mee Amy*.

(e) Multiple linguistic symbol sets: The symbol set used with textograms can be different from the output symbol set for the transducer model. For example, phonetic targets can be used at the RNNT's output while graphemes are used for textograms.

In this work, we use a fixed label duration for various text symbols along with label masking, to construct textogram features

3. TRAINING AND ADAPTING RNN-T MODELS

3.1. Training RNN-T models

Using notation from [6], an RNN-T models the conditional distribution $p(\mathbf{y}|\mathbf{x})$ of an output sequence $\mathbf{y} = (y_1, \ldots, y_U) \in \mathcal{Y}^*$ of length $\mathcal{U}$ given an input sequence $\mathbf{x} = (x_1, \ldots, x_T) \in \mathcal{X}^*$ of length T. In an ASR setting, while the elements of $\mathbf{x}$ are continuous multidimensional speech features, $\mathbf{y}$ is discrete and corresponds to an output symbol set, like the grapheme set of the language being modelled by the network. To facilitate the alignment of the two sequences, which in general have different lengths, the output alphabet is augmented with an additional BLANK symbol. $p(\mathbf{y}|\mathbf{x})$ is computed by marginalizing over all possible alignments between $\mathbf{x}$ and $\mathbf{y}$.

The probability of a particular alignment is computed in terms of embeddings, h^{enc}, of the input sequence computed by the encoder network and embeddings, h^{pred}, of the output sequence computed by a prediction network. The joint network combines these two embeddings to produce a posterior distribution over the output symbols. Training is based on an efficient forward-backward algorithm, with $T \times U$ complexity for both loss and gradient computation, that minimizes $-\log p(\mathbf{y}|\mathbf{x})$, the negative log-likelihood loss and uses speech data paired with corresponding text transcripts. Representing text as textograms allows us to extend the RNN-T training framework. In addition to training the network using samples comprising paired speech and text data, $(\mathbf{x}^{sp}, \mathbf{y})$, where the speech is represented by an acoustic feature sequence $\mathbf{x}^{sp}$ and the text is represented by a symbol sequence $\mathbf{y}$, we also train the network using samples comprising paired text representations $(\mathbf{x}^{txt}, \mathbf{y})$, where $\mathbf{x}^{txt}$ is the textogram representation of the text and $\mathbf{y}$ is the symbol sequence.

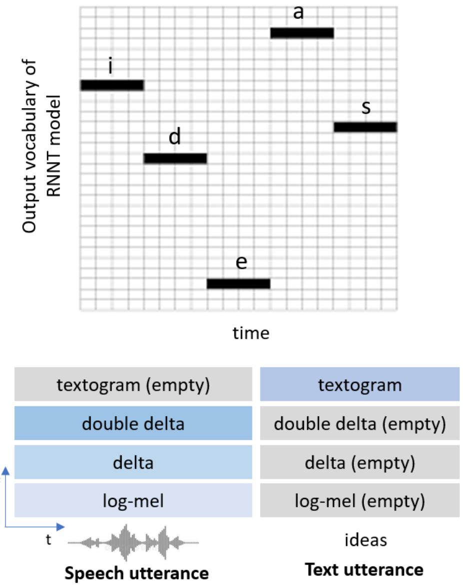

Fig. 2. Schematics of the textogram representation for an input text input, '*ideas*' (top), feature inputs used to train RNN-T models corresponding to speech and text (bottom).

For the (speech, text) samples in the training set, we extract log-mel features augmented with Δ and Δ^2 features and set the input dimensions corresponding to the textogram features to 0.0, as shown in Figure 2. To improve the robustness of the speech training, sequence noise injection [11] and SpecAugment [12] are applied to the speech features. For the (text, text) samples, we compute textogram features for each transcript and set the input dimensions corresponding to the acoustic features to 0.0, as shown in Figure 2. To prevent the (text, text) task from being completely trivial, we apply label masking to the textogram features. By integrating text inputs into the training pipeline, the RNN-T model's transcription network is now trained as a single encoder for two modalities: speech and text. With this joint training, the transcription network produces similar embeddings for both speech and text that can be further used along with a prediction and joint network that are shared by both modalities.

