INSTANT ONE-SHOT WORD-LEARNING FOR CONTEXT-SPECIFIC NEURAL SEQUENCE-TO-SEQUENCE SPEECH RECOGNITION

Christian Huber[1]*, Juan Hussain*[1]*, Sebastian Stüker*[1] *and Alexander Waibel*[1,2]

[1]Interactive Systems Lab, Karlsruhe Institute of Technology, Karlsruhe, Germany
[2]Carnegie Mellon University, Pittsburgh PA, USA

firstname.lastname@kit.edu, alexander.waibel@cmu.edu

ABSTRACT

Neural sequence-to-sequence systems deliver state-of-the-art performance for automatic speech recognition (ASR). When using appropriate modeling units, e.g., byte-pair encoded characters, these systems are in principal open vocabulary systems. In practice, however, they often fail to recognize words not seen during training, e.g., named entities, numbers or technical terms. To alleviate this problem we supplement an end-to-end ASR system with a word/phrase memory and a mechanism to access this memory to recognize the words and phrases correctly. After the training of the ASR system, and when it has already been deployed, a relevant word can be added or subtracted instantly without the need for further training. In this paper we demonstrate that through this mechanism our system is able to recognize more than 85% of newly added words that it previously failed to recognize compared to a strong baseline.

Index Terms: speech recognition, one-shot learning, new-word learning

1. INTRODUCTION

Up until recently automatic speech recognition (ASR) systems were implemented as Bayes classifiers in order to search for the word sequence $\hat{W}$, among all possible word sequences W, with the highest posterior probability given a sequence of feature vectors X which is the result of pre-processing the acoustic signal to be recognized:

$$\begin{aligned}\hat{W} &= \underset{W}{\operatorname{argmax}}\, P(W|X) \\ &= \underset{W}{\operatorname{argmax}}\, P(X|W)P(W)\end{aligned}$$

In the context of ASR $P(X|W)$ is called the acoustic model, $P(W)$ the language model. The space of allowed word sequences to search among was usually defined by a list of words, the vocabulary, of which permissible word sequences could be composed. Words that were not in the vocabulary could not be recognized. In turn this means that by adding words to the vocabulary and appropriate probabilities to the language model, previously unknown words could be easily added to the ASR system manually or even automatically.

In contrast, for neural sequence-to-sequence trained end-to-end ASR systems, this is not possible anymore. While in principal end-to-end systems are open-vocabulary systems, when using appropriate modeling units, such as byte-pair encoded (BPE) characters, in practice, words not seen during training are often not reliably recognized. This is especially true for named entities. The reasons for that are that, a) the end-to-end network implicitly learns language model knowledge when being trained on transcribed speech data, and b) especially named entities often have a grapheme-to-phoneme relation that deviates from the general pronunciation rules of the language, as learnt implicitly by the networks of the end-to-end system.

In order to solve this problem, in this paper we extend an end-to-end ASR system by a memory for words and phrases. We further introduce a mechanism that enables the system to recognize the words stored in this memory without further training. This enables us to add new words and short phrases (only text, no audio) to the recognition system, long after the training of the ASR system has finished, and when it has already been deployed, without the need for further training.

This is achieved by, a) a memory-attention layer which predicts the availability and location of relevant information in the memory, and b) a memory-entry-attention layer which extracts the information of a memory entry.

2. RELATED WORK

The problem of new or rare words in speech recognition with sequence-to-sequence models is an active field of research. The closest to our work is [1]. In their work, they train a LSTM-based ASR model based on [2] which outputs graphemes. The model is supplied by dynamically biased context of rare words. [3] injects additional phonetic information on top of that. [4] predicted a vector presenting the context of a word with a low confidence (misrecognized) and

