Improving Seq2Seq TTS Frontends with Transcribed Speech Audio

Improving Seq2Seq TTS Frontends with Transcribed Speech Audio

Siqi Sun, Korin Richmond, Hao Tang

***Abstract*—Due to the data inefficiency and low speech quality of grapheme-based end-to-end text-to-speech (TTS), having a separate high-performance TTS linguistic frontend is still commonly regarded as necessary. However, a TTS frontend is itself difficult to build and maintain, since it requires abundant linguistic knowledge for its construction. In this paper, we start by bootstrapping an integrated sequence-to-sequence (Seq2Seq) TTS frontend using a pre-existing pipeline-based frontend and large amounts of unlabelled normalized text, achieving promising memorization and generalisation abilities. To overcome the performance limitation imposed by the pipeline-based frontend, this work proposes a Forced Alignment (FA) method to decode the pronunciations from transcribed speech audio and then use them to update the Seq2Seq frontend. Our experiments demonstrate the effectiveness of our proposed FA method, which can significantly improve the word token accuracy from 52.6% to 91.2% for out-of-dictionary words. In addition, it can also correct the pronunciation of homographs from transcribed speech audio and potentially improve the homograph disambiguation performance of the Seq2Seq frontend.**

***Index Terms*—Text-to-Speech synthesis, sequence-to-sequence model, linguistic frontend, pronunciation learning, grapheme-to-phoneme.**

I. Introduction

BUILDING a conventional text-to-speech (TTS) synthesis system requires significant expert manual effort. Recently though, machine learning has promised increasingly to reduce or remove requirements for expert knowledge and input. With the emergence of sequence-to-sequence (Seq2Seq) neural networks in the past few years, it has been proposed a Seq2Seq network could directly convert input text to frequency-domain speech audio (e.g. acoustic features like Mel-spectrogram) in an end-to-end (E2E) manner, only requiring a large training corpus of ⟨text, speech audio⟩ [1]. This is known as *grapheme-based E2E TTS*, where conventional pronunciation modelling is embodied within the network implicitly [2].

Appealing as it might be, grapheme-based E2E TTS can prove highly data-inefficient. Many languages (e.g. English) have complicated letter-to-sound mappings, meaning that the pronunciation for many words cannot be reliably predicted, but must instead just be known. This requires a TTS system to see as many words as possible during training. For natural speech audio, word frequencies follow a Zipf distribution, so that the number of new words added for each additional hour of training data tends to flatten to a very long low tail [3]. Thus, in order to achieve good coverage of words, the training corpus needs to include thousands of hours of speech audio, which is two orders of magnitude larger than current corpus sizes typically. For ideographic languages (e.g. Chinese and Japanese), the problem is even worse, since there are no direct letter-to-sound mappings for these languages [4].

To avoid the data inefficiency of grapheme-based E2E TTS, most researchers have stepped back to a conventional two-stage architecture, which is composed of a linguistic frontend and an acoustic backend. The frontend converts input text to the internal linguistic pronunciation sequence (e.g., a string of phones, lexical stresses, syllable boundaries and prosodic boundaries), which the backend consumes to output a sequence of acoustic features. Usually, the backend is modelled by an integrated neural network [5]–[7], whereas the frontend is modelled by a pipeline of several modules [8], [9]. With a separate frontend, the aforementioned problems can be greatly relieved. The frontend can be constructed from much larger text corpora to ensure sufficient word coverage, since text is much cheaper and more accessible than speech audio. Several studies have shown having a separate linguistic frontend results in better speech quality and naturalness [10]–[12].

Moving from E2E TTS to this two-stage configuration inevitably complicates TTS system building. In particular, the pipeline structure in the first stage is notoriously difficult to build and maintain, as several labelled datasets are needed for constructing various modules and expert knowledge is required in the construction. Recently though, some work has demonstrated the feasibility of replacing the frontend pipeline with another Seq2Seq model [13]–[15], only requiring a large training corpus of ⟨text, linguistic pronunciations⟩. Moreover, all the parts within the Seq2Seq model can be optimized in an integrated way to mitigate the compounding error problem widely existing in the pipeline structure.

Having said that, it is still not easy to develop and update a Seq2Seq frontend, as corpora of ⟨text, linguistic pronunciations⟩ do not widely exist and require abundant linguistic knowledge to build. The initial development requires large amounts of linguistic annotation, not to mention the ongoing effort incurred by frequently updating and growing the frontend to increase its word coverage over time. In this work, our overriding aim is to make it easier to (i) initialise the Seq2Seq frontend building and (ii) improve and update the Seq2Seq frontend.

To facilitate the initial building, in this work, we bootstrap

S. Sun, K. Richmond, and H. Tang are with the University of Edinburgh, Edinburgh, UK (email: Siqi.Sun@ed.ac.uk; Korin.Richmond@ed.ac.uk; Hao.Tang@ed.ac.uk).

Manuscript received xxx 00, 2022; revised xxx 00, 2022.

the Seq2Seq frontend using mature pipeline-based frontends which already exist for many languages. Specifically, we distill the knowledge from a pre-existing pipeline-based frontend (the teacher model) to a Seq2Seq frontend (the student model). A large unlabeled text dataset and a pre-existing pipeline frontend are required for this process. Our experiment results show that the bootstrapping is effective and the bootstrapped Seq2Seq frontend can achieve impressive performance.