3.2. Adapting RNN-T models

Once an RNN-T model has been trained on both speech and text, it can be easily be adapted to a new domain using only text data. Prior to the adaptation process, the text-only adaptation data is converted into textogram features. The RNN-T model is then adapted using these features. The parameters of the transcription network are kept constant during the adaptation process, while the parameters of the prediction and joint networks may be updated. This ensures that the model is still acoustically robust while being able to effectively process data from the new domain.

4. EXPERIMENTS AND RESULTS

4.1. The RNN-T base model with textogram features

The RNN-T models used in our experiments are trained on a collection of US English telephony data including Switchboard, Fisher, and proprietary data. Each RNN-T model has several sub-networks as illustrated in Figure 1. The transcription network contains 6 bidirectional LSTM layers with 640 cells per layer per direction. The prediction network is a single unidirectional LSTM layer with only 1024 cells. The joint network projects the 1280-dimensional stacked encoder vectors from the last layer of the transcription net and the 1024-dimensional prediction net embedding each to 256 dimensions, combines them multiplicatively, and applies a hyperbolic tangent. Finally, the output is projected to 42 logits, corresponding to 41 characters plus BLANK, followed by a softmax. More details on training settings and design choices can be found in [13]. The RNN-T based ASR models are trained using 40-dimensional, global mean and variance normalized log-Mel filterbank features, extracted every 10 ms. These features are augmented with Δ and Δ^2 coefficients, every two consecutive frames are stacked, and every second frame is skipped, resulting in 240-dimensional vectors every 20 ms. Likewise, every two frames of the textogram representations are stacked, and every second frame is skipped, resulting in 84-dimensional vectors every 20 ms. Thus, the transcription network takes a 324-dimensional input. The speech data is augmented using speed and tempo perturbation with values in $\{0.9, 1.1\}$ for both speed and tempo separately, resulting in 4 additional speech training data replicas. For sequence noise injection, we add, with probability 0.8, to the spectrum of each training utterance the spectrum of one random utterance of similar length scaled by a factor of 0.4. For SpecAugment we used the settings published in [12].

Textogram representations of the text data are generated using the same grapheme set that is modelled at the outputs of the RNN-T and a fixed duration of four frames per symbol. Label masking at a rate of 25% is applied to the textograms to prevent the model from simply reproducing the input.

The RNN-T models are trained in Pytorch on V100 GPUs for 20 epochs using an AdamW optimizer. The maximum learning rate is set to 2e-4 and the OneCycleLR policy consists in a linear warmup phase from 2e-5 to 2e-4 over the first 6 epochs followed by a linear annealing phase to 0 for the remaining 14 epochs. We use an effective batch size of 128 utterances. Batches are constructed from feature sequences of similar lengths without regard to whether the features are mel spectrograms or textograms, so generally each batch will contain both types of training sample.

We train two RNN-T models in our first set of experiments: an RNN-T model on all the available speech data and a textogram based model trained on both speech and text. Both the models have the same architecture, except that the first layer in the transcription network has a larger input in the model trained on speech and text, and both models are constructed using the same training procedure described above. In Table 1 we report results on the commonly used Hub5 2000 Switchboard (SWB) and CallHome (CH) test sets, which are processed using LDC segmentations and scored using Kaldi scoring setups for measuring WER. An RNN-T model trained with the proposed textogram (TOG) method, TOG-RNN-T, significantly improves over a competitive baseline model trained on just speech data. We hypothesize that the relative WER reduction of 10% and 13% on SWB and CH using the model jointly trained with speech and text inputs is due to a regularization effect caused by training on two modalities using twice as much data. The twofold increase in training data is because we use the transcripts corresponding to the speech data as additional text inputs to train the model. From one perspective, then, using textograms can be seen as another approach to data augmentation when training RNN-T models for ASR.