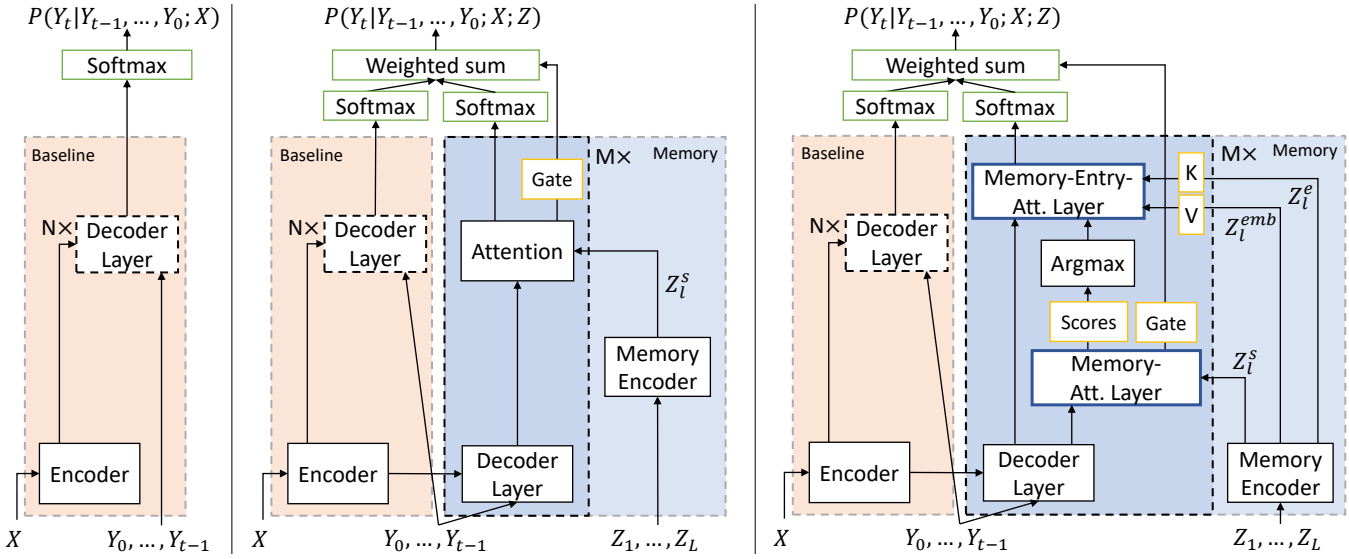

Fig. 1: Left: Baseline model, Middle: MEM model similar to [1], Right: 2MEM model. Dark blue: The two main components that are added compared to the baseline.

searched for the nearest vectors in a big corpus to find a substitution. In [5] a neural machine translation is augmented by a memory of word pairs to translate unseen or sparse pairs in the training data, but only for single words to be translated.

In our work, we build a memory for new words and phrases. The first modeling of a memory was done with Recurrent Neural Networks (RNNs) and Long-Short Term Memory (LSTMs), which capture long-term structure within the sequence in a latent state (local memory). In praxis, many applications require addressable memory (global memory) for the neural network in a kind of Neural Turing Machines [6]. For instance, Memory Networks [7] and End-To-End Memory Networks [8] are used to store answers for question-answering applications. Similarly, Pointer Networks [9] circumvent the out-of-vocabulary (OOV) problem by copying words from the input sequence that are not contained in the output vocabulary. The network does not possess any memory. A well-known example of a Pointer Network is the summarization with Pointer-Generator Networks [10]. [11] build a memory-augmented neural network for meta-learning which saves sample-label pairs and predicts upon the similarity of the input and the samples encoded in the memory. The most frequent memory samples are injected into the weights as long-term memory. [12] aim to automatically spell-correct ASR output text using a pointer network to copy out-of-vocabulary (OOV) words directly. They incorporate phonetic information as well, however they do not mention any kind of attending to audio as in our work.

Other related works address the problem of adaptation to a new domain or enhance the model by rescoring the output with a language model via two pass decoding. These works does not aim to solve the problem of OOV words directly. [13] incorporate a multi corpora language model for second pass rescoring, while [14] and [15] rescore with a second model by attending to the audio or as in [16], where the second model attends to both the audio and the output using a deliberation model.