One main drawback of this bootstrapping approach is that the Seq2Seq frontend's performance is largely upper-bounded by the pipeline frontend, since within the approach the teacher's output is viewed as the ground truth for training the student model but the teacher will unavoidably have fixed dictionary size, grapheme-to-phoneme (G2P) conversion performance, homograph disambiguation performance, and so on. One way to overcome this limitation is to update the bootstrapped Seq2Seq frontend using other training sources. For instance, to overcome the limitation of fixed dictionary size, we can update the Seq2Seq frontend with text-pronunciation pairs containing out-of-dictionary words.

In updating the Seq2Seq frontend, it would be desirable if we could utilize transcribed speech audio (i.e., corpora of ⟨text, speech audio⟩) rather than pronunciation annotation, as transcribed speech audio is more accessible. Specifically, as the main contribution of this paper, we propose a Forced-Alignment (FA) method which utilizes transcribed speech audio to improve upon and regularly update the bootstrapped Seq2Seq frontend. The proposed method decodes from transcribed speech audio the pronunciation sequence and then updates the Seq2Seq frontend with it. Our experiment results show that the FA method is effective in decoding the pronunciation sequence (containing out-of-dictionary words), and so the updated Seq2Seq frontend has a larger "dictionary size" along with other improvements, such as better homograph disambiguation ability.

This paper is organized as follows. Section II provides the background of TTS frontend modelling and the detail of the bootstrapping approach. Section III provides a detailed formulation of updating the Seq2Seq frontend with transcribed speech audio. Section IV shows the experimental results of our methods. Section V concludes this paper.

II. From Pipeline-based Frontend to Seq2Seq Frontend

In this section, we provide detailed formulations of the conventional pipeline-based frontend and recent Seq2Seq frontend modelling. Then, we introduce the detail of the bootstrapping procedure, which is used to facilitate the initial building of Seq2Seq frontend.

A. Pipeline-based Frontend

Although several paradigms, including concatenative systems, statistical parametric speech synthesis (SPSS) systems and integrated neural networks, have seen use in the acoustic backend, the linguistic frontend is typically a relatively standard pipeline-based system. This pipeline is usually a concatenation of three stages, which are text normalization (TN), phonetic analysis and prosodic analysis, each stage including one or more modules, such as non-standard word (NSW) normalization, dictionary lookup, G2P conversion, part-of-speech (POS) tagging, prosodic structure prediction and so on[1]. In recent years, much effort has been put on replacing each component with a corresponding neural network [17]–[29]. Nevertheless, each component is still usually trained independently.

TABLE I: Input and output for each stage within the pipeline-based frontend and the Seq2Seq frontend. TN is not included in our Seq2Seq frontend in this work.

Frontend / stage	Input	Output
Pipeline-based frontend		
Text normalization	unnormalized text	normalized text
Phonetic analysis	normalized text	phonetic sequence
Prosodic analysis	phonetic sequence	pronunciation
Seq2Seq frontend	normalized text	pronunciation

TN is usually the first step in the pipeline. The main objective of TN is to normalize NSWs such as dates, numbers, and various other alphabetical and numerical expressions to spoken form word sequences [17]–[21]. NSWs can be normalized to different spoken forms depending on the surrounding context. The output of TN is then sent to the phonetic analysis stage, which converts the normalized text into a phonetic sequence. Each word is first looked up in a dictionary to find a corresponding phonetic pronunciation sequence. If a word cannot be found, its phonetic sequence will instead then be predicted by the G2P module, which is usually error prone. Much research has focused on G2P conversion to improve its accuracy [22]–[28]. Finally, the output of the phonetic analysis stage is input to the prosodic analysis stage to include prosodic information such as prosodic boundaries, phrasing and so on. The input and output for each stage are shown in Table I.

Most existing TTS systems involve a pipeline-based frontend similar to the above. The long-standing Festival [30] system, for example, developed primarily for speech synthesis research, has a frontend that is a combination of dictionary lookup, rule-based models and statistical models. We use Festival as a typical example of the pipeline-based frontend in our experiments in the following sections of this paper.

B. Seq2Seq Frontend

Recently, Seq2Seq models have emerged to become the state-of-the-art model for the acoustic backend [1], [31], [32]. This has also precipitated work on moving towards Seq2Seq frontends [13]–[15]. A Seq2Seq frontend aims to replace the whole processing pipeline with a monolithic Seq2Seq model, converting the input text $\boldsymbol{x}_{1:S} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_S]$ to the pronunciation sequence $\boldsymbol{y}_{1:T} = [\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_T]$ directly, where S and T are the lengths of the text sequence and the pronunciation sequence respectively.

[1]The details of the modules within each stage and their interconnections are beyond the scope of this paper. Interested readers may refer to Chapter 8 in [16] and Section 2.2 in [12].

