Streaming on-device end-to-end ASR system for privacy-sensitive voice-typing

Abhinav Garg, Gowtham Vadiseti, Dhananjaya Gowda, Sichen Jin, Aditya Jayasimha, Youngho Han, Jiyeon Kim, Junmo Park, Kwangyoun Kim, Sooyeon Kim, Youngyoon Lee, Kyungbo Min, Chanwoo Kim

Samsung Research, Korea

Abstract

In this paper, we present our streaming on-device end-to-end speech recognition solution for a privacy sensitive voice-typing application which primarily involves typing user private details and passwords. We highlight challenges specific to voice-typing scenario in the Korean language and propose solutions to these problems within the framework of a streaming attention-based speech recognition system. Some important challenges in voice-typing are the choice of output units, coupling of multiple characters into longer byte-pair encoded units, lack of sufficient training data. Apart from customizing a high accuracy open domain streaming speech recognition model for voice-typing applications, we retain the performance of the model for open domain tasks without significant degradation. We also explore domain biasing using a shallow fusion with a weighted finite state transducer (WFST). We obtain approximately 13% relative word error rate (WER) improvement on our internal Korean voice-typing dataset without a WFST and about 30% additional WER improvement with a WFST fusion.

1. Introduction

Recent advances in the field of end-to-end neural network based speech recognition have brought the performance of a small foot-print low-latency on-device ASR almost on par with the larger server-side ASR systems [1, 2, 3, 4]. One of the key advantages of an on-device ASR is that it is privacy preserving, apart from saving huge server-side costs for the service provider. However, on-device speech recognition models still need significant engineering to make this work for privacy sensitive voice-typing. voice-typing can potentially serve as an efficient way to input especially for devices without a touch or physical keyboard such as smart television and wearable devices. The scenario of privacy sensitive voice-typing can be defined as one in which a user speaks his email ID, password, pin code, etc., character by character [5, 6]. While the transcription is typically restricted to English characters, numbers, special symbols and commands the spoken speech depends upon the native language or dialect of the speaker and the pronunciations in the corresponding language. Also, in many languages such as Korean the same output symbol can have multiple native as well as non-native spoken forms.

Recently, attention-based encoder-decoder models have gained popularity for developing end-to-end ASR systems [7, 8, 9, 10, 11, 12, 13]. These systems are capable of learning both the acoustic as well as linguistic information, unlike conventional systems which required separate modules for each of these tasks. Moreover, the end-to-end models are more suitable to be compressed since they do not need separate phonetic dictionaries and language models and hence making them one of the best candidates for on-device ASR system [2].

Attention-based models have been shown to perform better than other end-to-end models, namely, connectionist temporal classification (CTC) and recurrent neural network transducer (RNN-T) models [14]. With enough amount of data, they can even outperform the conventional models [7, 9]. However, most of these attention models use bidirectional long short-term memory (BLSTM) units and offline attention techniques which are not suitable for streaming decoding. Recently, there have been many efforts that make attention models streaming capable [15, 16, 17]. In [16], a hard monotonic attention based model was proposed for streaming decoding with acceptable accuracy degradation. A much improved soft monotonic chunk-wise attention (MoChA) based model was proposed in [17]. In this work, we propose MoChA based ASR system for the task of streaming on-device voice-typing. We aim at building a minimalistic ASR system that can run on-device under resource constrained environment of smart devices. While being under the umbrella of general ASR, voice-typing can be significantly different in terms of the challenges involved [6]. A voice-typing model primarily needs to run on-device for privacy issues, performance and needs to be small and efficient. Also, the average length of input utterance is much smaller for voice-typing, it has also been observed that online end-to-end model has delayed attention [2, 18] which can cause problems such as no output being generated for very short utterances and/or delayed output for others. Obtaining a large domain specific dataset is also a labor intensive task.