3. METHOD

3.1. Baselines

As baseline we use a transformer-based sequence-to-sequence model [17, 18]. It consists of 24 encoder layers and 8 decoder layers with a model dimension of 1024 and feed-forward dimension of 2048 (see fig. 1, left). The inputs X to the encoder are mel-filterbank coefficients with 40 features. Before the encoder layers a two-layer convolutional neural network with 32 channels and time stride two is used to downsample the input spectrogram [19]. The decoder has as input the BPE tokens of the already decoded sequence $Y_0,\dots,Y_{t-1}$ and predicts a probability distribution $P(Y_t|Y_0,\dots,Y_{t-1};X)$ over the BPE vocabulary. The BPE consists of 4000 tokens.

For comparison we trained a memory-enhanced model (MEM) similar to [1] (see fig. 1, middle). However we replaced the LSTMs they used with a transformer model. Furthermore we run the baseline decoder together with the memory decoder and combine the output at the end (see section 3.3). This has the advantage that we are able to change the baseline after the training.

Our proposed model[1] (see fig. 1, right) changes the memory-decoder compared to the MEM model such that the output is computed in a two-step fashion. First, the availabil-

[1] A demo can be found under `https://huberchristian.eu`

ity and location of relevant information is predicted, and second, the information of a memory entry is extracted. We call this the two-step memory-enhanced model (2MEM). Both the MEM model and the 2MEM model predict a probability distribution $P(Y_t|Y_0, \ldots, Y_{t-1}; X; Z)$ over the BPE vocabulary, where $Z = (Z_1, \ldots, Z_L)$ are additional words/phrases we want the model to recognize.

To demonstrate that our model is able to be used with any baseline model we also evaluate it using a LSTM-based baseline [19]. This baseline consists of 6 encoder layers and 2 decoder layers with a model dimension of 1536.

3.2. Memory Encoder

Our memory can contain (similar to [1]) words or phrases in each memory entry. We represent each memory entry as BPE and refer to them as $Z_1, \ldots, Z_L$, where $Z_l \in \{1, \ldots, |V_{BPE}|\}^{e_l}$, $l \in \{1, \ldots, L\}$. The term e_l denotes the number of tokens in the memory entry Z_l.

The memory encoder starts with an embedding layer which output is denoted by $Z_l^{emb} \in \mathbb{R}^{e_l \times d_{model}}$, where d_{model} is the dimension of the model. Then a transformer encoder is applied. It consists of eight layers, if not mentioned otherwise. The output of the encoder is denoted by $Z_l^e \in \mathbb{R}^{e_l \times d_{model}}$.

After that we compute the mean of each encoded memory entry Z_l^e over the BPE tokens of the entry. This leaves us with one vector per memory entry, which can be interpreted as summary vector of the memory entry, denoted by $Z_l^s \in \mathbb{R}^{d_{model}}$. We add to these vectors Z_l^s a learned dummy vector Z_0^s, which is later used to determine when there is no relevant information in the memory.

3.3. Memory Decoders

Both the decoders in the MEM model and the 2MEM model consist of M blocks. In our experiments we use $M = 4$ if not mentioned otherwise. Each block starts with a decoder layer similar to the baseline model. Then a memory-attention layer is applied. It extracts the availability and location of relevant information of the memory. This is done by calculating similarity scores, where for the query input Q and the key input K the decoder layer output and Z_l^s are used, respectively.

$$\begin{aligned} \hat{Q} &= W^Q Q, && \text{shape } T \times d_{model} \\ \hat{K} &= W^K K, && \text{shape } (L+1) \times d_{model} \\ scores &= \hat{Q}\hat{K}^T, && \text{shape } T \times (L+1) \\ gate &= scores[:, 0], && \text{shape } T \end{aligned}$$

where T is the number of tokens in the target sequence, W^Q and W^K are weight matrices and $scores[:, 0]$ are the similarity scores corresponding to the learned dummy vector Z_0^s.