In this work, the TN stage is not included in the Seq2Seq frontend modelling, as it has been shown that it is difficult to solve TN merely by having huge amounts of annotated training data and feeding it to a neural network [17], [33]. We leave incorporating TN into the Seq2Seq frontend for future work. As a result, the input to the Seq2Seq frontend $\boldsymbol{x}_{1:S}$ is normalized text, as shown in Table I.

A Seq2Seq model is based on an Encoder-Decoder architecture, often augmented with an attention mechanism. $\boldsymbol{x}_{1:S}$ is first encoded by the encoder to form a sequence of hidden representations $\boldsymbol{h}_{1:S} = [\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_S]$ of the same length

$$\boldsymbol{h}_{1:S} = \text{Encoder}(\boldsymbol{x}_{1:S}) \tag{1}$$

Then, for each decoding step $t \in [1, T]$, $\boldsymbol{h}_{1:S}$ is attended by the attention mechanism to generate a context vector $\boldsymbol{c}_t$.

$$\boldsymbol{\alpha}_{1:S} = \text{Attention}(\boldsymbol{h}_{1:S}, \boldsymbol{g}_{t-1}) \tag{2}$$

$$\boldsymbol{c}_t = \sum_{s=1}^{S} \alpha_s \boldsymbol{h}_s \tag{3}$$

where $\boldsymbol{\alpha}_{1:S} = [\alpha_1, \alpha_1, \ldots, \alpha_S]$ is the weight vector and $\boldsymbol{g}_{t-1}$ is the previous hidden state of decoder [34]. Together with the previous target $\boldsymbol{y}_{t-1}$ and $\boldsymbol{g}_{t-1}$, $\boldsymbol{c}_t$ is consumed by the decoder to model the conditional probability of the current target $\boldsymbol{y}_t$ and update $\boldsymbol{g}_t$.

$$\big(\Pr(\boldsymbol{y}_t|\boldsymbol{y}_{1:t-1}, \boldsymbol{x}_{1:S}), \boldsymbol{g}_t\big) = \text{Decoder}(\boldsymbol{c}_t, \boldsymbol{g}_{t-1}, \boldsymbol{y}_{t-1}) \tag{4}$$

The Seq2Seq model is trained using maximum likelihood estimation, and the ground-truth value of $\boldsymbol{y}_{1:t-1}$ is used to train the decoder in a teacher-forcing manner. During inference, the Seq2Seq model aims to decode the most likely linguistic pronunciation $\boldsymbol{y}^*_{1:T}$ by using the chain rule.

$$\boldsymbol{y}^*_{1:T} = \underset{\boldsymbol{y}_{1:T}}{\text{argmax}} \ [\log \Pr(\boldsymbol{y}_{1:T}|\boldsymbol{x}_{1:S})] \tag{5}$$

$$= \underset{\boldsymbol{y}_{1:T}}{\text{argmax}} \left[\sum_{t=1}^{T} \log \Pr(\boldsymbol{y}_t|\boldsymbol{y}_{1:t-1}, \boldsymbol{x}_{1:S})\right] \tag{6}$$

which can be approximated by beam search according to the conditional probability in an auto-regressive manner.

$$\boldsymbol{y}^*_t \sim \Pr(\boldsymbol{y}_t|\boldsymbol{y}_{1:t-1}, \boldsymbol{x}_{1:S}) \tag{7}$$

Note the normalized text $\boldsymbol{x}_{1:S}$ can have varying granularity, including characters, subword units (e.g. Byte Pair Encoding [35]) and words. In this work, we use characters as input, as character input can handle unseen words during inference and the number of embedding vectors is much smaller.

C. Bootstrapping the Seq2Seq Frontend

In order to ensure sufficient word coverage, a large corpus of ⟨normalized text, pronunciations⟩ is required in training a Seq2Seq frontend. Moreover, a large training corpus is also needed to prevent a neural network from overfitting. This kind of labelled data does not widely exist and abundant linguistic knowledge is required to create it.

One solution to the lack of readily available training data is to bootstrap the Seq2Seq frontend using a pre-existing pipeline-based frontend, in which unlabelled normalized text

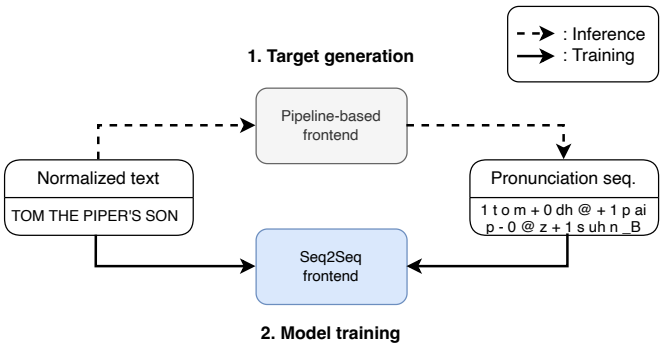


Fig. 1: Bootstrapping the Seq2Seq frontend using a pre-existing pipeline-based frontend and large amounts of unlabelled normalized text.

is first input into the pipeline-based frontend (the teacher model) to generate the pronunciation target, and then the pairs of normalized text and generated target serve as the training data for training the Seq2Seq frontend (the student model), as shown in Figure 1. In the bootstrapping procedure, a large amount of unlabelled normalized text and a pre-existing pipeline-based frontend are the only requirements. Fortunately, many languages already have mature pipeline-based frontends and unlabelled normalized text is also widely accessible.