In this work, we address some of the above mentioned challenges. Due to unavailability of large scale domain specific data, we leverage the existing open domain dataset by training our baseline model on it. We then use data-augmentation techniques to adapt our model to the voice-typing domain. Although our main focus is to improve our performance for voice-typing, we want our system to perform well even for open domain usage. Finally, we also explore voice-typing specific post-processing techniques such as inverse text normalizer (ITN) and shallow fusion with a weighted finite state transducer (WFST). We achieve approximately 13% relative word error rate (WER) improvement on our internal Korean voice-typing test set without significant degradation in open domain WER.

2. Streaming attention encoder-decoder model for voice-typing

In this section, we define the model architecture and the training process for our baseline models similar to our previous work [2]. We also explore post processing techniques that we use to bias output for voice-typing.

2.1. Model architecture

We use an attention based encoder-decoder (AED) architecture which is composed of an encoder, a decoder, and an attention block. Our AED system is constructed using a Tensorflow 2.0 Keras model. This model is trained using an in-house trainer built using Tensorflow 2.0-Keras APIs and tf.data. The encoder converts the input sequence into a sequence of hidden vector representations called encoder embeddings. The decoder is a generative module and predicts the target labels which are byte-pair encodings(BPE) in this work. Attention is used to learn how heavily a particular hidden vector sequence influences the current target label prediction.
The encoder consists of several unidirectional LSTM (ULSTM) layers. It converts an input audio feature sequence \( \mathbf{x} = \{x_1, x_2, \ldots, x_T\} \) of length \( T \) into a sequence of hidden encoder embeddings \( \mathbf{h} = \{h_1, h_2, \ldots, h_T\} \), of length \( T \). Due to difference in input and output lengths, encoder-decoder models have difficulties in convergence. Hence, we use max-pooling layers having reduction factor two after the first three ULSTM layers.

The attention block computes a context vector \( \mathbf{c} \) based on the encoder embeddings \( \{h_1, h_2\} \) and the previous decoder state \( s_{t-1} \). Where \( l \) is the decoder output step. In this paper, we use MoChA for the attention block to ensure online or streaming decoding [17]. The decoder consists of one or more ULSTM layers followed by a few fully connected layers and a softmax layer. The decoder uses the attention context vector \( \mathbf{c} \), the previous state of the decoder \( s_{t-1} \), and the previous decoded label \( y_{t-1} \) to generate the label probabilities \( P(y_t|y_{1:t-1}, x_{1:i}) \) at the softmax output.

### 2.2. Training

For training, cross entropy loss is calculated between the final output of the decoder and ground truth labels \( y^* \):

\[
L_{CE} = - \sum_{t=1}^T \log(P(y^*_t|y^*_1:t-1, x_{1:i})),
\]

where \( x_{1:i} \) is used to denote streaming input. We additionally apply a feed forward layer and a softmax layer over encoder embeddings \( h \) and compute CTC loss between the decoder outputs and the ground truth labels. We then use joint CTC-CE loss as our final loss objective[20].

As our final recognition accuracy measure was word error rate (WER), we also define a loss on the candidate beams of the beam search which effectively minimizes the expected WER. We use the N-best variant for minimum word error rate(MWER) loss [21] defined as:

\[
L_{MWER} = \sum_{y^* \in \text{Beam}(x, N)} P(y^*|x)(W(y^*, y^*) - \bar{W})
\]

Where \( \text{Beam}(x, N) \) is set of top N hypothesis computed using beam search. \( P(y^*|x) \) is re-normalized distribution over top N hypothesis. \( W(y^*, y^*) \) represents number of word errors in hypothesis \( y^* \). \( \bar{W} \) is average number of word error calculated over top N hypothesis.

### 2.3. Compression and quantization

We use a singular value decomposition based low rank approximation algorithm (LRA) to compress our model. However, for large compression ratio it is difficult for the LRA algorithm to converge as the LRA distortion (the difference between LRA approximated weight matrix and the actual weight matrix) propagates through layers and increases the training loss. To deal with this issue we use the Hyper-LRA algorithm proposed in [22] which uses DeepTwist method to find the optimal LRA weights. To further reduce the memory footprint and increase the speed in order to meet real-time requirements we apply 8-bit quantization using TensorFlow-lite.