Model	SWB	CH
RNN-T	6.9	11.9
TOG-RNN-T	6.2	10.5

Table 1. Baseline WER on Hub5'00 Switchboard and CallHome using only speech data (RNN-T) and the proposed approach with both speech and text inputs (TOG-RNN-T).

4.2. RNN-T adaptation to various domains

In our next set of experiments, we adapt the general purpose textogram based model to various new domains and settings. To measure the usefulness of our proposed technique we also implement the text-only adaptation technique proposed in [10] and compare results.

In [10], the prediction network is interpreted as a neural LM. As shown in Figure 1, to adapt the prediction network on text-only input, a temporary LM layer is first attached and trained along with a softmax output layer on training data transcripts using a standard cross-entropy loss. Once the LM layer has been trained, it is kept fixed while the prediction network is further adapted to novel text data from the new domain. This training uses two auxiliary losses for better regularization: a KL divergence loss that controls how similar the adapted model's output distribution is to the original unadapted model's distribution and a weight regularization loss that prevents the adapted model's weights from drifting away from the base model. Compared to this NN-LM based technique, our proposed textogram based method performs adaptation by optimizing the RNN-T loss rather than cross-entropy. We conduct our adaptation experiments on three diverse datasets.

A. Adaptation to the SLURP dataset: The SLURP dataset [14] is a recently released dataset for spoken language understanding. The audio data was collected in challenging acoustic conditions found in a typical home or office environment using far-field and close speaking microphones. The training partition of the dataset has about 11K text sentences. We adapt the base TOG-RNN-T model using text-only data from this dataset and test it on the corpus's speech test set, which corresponds to 10 hours of speech. The test data is also downsampled to 8 kHz for our use. Given that the SLURP dataset was collected for developing an in-home personal robot assistant, this domain is quite different from the original base model training data. Table 2 shows the performance of various adaptation techniques on this dataset. The unadapted model's WER is quite high since the SLURP dataset is substantially different both acoustically and lingustically. The neural LM based adaptation technique [10] reduces the WER by about 15% relative. We next adapt the RNN-T model in three different ways using our proposed method: (1) in *TOG adapt (P+J)* we adapt both the prediction and joint networks, (2) in *TOG adapt (P)* we adapt only the prediction network, and (3) in *TOG adapt (P) + NN-LM* we combine textogram adaptation with the NN-LM technique. Adapting the RNN-T base model with textogram based features significantly improves the WER reduction. We observe more gains by adapting just the prediction network than adapting both the prediction and the joint networks. Combining the NN-LM adaptation method provides further regularization to the textogram based adaptation and provides the best relative WER reduction of 23%.

Model	WER%
Unadapted TOG-RNN-T	47.5
NN-LM adapt [10]	40.4
TOG adapt (P+J)	38.6
TOG adapt (P)	37.6
TOG adapt (P) + NN-LM	36.5

Table 2. Performance of various models adapted to SLURP.

B. Adaptation to the ATIS dataset: In our second set of experiments we use the ATIS [15] training and test sets: 4976 training utterances from Class A (context independent) training data in the ATIS-2 and ATIS-3 corpora and 893 test utterances from the ATIS-3 Nov93 and Dec94 data sets. The test utterances comprise about 1.5 hours of audio from 55 speakers. The data was originally collected at 16 kHz, but is downsampled to 8 kHz to match the telephony base model. We repeat the same set of adaptation experiments on this dataset as well. Although the WER of the unadapted model is much lower compared to results on SLURP, the model still benefits from adaptation. Similar to previous results, the NN-LM adaptation technique improves WER reduction by 30% relative. Although the proposed textogram adaptation of just the prediction network improves on top of these gains to 40% relative WER reduction, combining with the NN-LM improves the performance further by 45% relative.