Such a bootstrapping approach has already been successfully applied in the TTS research field to facilitate the transformation of a pipeline-based system to an integrated neural network. For instance, Conkie & Finch [13] run web-crawled sentences through a working production synthesizer to produce phone sequences for training multilingual neural TTS frontends. Sproat & Jaitly [17] and Zhang et al. [18] run web-crawled unnormalized text through Google's Kestrel TN component to produce normalized text for training neural TN models. Jia et al. [36] run web-crawled sentences through Kestrel to produce phone sequences for pre-training the encoder of Seq2Seq acoustic model.

III. Updating the Seq2Seq Frontend with Transcribed Speech Audio

In this section, we first consider the limitation of the above bootstrapping approach, which motivates the need for updating the bootstrapped Seq2Seq frontend. We argue it would be more desirable to use transcribed speech audio rather than pronunciation annotations when updating the Seq2Seq frontend, which in turn motivates our proposed FA method.

A. Limitation of the bootstrapping approach

There is one major limitation inherent in the above bootstrapping approach. Since the pipeline's output is given as the ground-truth target for training the Seq2Seq frontend, the pipeline's performance (including the dictionary size, G2P performance, homograph disambiguation performance and so on) becomes a likely upper bound for the Seq2Seq model's performance.

For *In-Dictionary* (ID) words, the pipeline must have a large expert-crafted dictionary in which to look up word

pronunciations (including the phones, lexical stresses and syllable boundaries). However, this dictionary is fixed in size and requires ongoing expert effort to keep it up to date. Thus, the considerable *memorization* ability of a Seq2Seq frontend is limited by this dictionary. In the pipeline, a statistical learning model might determine the pronunciation of homographs (a special category of ID words), which for example also limits the *generalisation* ability of the Seq2Seq frontend as far as homograph disambiguation is concerned.

For *Out-Of-Dictionary* (OOD) words, the pipeline heavily relies upon statistical learning models and rules, which may be of variable accuracy. Specifically, the G2P module in the pipeline can limit the ability of the Seq2Seq frontend for generalising phone prediction. Stress prediction and syllabification rules in the pipeline can limit the stress prediction generalisation and the syllabification generalisation of the Seq2Seq frontend respectively. Since acquiring inaccurate knowledge from OOD words would harm the generalisation ability of the Seq2Seq frontend and the same kind of knowledge (phone/stress/syllabification) can be acquired from ID words, we opt to avoid learning from OOD words entirely. We call this method *OOD-free bootstrapping*, which is presented in Section IV-A.

Consequently, the Seq2Seq frontend will mainly be limited by the dictionary size and the homograph disambiguation performance of the pipeline. These limitations are inherent in the bootstrapping approach itself and so are challenging to overcome. One straightforward way to overcome these limitations is to update the bootstrapped Seq2Seq frontend using other training sources. Specifically, to overcome the limitation of dictionary size, we can update the Seq2Seq frontend with text-pronunciation pairs containing OOD words. To overcome the limitation of homograph disambiguation, we can update the Seq2Seq frontend with text-pronunciation pairs containing homographs in specific contexts.

In the rest of this section, we propose a method to counterbalance the limitation of the bootstrapping method to some extent. In contrast to relying on corpora of ⟨normalized text, pronunciations⟩ which do not widely exist and are expensive to create, our proposed method makes efficient use of transcribed speech audio (i.e. corpora of ⟨normalized text, speech audio⟩), a resource which is much more readily accessible.

B. Learning Pronunciations from Other Training Sources

Previous studies have demonstrated the feasibility of learning pronunciations indirectly from other training sources, rather than directly from pronunciation annotations. One such approach is multilingual pronunciation modelling [37]–[40], which models the pronunciation of multiple languages in a shared neural network. Multilingual pronunciation modelling has been proven to benefit low-resource languages which have limited pronunciation annotations. Specifically, the pronunciations of the low-resource languages can be learned indirectly from the pronunciations of rich-resource languages, utilizing shared knowledge between different language pronunciation systems. This can be viewed as a multi-task learning method, where the pronunciation modelling of each language corresponds to a separate task.

Pronunciations can also be learned indirectly from other types of data. For instance, translation data has been used to improve diacritization for Arabic in a multi-task learning setting [41], exploiting the implicit linguistic and semantic knowledge involved in translation data which are helpful to pronunciation modelling. Similarly, transcribed speech audio has been used to improve G2P conversion in a multi-task learning setting [42], drawing from pronunciation information in speech audio.

Instead of using transcribed speech audio in the multi-task learning manner as in [42], this work directly decodes the pronunciation sequence from the transcribed speech audio and then updates the Seq2Seq frontend, which is conceptually more straightforward. Previous studies have demonstrated the feasibility of decoding word-level or phrase-level pronunciations (phones only) from transcribed speech audio [43]–[46], offering us an efficient way to use transcribed speech audio to improve the Seq2Seq frontend. In their method, a pre-trained phonetizer (e.g., a G2P model) is used as a candidate generator to generate for each target word a batch of pronunciation candidates which they then forced-align [43] to decode the closest match using the speech audio.