### 2.4. Domain specific post processing during inference

#### 2.4.1. Domain adaptation

To bias the output for voice-typing, we use on-the-fly rescoring approach [23]. Word-level voice-typing phrases are prepared in advance and are compiled into a language model WFST, \( G \). It is then composed with a lexicon FST, \( B \), which spells out BPE combinations for each word instead of the FST \( L \) used in conventional HCLG approach [24]. The domain LM is made after determination and minimization (\( P_{LM} = \min(\det(B \circ G)) \)).

The integration between the decoder model and the domain LM is accomplished with shallow fusion [25]:

\[
y = \arg \max_y P(y|y_{1:t-1}, x) + \lambda P_{LM}(y|y_{1:t-1})
\]

### 2.4.2. Inverse Text Normalization (ITN)

As the final step in our inference process, we use a rule-based ITN to convert our model output to user-displayed results. The rules consist of one-to-many mappings between numbers, alphabets, special characters to their corresponding texts in target language. For cases where we have multiple matches we select the candidate having the maximum string length. While building ITN for Korean we also resolve the ambiguity in Korean to English translation using output context (Eg:- \( \sigma(e) \) in Korean can refer to English alphabet \( e \) as well as Korean pronunciation for number 2. If output proceeding \( \sigma(e) \) is a number we choose 2 else we choose \( e \).

### 3. Challenges for voice-typing system

Apart from being privacy preserving, on-device ASR solution also reduces the network latency and improves the responsiveness. An on-device ASR solution needs to be both memory efficient and computationally inexpensive to be able to perform under the resource constraint environments of smart devices. Apart from the resource constraints, on-device voice-typing ASR also presents other distinctive challenges and choices. In this section, we discuss some such challenges and their proposed solutions.

#### 3.1. Choice of output units

The modeling unit for AED models has been shown to have a large impact on the overall performance [8, 26, 27, 28]. While conventional speech recognition systems used phoneme-based models, it was shown recently that word-piece models for AED models consistently outperform phoneme-based models [28]. Using phoneme or grapheme for voice-typing has the following advantages: (i) As the current output has little to no dependency on the previous output, there might be no potential benefit from the dependence captured by word-piece units. Hence, for voice-typing grapheme represents the output most closely. (ii) Lower vocabulary size would mean a lower memory footprint. However, these advantages disappear for open domain usage. Even with a low memory footprint, the total number of decoder runs required by grapheme based models is high leading to a higher overall time when compared to word-piece based models.

As we want our system to perform well on open domain as well, we choose to use word-piece as our modeling unit. Most word-piece based ASR models use a vocabulary size of 10K-15K [8, 10, 26, 9] for best performance. However, for voice-typing using a high vocab size makes the final memory footprint of the model bigger and has output coupling problem to be discussed in section 3.2.

#### 3.2. Output coupling and delayed attention

For a real-time ASR system, latency is also an important metric along with recognition accuracy. For online attention models, it has been observed that attention alignment is delayed as compared to full-attention models [2, 18] as they cannot use future information. While this does not affect the recognition accuracy of an ASR model, it affects the latency of the output.

In voice-typing, using a larger vocabulary size makes the latency problem more severe. As the output units now consist of coupled characters as well, the ASR system is waiting for the next character to be spoken in order to output a composite BPE unit having both characters together.
To solve the above problems of the choice of output units, output coupling and delayed attention, we propose to use a smaller BPE vocabulary consisting of about 2K BPE units. Also, to make sure that the effect of output coupling is not severe, we build our vocabulary from a text corpus consisting of 60% voice-typing data and 40% open domain data. Fig 1 shows an example utterance having speech signal as Korean pronunciation for abed. While, the model with 10K vocabulary runs for 3 decoder steps, and outputs cd coupled. Model with 2K vocabulary runs for 4 steps and outputs each character separately.

3.3. No or delayed attention
The delayed attention phenomenon discussed above causes a severe problem for short utterances having transcripts consisting of about 1-2 BPE units. For such utterances, on multiple occasions, we observe empty output from the MoChA model. In order to address this issue, we append a short segment of silence or dummy audio at the end of input audio if the length of input audio is smaller than 2 sec. This improves the system performance significantly on short utterances.