Model	WER%
Unadapted TOG-RNN-T	3.1
NN-LM adapt [10]	2.2
TOG adapt (P+J)	1.8
TOG adapt (P)	1.9
TOG adapt (P) + NN-LM	1.7

Table 3. Performance of various models adapted to ATIS.

C. Adaptation to the Harper Valley dataset: In our final set of experiments, we adapt the baseline RNN-T model to the Harper Valley Bank corpus [16]. This dataset is a public domain corpus with spoken dialogs that simulate simple consumer banking interactions between users and agents. There are 1,446 human-human conversations between 59 unique speakers. We adapt the model on 15K text transcripts from the training partition used in [17] before testing the adapted model on a 1.5 hour speech test set.

Model	WER%
Unadapted TOG-RNN-T	10.4
NN-LM adapt [10]	7.5
TOG adapt (P+J)	7.3
TOG adapt (P)	7.0
TOG adapt (P) + NN-LM	6.8

Table 4. Performance of various models adapted to Harper Valley.

Similar to the previous two customization experiments, we see significant performance gains using the proposed adaptation techniques. Consistent gains are observed by adapting just the prediction network using the text-only transcripts. Compared to the NN-LM method, the textogram based method is able to better adapt this sub-network. We hypothesize that this is because of the holistic nature of the adaptation process where the prediction network is not adapted independently using a separate loss, but is adapted along with embeddings from the transcription network using the original RNN-T training loss, via the joint network. In this case as well, we see up to 34% relative WER reduction after adaptation compared to the unadapted model. Combining the NN-LM training loss with the RNN-T training loss gives the best improvements, suggesting that the prediction network can benefit further when good regularization is used.

For all our adaptation experiments, we train for 20 epochs using an AdamW optimizer and a OneCycleLR policy set to a maximum learning rate of 2e-4. In our experiments where we integrate the NN-LM training loss with the RNN-T training loss, we weight the NN-LM loss by an empirically determined weight of 2e2 before combining it with the original RNN-T loss. While generating textogram features in both train and adaptation, certain parameters have been selected intuitively. The frame duration of each constituent symbol in the textogram representation is currently set to 4 as a close approximation to an average speaking rate of 4-5 syllables (12-15 symbols) per second. Similarly, while stacking feature frames, we stack textogram features in exactly the same way that we stack speech features. We have also not yet explored the use of label confusions, multiple linguistic symbol sets or modeling pronunciations. These parameters and settings need be explored further.

In our final set of experiments we compare the performance of the adapted TOG-RNN-T model with our baseline RNN-T model combined with an external LM. The external NN-LM is a two layer LSTM with 1024 nodes trained on the same adaptation data used to adapt the TOG-RNNT model. The scores from the external LM and the RNN-T model are combined via shallow fusion [13]. Although improvements with shallow fusion are observed (row 2 vs. row 1 of Table 5), the gains using the TOG model are equally significant (row 3). Compared to the shallow fusion approach, our proposed method has several benefits. Adapting the prediction network via textograms does not require a more complex decoder that combines scores from multiple models, while at the same time providing better performance. The results with shallow fusion were obtained after carefully tuning the external LM weight. The need to optimize the LM weight is also eliminated with our method.

Model	SLP	ATIS	HVB
Unadapted RNN-T	51.3	3.6	11.3
+ Shallow Fusion	41.8	1.9	7.0
TOG adapt (P) + NN-LM	36.5	1.7	6.8

Table 5. Comparison of baseline RNN-T WER performance with shallow fusion on SLURP (SLP), ATIS and Harper Valley (HVB).

5. CONCLUSIONS

In this paper, we have presented a novel representation for training E2E ASR models with additional text inputs. We have shown that this framework reduces the need for paired speech-text data and allows for a simplified adaptation pipeline as well. This work is also novel since we now show how a single ASR encoder can be trained within the RNN-T framework for two modalities: speech and text. The usefulness of the proposed approach is demonstrated by customizing an RNN-T model to three separate datasets with text-only data. We observe 20-45% relative word error rate (WER) reduction in these settings with this novel technique.