The main difference between the MEM model and the 2MEM model is the way in which the information is extracted of the memory. For the MEM model all the information of a memory entry has to be encoded in the vector Z_l^s. For the 2MEM model the vector Z_l^s has to only contain the information if a memory entry is currently relevant. If thats the case the memory-entry-attention layer can extract the information.

In the MEM model the score output of the memory-attention layer is processed by a softmax layer and the result is multiplied with a linear transformation of values V generated by Z_l^s.

$$\begin{aligned} \hat{V} &= W^V V, && \text{shape } (L+1) \times d_{model} \\ output &= sm(scores)\hat{V}, && \text{shape } T \times d_{model} \end{aligned}$$

where W^V is a weight matrix, sm is the softmax function, which is computed w.r.t. the last dimension. The memory-attention layer together with this transformation is equivalent to an attention layer similar to the encoder-decoder attention. The difference is that the gate output is used in the further processing.

In the 2MEM model the memory-entry-attention layer is used to extract relevant information of a specific memory entry. This is done by using an attention mechanism similar to the encoder-decoder attention, however not the vectors Z_l^s are used to generate the values as in the MEM model. For the query again the decoder layer output is used and for the keys and values, we use Z_l^e and Z_l^{emb}, respectively, where l is the argmax of the scores of the memory-attention layer for a given query.

Let $l = \text{argmax}(scores_t)$ be the index of the largest score for the query Q_t at time step t. If l is not zero, i.e. the index does not correspond to the learned dummy vector, the following is computed:

$$\begin{aligned} \hat{Q} &= \bar{W}^Q Q_t, && \text{shape } d_{model} \\ \hat{K} &= \bar{W}^K Z_l^e, && \text{shape } e_l \times d_{model} \\ \hat{V} &= \bar{W}^V Z_l^{emb}, && \text{shape } e_l \times d_{model} \\ output &= sm(\hat{Q}\hat{K}^T)\hat{V}, && \text{shape } T \times d_{model} \end{aligned}$$

where $\bar{W}^Q$, $\bar{W}^K$ and $\bar{W}^V$ are weight matrices. If the maximum score corresponds to the learned dummy vector (see section 3.2), no attention is calculated. A residual connection is used around the attention layer in the MEM model and the memory-entry-attention layer in the 2MEM model.

For both the MEM model and the 2MEM model, the outputs of the baseline decoder and the memory decoder are combined at the end by a weighted sum. The weighting is calculated as a linear transformation with one output neuron of the stacked gate outputs of all memory attention layers followed by a sigmoid layer.

4. TRAINING

During training we freeze the original baseline components and train only the memory related components.

4.1. Memory Content during Training

The content of the memory during training is sampled for each batch dependent on the target labels of the batch. We randomly select up to three words of a target label and use these as a memory entry. In total we sample 200 memory entries for each batch.

4.2. Loss Function

The loss function L consists of two cross entropy parts. The first one classifies the next token Y_t of the sequence and the second one classifies the most important memory entry:

$$L = \frac{1}{T}\sum_{t=1}^{T} CE(f_\theta(Y_{t-1}, \dots, Y_0; X; Z), Y_t) + \frac{\lambda}{MT}\sum_{m=1}^{M}\sum_{t=1}^{T} CE(scores_t^m, label_t^{mem}),$$

where $f_\theta(Y_{t-1}, \dots, Y_0; X; Z)$ is the output distribution of the decoder given the already decoded sequence $Y_{t-1}, \dots, Y_0$, the audiofeatures X and the memory/context $Z = (Z_l)_l$. Furthermore, $scores_t^m \in \mathbb{R}^{L+1}$ are the score output of the memory attention layer in the m-th memory decoder block and $label_t^{mem} \in \{0, \dots, L\}$ is the memory label. This memory label is constructed when sampling the memory content (see section 4.1). It contains the index zero of the learned dummy vector if a token is not chosen for the memory and otherwise the index of the corresponding memory entry. We use $\lambda = 1$ and label smoothing of 0.1.