3.4. Model size
For our network architecture, the encoder is the major contributor to model size. While using a small cell size for ULSTM units can lead to smaller memory footprint and better decoding speeds, it has been observed [10] that using bigger cell size gives significantly better recognition accuracy for AED models. Hence, we decide to use a bigger cell size for our experiments. As mentioned in Section 2.2 we use compression and quantization techniques to compress our model without significant loss in recognition accuracy.

3.5. WFST biasing issues
For calculating domain LM probability, \( P_{LM}(y_i|y_{1:t-1}) \), we proceed a decoding step with all possible \( y_i \) from the vocabulary. Since we are using word level language model \( G \), lexicon \( B \), WFST paths for some \( y_i \) might be absent. For such \( y_i \), they would get pruned even though the E2E model gives a high probability for them. We introduce two techniques to prevent this: (i) Giving a low default LM weight to the BPE sequences which were not present in the WFST. (ii) Using all possible BPE expressions of a word when creating the lexicon FST \( B \). While the second technique provides significant improvements in recognition accuracy it also leads to exponential growth in the size of WFST for longer words. We explore splitting long phrases into shorter words to keep the FST size small.

4. Model adaptation and data augmentation
An on-device voice-typing system for smart device should be capable of running on a wide variety of user devices such as TV, smartwatch, AI speaker. Software wise these systems can be quite similar, but their hardware (processors, memory, mic, etc) varies significantly from device to device. Hence, apart from being compact and fast, an on-device voice-typing system should be trained on speech data from variety of environments. Recently, many large scale datasets have been proposed, fueling the development of end-to-end ASR systems [29, 30]. However, most of these datasets are open domain. While they can be used to train an open domain ASR, we must use data augmentation techniques to adapt our model for a limited domain. For this purpose, we use text-to-speech (TTS) generated utterances for voice-typing. We also collect a small dataset of about 6 hours consisting of voice-typing data. To up sample our recorded dataset and make our model robust, we use acoustic room simulator [31] and adapt our model obtained from Section 2 to the voice-typing domain.

4.1. TTS data augmentation
As an initial augmentation method, we use Deep Convolution TTS/DC-TTS [32]) to generate voice-typing data. We randomly generate transcriptions for voice-typing which contains a mixture of alphabet, numbers and email formatted sentences. In Korean, the pronunciations of numbers depend on them being ordinal or cardinal numbers, and we randomly sample from the two pronunciations. For the alphabet, we consider both capital and small letters. Regarding symbols, we convert them to their pronunciations in Korean. With all these variants, we generate about 41,563 speech utterances for each male and female professional voice actor.

4.2. Acoustic Simulator
To augment our dataset with simulated audio of different acoustic environments, we use the acoustic simulator as described in [31]. Using our acoustic simulator, we generate simulated speech utterances for various room dimensions, a wide distribution of reverberation time and signal-to-noise(SNR) ratio [31]. We also add a small bias to the reverberation time and SNR distributions for simulating near-field audio more frequently.

5. Experiments
We evaluated our baseline approaches for Korean and English. For training our baseline models we used anonymized open domain corpus consisting of around 10K hours of transcribed speech for Korean and English each [2]. Out of which we sampled one hour data for our test and validation sets for the open domain. We also add a small bias to the reverberation time and SNR conditions on short utterances for voice-typing. We also collect a small dataset of about 6 hours consisting of voice-typing data. To up sample our recorded dataset and make our model robust, we use acoustic room simulator [31] and adapt our model obtained from Section 2 to the voice-typing domain.

4.1. TTS data augmentation
As an initial augmentation method, we use Deep Convolution TTS/DC-TTS [32]) to generate voice-typing data. We randomly generate transcriptions for voice-typing which contains a mixture of alphabet, numbers and email formatted sentences. In Korean, the pronunciations of numbers depend on them being ordinal or cardinal numbers, and we randomly sample from the two pronunciations. For the alphabet, we consider both capital and small letters. Regarding symbols, we convert them to their pronunciations in Korean. With all these variants, we generate about 41,563 speech utterances for each male and female professional voice actor.