6. REFERENCES

[1] Yanzhang He, Tara N Sainath, Rohit Prabhavalkar, Ian McGraw, Raziel Alvarez, Ding Zhao, David Rybach, Anjuli Kannan, Yonghui Wu, Ruoming Pang, et al., "Streaming end-to-end speech recognition for mobile devices," in *Proc. ICASSP*, 2019.

[2] Kanishka Rao, Haşim Sak, and Rohit Prabhavalkar, "Exploring architectures, data and units for streaming end-to-end speech recognition with RNN-transducer," in *Proc. ASRU*, 2017.

[3] Jinyu Li, Rui Zhao, Hu Hu, and Yifan Gong, "Improving RNN transducer modeling for end-to-end speech recognition," in *Proc. ASRU Workshop*, 2019.

[4] Laurent El Shafey, Hagen Soltau, and Izhak Shafran, "Joint speech recognition and speaker diarization via sequence transduction," in *Proc. Interspeech*, 2019.

[5] Mohammadreza Ghodsi, Xiaofeng Liu, James Apfel, Rodrigo Cabrera, and Eugene Weinstein, "RNN-transducer with stateless prediction network," in *Proc. ICASSP*, 2020.

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

[7] Erik McDermott, Hasim Sak, and Ehsan Variani, "A density ratio approach to language model fusion in end-to-end automatic speech recognition," in *Proc. ASRU*, 2019.

[8] Zhong Meng, Sarangarajan Parthasarathy, Eric Sun, Yashesh Gaur, Naoyuki Kanda, Liang Lu, Xie Chen, Rui Zhao, Jinyu Li, and Yifan Gong, "Internal language model estimation for domain-adaptive end-to-end speech recognition," in *Proc. SLT*, 2021.

[9] Gakuto Kurata, George Saon, Brian Kingsbury, David Haws, and Zoltán Tüske, "Improving customization of neural transducers by mitigating acoustic mismatch of synthesized audio," in *Proc. Interspeech*, 2021.

[10] Janne Pylkkönen, Antti Ukkonen, Juho Kilpikoski, Samu Tamminen, and Hannes Heikinheimo, "Fast text-only domain adaptation of rnn-transducer prediction network," in *Proc. Interspeech*, 2021.

[11] George Saon, Zoltán Tüske, Kartik Audhkhasi, and Brian Kingsbury, "Sequence noise injected training for end-to-end speech recognition," in *Proc. ICASSP*, 2019.

[12] Daniel S Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D Cubuk, and Quoc V Le, "SpecAugment: A simple data augmentation method for automatic speech recognition," *arXiv preprint arXiv:1904.08779*, 2019.

[13] George Saon, Zoltán Tüske, Daniel Bolanos, and Brian Kingsbury, "Advancing RNN transducer technology for speech recognition," in *Proc. ICASSP*, 2021.

[14] Emanuele Bastianelli, Andrea Vanzo, Pawel Swietojanski, and Verena Rieser, "Slurp: A spoken language understanding resource package," in *Proc. EMNLP*, 2020.

[15] Charles T Hemphill, John J Godfrey, and George R Doddington, "The ATIS spoken language systems pilot corpus," in *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania*, 1990.

[16] Mike Wu, Jonathan Nafziger, Anthony Scodary, and Andrew Maas, "Harpervalleybank: A domain-specific spoken dialog corpus," *arXiv preprint arXiv:2010.13929*, 2020.

[17] Jatin Ganhotra, Samuel Thomas, Hong-Kwang J Kuo, Sachindra Joshi, George Saon, Zoltán Tüske, and Brian Kingsbury, "Integrating dialog history into end-to-end spoken language understanding systems," in *Proc. Interspeech*, 2021.