We experienced that the MEM model learns to only focus on the baseline decoder output. Therefore, we force the model to use the memory when advantageous. This is done by permuting the output of the baseline decoder and the memory decoder dependent on $label^{mem}$ with a certain probability before the weighted sum is applied. This forces the model to use the memory decoder output if there is relevant information in the memory and the baseline decoder output otherwise. In particular, the probabilities of the label index and a random other index of the baseline decoder output are interchanged where $label^{mem}$ is non-zero and the same is done for the memory decoder output where $label^{mem}$ is zero. From the permuted probabilities no gradient is propagated backwards to not confuse the model.

4.3. Data

For training and evaluation of our models, we used Mozilla Common Voice v6.1 [20], Europarl [21], How2 [22], Librispeech [23], MuST-C v1 [24], MuST-C v2 [25] and Tedlium v3 [26] datasets. The data split is presented in the following table 1.

Furthermore we created a new-words testset containing 239 utterances each containing a word, e.g. named entities, we assumed the model could do wrong because it has not seem these words during training. We use this new-words testset to evaluate the performance of the models to learn these new words when they are provided as additional context through the memory.

Corpus	Utterances	Speech data [h]
A: Training Data		
Mozilla Common Voice	1225k	1667
Europarl	33k	85
How2	217k	356
Librispeech	281k	963
MuST-C v1	230k	407
MuST-C v2	251k	482
Tedlium	268k	482
B: Test Data		
Tedlium	1155	2.6
New-words testset	239	0.5

Table 1: Summary of the data-sets used.

5. RESULTS

5.1. Evaluation

For the evaluation we report an accuracy on new-words testset with empty and full memory as well as the WER on the ted testset also with empty and full memory. As memory entries we use all the (239) new words of the new-words testset. The accuracy is thereby calculated such that the output of the model is counted as correct if the new word is present in the transcript.

The results can be seen in table 2. We obtained that the MEM model learned to only focus on the baseline decoder. A reinitialization of the parameters in the weighted sum during training did not help. This changes when permuting the output probabilities of the decoder (see section 4.2) with probability 0.5. The accuracy on the new-words testset increases from 44.8% to 79.5%.

Surprisingly, the 2MEM model works well on the new-words testset without permuted output probabilites (88.7%), however there is also a performance gain when permuting the output probabilities of the 2MEM model (90.4%).

On the ted testset the performance drops slightly for all models with good performance on the new-words testset. This happens since the models sometimes uses a word in the memory when the target word starts similarly.

In table 3 an example of correcting the sentence "[we will] hopefully have covid nineteen behind [us soon]" with the model 2MEM with permutation probability 0.5 can be seen. The word covid nineteen was not seen in the training data.

Metric	Acc.↑ (%) new words testset		WER↓ (%) ted testset	
Model \ Full memory	✗	✓	✗	✓
Transformer baseline	45.2		5.0	
MEM model	44.8	44.8	4.8	4.8
+ permute 0.5	44.4	80.8	5.0	5.3
2MEM model	42.3	88.7	4.9	5.2
+ permute 0.5	45.2	**90.4**	5.0	5.2

Table 2: Summary of the results. Accuracy on the new-words test set and WER on the ted test set.

a) Decoding with empty memory

hope	fully	have	co	ve	t	nin	et	een	behind
0.90	0.88	0.89	0.83	0.89	0.93	0.93	0.88	0.88	0.89
0	0	0	0	0	0	0	0	0	0

b) Decoding with word "covid" as Z_1

hope	fully	have	co	v	id	nin	et	een	behind
0.90	0.88	0.87	**0.03**	**0.03**	**0.03**	0.92	0.88	0.88	0.89
0	0	0	**1**	**1**	**1**	0	0	0	0

Table 3: Example for correcting "covid nineteen". Rows: Predicted tokens, gate (weighting between baseline decoder and memory decoder) and memory location output (argmax of the scores; all layers have the same argmax in this example).