Our proposed FA method overlaps with these studies [43]–[46], but differs from them in several important ways. First, our work aims to model the whole pronunciation sequence of the frontend (a string of phones, lexical stresses, syllable boundaries and prosodic phrase boundaries), whereas the previous studies have focused on the pronunciation model (PM) of an automatic speech recognition (ASR) system, only modelling the phone sequence. Second, their work has focused on the word/phrase level, whereas our method is applicable at both the sentence level and the word/phrase level (by viewing each word/phrase as a separate sentence). There are multiple potential benefits to this, for example improved and implicit modelling of postlexical processes and homograph disambiguation. Finally, our aim is to improve upon an existing integrated Seq2Seq frontend for TTS, whereas they aim for a larger dictionary for ASR. Note that though this method is mainly proposed to update the bootstrapped Seq2Seq frontend, it is independent of the bootstrapping approach. It can also be used to improve/update any existing Seq2Seq frontend as well.

C. A General Formulation

In this subsection, we provide a general formulation of updating the Seq2Seq frontend with transcribed speech audio, and in the next subsection we provide one specific implementation of it.

An analogy to our formulation is how an external language model (LM) is integrated into an ASR system[2]. An external LM can greatly improve accuracy during decoding [48]–[50] as follows[3]

$$\boldsymbol{y}^*_{1:T} = \underset{\boldsymbol{y}_{1:T}}{\operatorname{argmax}} \; [\lambda \underbrace{\log \Pr(\boldsymbol{y}_{1:T})}_{\text{LM}} + \underbrace{\log \Pr(\boldsymbol{y}_{1:T}|\boldsymbol{z}_{1:F})}_{\text{AM}}] \tag{8}$$

[2]Although external LMs are not limited to ASR only (e.g., see neural machine translation [47]), we choose ASR as an illustration here for clarity.

[3]Here, for simplicity, we drop the length term which is usually required for length control in ASR decoding.

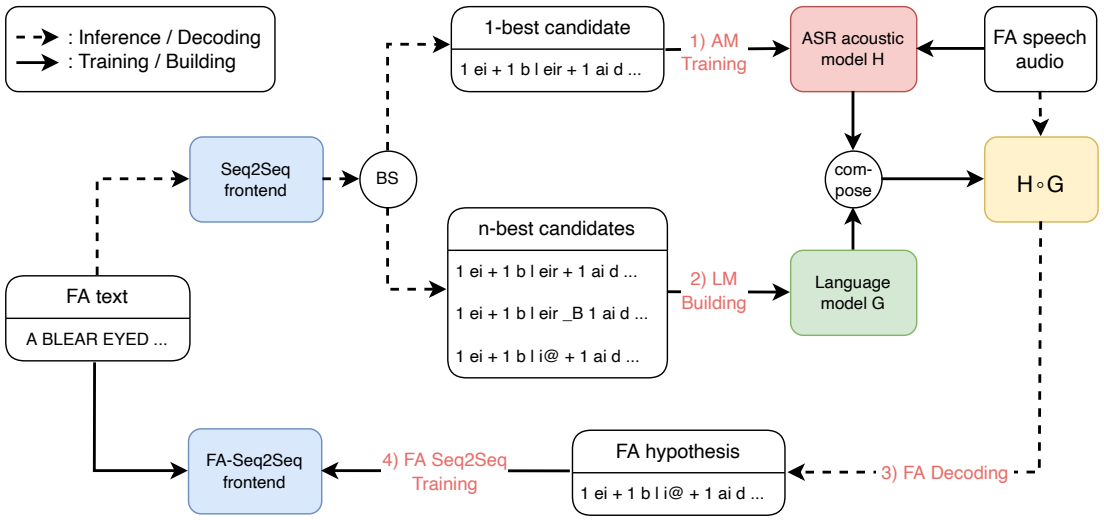


Fig. 2: The flow chart of our proposed FA method, where BS stands for Beam Search.

where $\boldsymbol{y}_{1:T} = [\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_T]$ is the phonetic sequence and $\boldsymbol{z}_{1:F} = [\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_F]$ is a sequence of acoustic features (e.g., MFCCs) derived from the corresponding speech audio, in which T is the length of phonetic sequence and F is the total number of acoustic frames. λ is a weighting factor balancing the importance between the LM, i.e., $\log \Pr(\boldsymbol{y}_{1:T})$, and the acoustic model (AM), i.e., $\log \Pr(\boldsymbol{y}_{1:T}|\boldsymbol{z}_{1:F})$.

The decoding of our baseline Seq2Seq frontend (Equation 5) can be viewed as a special variant of Equation 8, in which the AM is omitted and the LM is replaced by a conditional LM conditioned on the input text $\boldsymbol{x}_{1:S}$ (i.e., $\log \Pr(\boldsymbol{y}_{1:T}|\boldsymbol{x}_{1:S})$).