4.2. Acoustic Simulator
To augment our dataset with simulated audio of different acoustic environments, we use the acoustic simulator as described in [31]. Using our acoustic simulator, we generate simulated speech utterances for various room dimensions, a wide distribution of reverberation time and signal-to-noise(SNR) ratio [31]. We also add a small bias to the reverberation time and SNR distributions for simulating near-field audio more frequently.

5. Experiments
We evaluated our baseline approaches for Korean and English. For training our baseline models we used anonymized open domain corpus consisting of around 10K hours of transcribed speech for Korean and English each [2]. Out of which we sampled one hour data for our test and validation sets for the open domain. We also add a small bias to the reverberation time and SNR conditions on short utterances for voice-typing. We also collect a small dataset of about 6 hours consisting of voice-typing data. To up sample our recorded dataset and make our model robust, we use acoustic room simulator [31] and adapt our model obtained from Section 2 to the voice-typing domain.

4.1. TTS data augmentation
As an initial augmentation method, we use Deep Convolution TTS/DC-TTS [32]) to generate voice-typing data. We randomly generate transcriptions for voice-typing which contains a mixture of alphabet, numbers and email formatted sentences. In Korean, the pronunciations of numbers depend on them being ordinal or cardinal numbers, and we randomly sample from the two pronunciations. For the alphabet, we consider both capital and small letters. Regarding symbols, we convert them to their pronunciations in Korean. With all these variants, we generate about 41,563 speech utterances for each male and female professional voice actor.

4.2. Acoustic Simulator
To augment our dataset with simulated audio of different acoustic environments, we use the acoustic simulator as described in [31]. Using our acoustic simulator, we generate simulated speech utterances for various room dimensions, a wide distribution of reverberation time and signal-to-noise(SNR) ratio [31]. We also add a small bias to the reverberation time and SNR distributions for simulating near-field audio more frequently.

5. Experiments
We evaluated our baseline approaches for Korean and English. For training our baseline models we used anonymized open domain corpus consisting of around 10K hours of transcribed speech for Korean and English each [2]. Out of which we sampled one hour data for our test and validation sets for the open domain. We also add a small bias to the reverberation time and SNR conditions on short utterances for voice-typing. We also collect a small dataset of about 6 hours consisting of voice-typing data. To up sample our recorded dataset and make our model robust, we use acoustic room simulator [31] and adapt our model obtained from Section 2 to the voice-typing domain.
We present our streaming on-device end-to-end ASR solution for the task of voice-typing using MoChA attention. The proposed architecture is compact enough to fit on most smart devices and fast enough to do real-time online decoding. Using data augmentation we achieve about 13% improvement on the voice-typing dataset without significant loss in open-domain accuracy. Finally, we apply domain biasing with WFST and achieve an additional 30% improvement for voice-typing.

7. Conclusion

We evaluate our baseline model performances on both open-domain and voice-typing test set. All models shown in Table 1 use 10K vocabulary size, and were trained with only open domain corpus. Initial baselines were trained using the joint CE-CTC loss. After which SVD was applied to compress the model and MWER was jointly applied to compensate the performance loss. WFST experiments are all possible subword sequences of a word made using symbols from BPE and in the second variation WFST were considered two variations for sub-word sequence. In the first variation, WFST, we used the one-best BPE sequence, and in the second variation WFST, all we used all the possible subword sequences of a word made using symbols from BPE vocabulary. We compare domain biasing with the various combinations of WFST and G. We performed a linear search for most optimal fusion weights. As shown in 3 WFST B, all worked better than B, one for both WFST G, due to pruning phenomenon discussed in Section 3.5. We achieved about 30% improvement using WFST final all with fusion weight 0.45 and achieved 13.94% WER on Korean voice-typing dataset.

The average output latency of our final models was 252ms and 220ms for Korean and English respectively. The total memory (RAM, Code + Data) usage was 50MB and 70MB respectively for Korean and English. The real time factor (xRT) was 0.62 on Cortex A72 (1.75GHz) and 0.4 on Cortex-A75 (2.31GHz) for both the models.
8. References