5.2. Ablation

For the ablation we start with the best performing model, the 2MEM model with permutation probability 0.5, and explore different changes. The results can be seen in table 4.

Metric	Acc.↑ (%) new words testset		WER↓ (%) ted testset	
Model \ Full memory	✗	✓	✗	✓
LSTM baseline	47.3		3.9	
2MEM model, permute 0.5	45.2	90.4	5.0	5.2
+ encode values	44.8	90.4	4.9	5.2
+ residual connection	44.8	88.3	5.1	5.3
+ LSTM baseline	46.9	**92.1**	3.9	**4.2**
one memory encoder layer	43.1	84.5	5.3	6.0
one memory decoder layer	42.7	87.9	5.2	7.1

Table 4: Summary of the ablation results. Accuracy (in %) on the new-words test set, WER on the Ted test set.

We tried to use Z_l^e instead of Z_l^{emb} (see sections 3.2 and 3.3) as value input for the memory-entry-attention layer. We refer to this as encode values and obtained that using these values delivers almost the same performance. Furthermore, we tried to add residual connections to the scores of the memory-attention layer. The performance on the new-words testset as well as on the ted testset decreased. As already mentioned in section 3.1 our model can be used with any baseline. We substituted our transformer-based baseline with a LSTM-based one which has a better performance on the ted testset. We obtain that this performance increase transfers to the new-words testset. We also trained models with only one memory encoder or one memory decoder layer. We see that this also works on the new-words testset, however the performance on the ted testset degrades to 6.0% and 7.1%, respectively, when the new words are in the memory. It seems that the memory decoder layers are more critical than the memory encoder layers.

Overall the 2MEM model with LSTM-based baseline performed the best with 92.1% accuracy on our new-words testset. This model was able to correct more than 85% of the errors via the memory component.

6. CONCLUSION AND FUTURE WORK

In this paper we demonstrated that it is possible to enhance a sequence-to-sequence ASR model with a memory component that allows to add new words to the recognition system, e.g., named entities, during deployment without the need for further training. In this way more than 85% of the previously misrecognized words not seen during training, were recognized correctly. Our 2MEM model outperforms the MEM model (which is similar to the model in [1]) significantly.

One drawback of the proposed method is that one only provides text for the memory. Therefore, if the pronunciation of a new word deviates from the implicitly learned pronunciation rules, it is unlikely to be found by the memory-attention layer. We therefore plan to extend our model by populating the memory with an audio sample of the new word in addition to text. Furthermore, a subword regularization [27], where a word can be encoded into multiple BPE candidates, could improve the performance of the system.

7. ACKNOWLEDGEMENTS

The projects on which this paper is based were funded by the Federal Ministry of Education and Research (BMBF) of Germany under the numbers 01IS18040A and 01EF1803B.

8. REFERENCES

[1] 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.

[2] William Chan, Navdeep Jaitly, Quoc Le, and Oriol Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition,"

in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, pp. 4960–4964.

[3] Antoine Bruguier, Rohit Prabhavalkar, Golan Pundak, and Tara N Sainath, "Phoebe: Pronunciation-aware contextualization for end-to-end speech recognition," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 6171–6175.

[4] Alejandro Coucheiro-Limeres, Fernando Fernández-Martínez, Rubén San Segundo, and Javier Ferreiros López, "Attention-based word vector prediction with lstms and its application to the oov problem in asr.," in *INTERSPEECH*, 2019, pp. 3520–3524.

[5] Yang Feng, Shiyue Zhang, Andi Zhang, Dong Wang, and Andrew Abel, "Memory-augmented neural machine translation," *arXiv preprint arXiv:1708.02005*, 2017.

[6] Alex Graves, Greg Wayne, and Ivo Danihelka, "Neural turing machines," *arXiv preprint arXiv:1410.5401*, 2014.

[7] Jason Weston, Sumit Chopra, and Antoine Bordes, "Memory networks," *arXiv preprint arXiv:1410.3916*, 2014.