In order to use transcribed speech audio for decoding and then the decoded pronunciation for updating, we can extend Equation 5 to include an AM. For each text and audio pair $\langle \boldsymbol{x}_{1:S}, \boldsymbol{z}_{1:F} \rangle$, where $\boldsymbol{x}_{1:S}$ is the input text sequence and $\boldsymbol{z}_{1:F}$ is the corresponding acoustic feature sequence (e.g., MFCCs), we can instead decode the best pronunciation sequence $\boldsymbol{y}^*_{1:T}$ as follows

$$\boldsymbol{y}^*_{1:T} = \underset{\boldsymbol{y}_{1:T}}{\operatorname{argmax}} \left[\lambda \underbrace{\log \Pr(\boldsymbol{y}_{1:T}|\boldsymbol{x}_{1:S})}_{\text{conditional LM}} + \underbrace{\log \Pr(\boldsymbol{y}_{1:T}|\boldsymbol{z}_{1:F})}_{\text{AM}} \right] \quad (9)$$

Subsequently, the pairs of text and corresponding decoded pronunciation sequence $\langle \boldsymbol{x}_{1:S}, \boldsymbol{y}^*_{1:T} \rangle$ form a new dataset for frontend model update.

Note the difference between Equation 8 and Equation 9, where the LM in the former case is unrestricted, whereas the conditional LM in the latter case is highly restricted.

D. Forced Alignment Method

There are several ways to implement Equation 9. In this work, we choose an implementation based on Hidden Markov Models (HMM) and forced-alignment [43], because it is most straightforward to see its correspondence to Equation 9. For convenience, we call this implementation the *FA method*. We leave other implementations for future work.

In the FA method, Equation 9 is approximated by a candidate generation process followed by a decoding process. For each text and audio pair $\langle \boldsymbol{x}_{1:S}, \boldsymbol{z}_{1:F} \rangle$, a candidate generator generates a list of pronunciation candidates $\{\boldsymbol{y}_{1:T}\}_1^n$ for $\boldsymbol{x}_{1:S}$, sorted according to the conditional LM score, where n denotes the length of the candidate list. Then, the AM scores these candidates to produce the acoustic scores. Finally, two scores are added for each candidate and the candidate with the largest overall score is selected by the decoding process (termed *FA hypothesis*). In this work, the bootstrapped Seq2Seq frontend serves as the candidate generator. The transcribed speech audio dataset used throughout is called the *FA dataset*. Note that the accent variant of the bootstrapped Seq2Seq frontend should match that of the FA dataset speech audio.

Concretely, the flow chart of the FA method is shown in Figure 2, which consists of four main steps:

1) AM Training: The bootstrapped Seq2Seq frontend is first used to generate the 1-best candidate for the text of the FA dataset. Meanwhile, acoustic features (e.g., MFCCs) are extracted from the corresponding speech audio. Expectation-Maximization (EM) training is then carried out on the training pairs to obtain the AM H. All of this is done efficiently with HTK [51] in this work. Specifically, each vowel is modelled by 4 stress-vowel 5-state[4] HMMs (e.g., vowel 'a' is modelled by 'a0', 'a1', 'a2', 'a3' models), so that they are sensitive to different lexical stress patterns in the acoustics. Each consonant is modelled by one 5-state model. The syllable boundary is modelled by a skip model ('_'), so it is (largely) insensitive to acoustic variations. The word boundary is modelled by a 3-state tee model ('sp') with its emitting state tied to the centre state of the 'sil' silence model, and the phrase boundary is modelled by a 3-state tee model ('sp') followed by a 5-state model ('sil'), making them sensitive to acoustic variations too.

2) LM Building: The bootstrapped Seq2Seq frontend is also used as a candidate generator to generate the n-best candidates for each input sentence. For the Seq2Seq model, this can be done efficiently via beam search. Next, we use a weighted finite state transducer (WFST) [52], [53] to build a sentence-level LM G from these candidates, regarding the

[4]Three emitting states and 2 non-emitting states

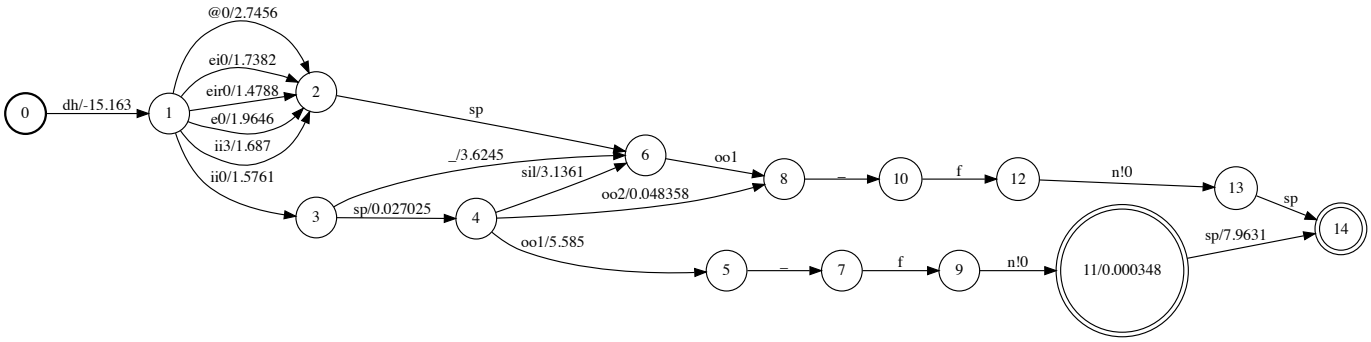


Fig. 3: An example WFST of G after determinization and minimization, constructed from 10-best candidates of "THE ORPHAN". The weights on the arcs are derived from the the beam search scores produced by the bootstrapped Seq2Seq frontend.