[8] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus, "End-to-end memory networks," *arXiv preprint arXiv:1503.08895*, 2015.

[9] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly, "Pointer networks," *arXiv preprint arXiv:1506.03134*, 2015.

[10] Abigail See, Peter J Liu, and Christopher D Manning, "Get to the point: Summarization with pointer-generator networks," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2017, pp. 1073–1083.

[11] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap, "One-shot learning with memory-augmented neural networks," *arXiv preprint arXiv:1605.06065*, 2016.

[12] Yu-Chieh Chao and Chia-Hui Chang, "Automatic spelling correction for asr corpus in traditional chinese language using seq2seq models," in *2020 International Computer Symposium (ICS)*. IEEE, 2020, pp. 553–558.

[13] Anirudh Raju, Denis Filimonov, Gautam Tiwari, Guitang Lan, and Ariya Rastrow, "Scalable multi corpora neural language models for asr," *arXiv preprint arXiv:1907.01677*, 2019.

[14] Ankur Gandhe and Ariya Rastrow, "Audio-attention discriminative language model for asr rescoring," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 7944–7948.

[15] Tara N Sainath, Ruoming Pang, David Rybach, Yanzhang He, Rohit Prabhavalkar, Wei Li, Mirkó Visontai, Qiao Liang, Trevor Strohman, Yonghui Wu, et al., "Two-pass end-to-end speech recognition," *arXiv preprint arXiv:1908.10992*, 2019.

[16] Ke Hu, Tara N Sainath, Ruoming Pang, and Rohit Prabhavalkar, "Deliberation model based two-pass end-to-end speech recognition," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 7799–7803.

[17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[18] Ngoc-Quan Pham, Thai-Son Nguyen, Jan Niehues, Markus Müller, and Alex Waibel, "Very deep self-attention networks for end-to-end speech recognition," *Proc. Interspeech 2019*, pp. 66–70, 2019.

[19] Thai-Son Nguyen, Sebastian Stueker, Jan Niehues, and Alex Waibel, "Improving sequence-to-sequence speech recognition training with on-the-fly data augmentation," *arXiv preprint arXiv:1910.13296*, 2020.

[20] Rosana Ardila, Megan Branson, Kelly Davis, Michael Henretty, Michael Kohler, Josh Meyer, Reuben Morais, Lindsay Saunders, Francis M Tyers, and Gregor Weber, "Common voice: A massively-multilingual speech corpus," *arXiv preprint arXiv:1912.06670*, 2019.

[21] Philipp Koehn, "Europarl: A parallel corpus for statistical machine translation," in *MT summit*. Citeseer, 2005, vol. 5, pp. 79–86.

[22] Ramon Sanabria, Ozan Caglayan, Shruti Palaskar, Desmond Elliott, Loïc Barrault, Lucia Specia, and Florian Metze, "How2: a large-scale dataset for multimodal language understanding," *arXiv preprint arXiv:1811.00347*, 2018.

[23] 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)*. IEEE, 2015, pp. 5206–5210.

[24] Mattia A. Di Gangi, Roldano Cattoni, Luisa Bentivogli, Matteo Negri, and Marco Turchi, "MuST-C: a Multilingual Speech Translation Corpus," in *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, June 2019.

[25] Roldano Cattoni, Mattia Antonino Di Gangi, Luisa Bentivogli, Matteo Negri, and Marco Turchi, "Must-c: A multilingual corpus for end-to-end speech translation," *Computer Speech & Language*, vol. 66, pp. 101155, 2021.

[26] François Hernandez, Vincent Nguyen, Sahar Ghannay, Natalia Tomashenko, and Yannick Estève, "Ted-lium 3: twice as much data and corpus repartition for experiments on speaker adaptation," in *International Conference on Speech and Computer*. Springer, 2018, pp. 198–208.

[27] Taku Kudo, "Subword regularization: Improving neural network translation models with multiple subword candidates," *arXiv preprint arXiv:1804.10959*, 2018.