score produced by the beam search as the LM score (i.e. $\log \Pr(\boldsymbol{y}|\boldsymbol{x})$). Hence, LM G contains all the pronunciation information of the n-best candidates generated by the Seq2Seq frontend. LM building is done efficiently using OpenFST [53]. An example WFST of G constructed from 10-best candidates of the input text "THE ORPHAN" is shown in Figure 3.

3) FA Decoding: After composing H with G ($H \circ G$), we force-align [43] using the acoustic features to decode the closest matching candidate (i.e., the *FA hypothesis*). This is equivalent to re-ranking the n-best candidates based on the sum of the AM score (i.e. $\log \Pr(\boldsymbol{y}|\boldsymbol{z})$) and the LM score (i.e. $\log \Pr(\boldsymbol{y}|\boldsymbol{x})$). In order to incorporate λ, we multiply the LM score by λ before the composition. Since we treat stressed vowels, consonants and various boundaries as different HMMs in a systematic way, the decoding is done efficiently using the Viterbi decoding algorithm (`HVite` function in HTK [51]) and the decoded *FA hypothesis* is just a single string of symbols containing all levels of linguistic analysis (phones, lexical stresses, and the various boundaries).

4) FA Seq2Seq Training: After decoding, the paired text and FA hypothesis form a new dataset for subsequent model update. We can either combine this dataset with the original training set to train a new Seq2Seq frontend from scratch (as indicated in Figure 2), or use this new dataset to finetune the original Seq2Seq frontend (not shown in Figure 2 for clarity), the differences between which are explored in Section IV-B. For brevity, we call the resulting model the *FA-Seq2Seq* frontend.

IV. Experiments

A. Bootstrapping the Seq2Seq Frontend

In this subsection, we present the experimental results of bootstrapping the Seq2Seq frontend. We show the feasibility of bootstrapping the Seq2Seq frontend using a pipeline-based frontend and that the bootstrapped Seq2Seq frontend achieves promising results. At the same time, we also show the limitation of the approach, i.e., the pipeline's performance can limit the performance of the resulting Seq2Seq frontend.

1) Experimental Settings: LibriSpeech [54] is used in this experiment. It contains the speech recordings and the corresponding normalized transcriptions. We use the transcription parts of this corpus as our unlabelled normalized text. We merge all the transcriptions of three subsets of LibriSpeech (train-clean-100, train-clean-360 and train-other-500) to form the unlabelled dataset for bootstrapping (281,241 sentences in total). The transcriptions of Dev-clean (2,703 sentences) and Test-clean (2,620 sentences) of LibriSpeech are used to form the validation and test sets in our experiment respectively. To generalise our conclusion, we also use an out-of-domain dataset LJSpeech [55] (13,100 sentences) to form our second test set.

Festival [30] is used as our pipeline-based frontend. `unilex-rpx` (British English) is used as our dictionary and defines the phone set. Our Seq2Seq model is based on RNN models[5], in which the encoder is a 2-layer bidirectional LSTM and the decoder is a 2-layer unidirectional LSTM. The embedding vector size is set to 512 and the RNN hidden unit size is set to 512. Dropout rate is set to 0.3. We use the general attention mechanism [34] in our Seq2Seq model. The Adam optimizer [56] is used, where $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The initial learning rate is set to 0.001. After 50k steps, the learning rate decays by half every 10k steps. We implement our Seq2Seq model using OpenNMT[6] [57] and use character sequences as model input.

Before training the model, we pre-process Festival's output to remove symbol redundancies. For example, when inputting the normalized text "TOM THE PIPER'S SON", Festival generates a string of structured output:

```
<{(1 t o m )}{(0 dh @ )}{(1 p ai p )(0 @ z
)}{(1 s uh n )}> _B
```

in which the braces indicate word boundaries, the parentheses indicate syllable boundaries and the angle brackets indicate phrase boundaries. Numbers correspond to the lexical stresses, and "_B" indicates the phrase boundary type. The other alphabetical symbols are the phones. We preprocess its output as follows:

[5]We tried Transformer models in preliminary experiments, but the results were found to be inferior. We suspect Transformer models can prove more powerful and hence also require more data to overcome overfitting problems.

[6]https://github.com/OpenNMT/OpenNMT-py

TABLE II: Word-level results of the naive bootstrapping approach, where 'total' is the combination of 'seen' and 'unseen' for each group. Word ACC stands for word-level accuracy of word tokens considering the phones, stresses and syllable boundaries, while Word PACC stands for word-level accuracy of word tokens considering phones only.

(a) LibriSpeech Test-clean

Category		# word tokens	Word ACC (%)	Word PACC (%)	PER (%)	Stress ACC (%)	Syl ACC (%)
ID	seen	51,121	99.9	99.9	0.03	99.9	100
	unseen	115	65.2	85.2	4.8	70.4	92.2
	total	51,236	99.8	99.9	0.05	99.9	100
OOD	seen	291	90.7	95.9	0.75	92.1	98.6
	unseen	208	29.8	59.1	10.3	43.8	86.0
	total	499	65.3	80.6	4.9	71.9	93.4

(b) LJSpeech

Category		# word tokens	Word ACC (%)	Word PACC (%)	PER (%)	Stress ACC (%)	Syl ACC (%)
ID	seen	211,862	99.7	99.8	0.06	99.9	100
	unseen	1,543	56.9	80.6	4.3	64.5	94.8
	total	213,405	99.4	99.7	0.12	99.6	100
OOD	seen	698	79.6	89.0	2.3	83.8	94.4
	unseen	1,711	24.2	51.4	11.2	42.8	88.7
	total	2,409	40.3	62.3	8.5	54.7	90.4

```
1 t o m + 0 dh @ + 1 p ai p - 0 @ z + 1 s uh n _B
```

where "+" corresponds to a word boundary and "–" corresponds to a syllable boundary.

2) Naive Bootstrapping Approach: To start with, we bootstrap the Seq2Seq frontend directly, without considering the negative effect of OOD words. As mentioned in Section III-A, for OOD words, Festival relies on relatively simple statistical models and rules for phone/stress/syllabification prediction, which are not necessarily accurate. For instance, when `unilex-rpx` is used, Festival applies a default rule for stress prediction, predicting 0 stresses for all the syllables within an OOD word.

To illustrate the negative effect of learning from OOD words, we separate the word tokens[7] in the test sets into four categories: (1) *seen*[8] ID words, (2) *unseen* ID words, (3) *seen* OOD words and (4) *unseen* OOD words. The test results on Test-clean and LJSpeech are summarized in Table II.

Besides showing overall word accuracy which takes the correctness of phones, stresses and syllabification into account (Word ACC), we also show here the traditional word accuracy only considering the phones (Word PACC), Phone Error Rate (PER), lexical stress accuracy and syllabification accuracy.

For ID words, we can regard Festival's predictions as being correct in most cases. For both test sets, the word accuracy is high for *seen* ID words ($> 99\%$), since those words' pronunciations have been seen during training. Note that some of the mismatches are due to the presence of homographs. For *unseen* ID words, the word accuracy drops dramatically (65.2% & 56.9%). When inspecting the predictions, we observe that many of the mismatches are caused by the all-zero lexical stress issue. The Seq2Seq frontend has apparently generalised the default stress prediction rule of OOD words to unseen ID words, which is not desirable. This is also reflected in the low stress accuracy (70.4% & 64.5%).

For OOD words, we can regard Festival's predictions as being unreliable, since they always contain a null all-zero stress pattern. For *seen* OOD words, the word accuracy is also high, as they have been seen during training. Since this word-level accuracy is equivalent to the stress defect rate, we want this value to be as low as possible. For *unseen* OOD words, the word accuracy is not too high ($\sim 25\%$). When inspecting the predictions, we observe that the Seq2Seq frontend successfully generalises the correct stress pattern learned from ID words to some unseen OOD words, which confirms our hypothesis. However, it also generalises the defective stress pattern to some other unseen OOD words. We also want this value to be kept as low as possible.

3) OOD-free Bootstrapping Approach: One straightforward strategy of OOD-free Bootstrapping is to exclude sentences with OOD words from the unlabelled dataset before bootstrapping. We evaluate this strategy here. After excluding those sentences, the total number of sentences drops from 281,241 to 206,056 (73.3%). The total number of unique word types drops from 89,114 to 51,429 (57.7%). We perform the bootstrapping as before, with the same hyper-parameters and training schedule. The test results are summarized in Table III.

For both test sets, the word accuracy largely remain unchanged for *seen* ID words ($> 99\%$). However, the word accuracy of the *unseen* ID words indicate that the Seq2Seq model now generalises much better to *unseen* ID words than before (81.9% vs. 65.2% and 65.3% vs. 56.9%). This is also reflected by the improvements in stress accuracy and the syllabification accuracy and the reduction in PER for both test sets respectively. This confirms the benefits of not learning from bad teachers, i.e., OOD words.

For OOD words, the stress accuracy equals or becomes close to 0. When inspecting the predictions, we do not observe the all-zero stress pattern anymore. Note that all OOD words are excluded from the training set, and hence the total number of seen OOD words is 0.

To further generalise our conclusions, we conduct the OOD-free approach on a third test set, which is one order of magnitude larger in size than LJSpeech, shown in the column labelled 'Baseline Seq2Seq' of Table IV. The ID (seen) result (word token ACC of 99.94%) shows that the Seq2Seq frontend is good at **memorizing** the words it has seen. Note that word

[7]Following standard terminology in the natural language processing community, a 'word token' is an individual occurrence of a distinct 'word type' in the text.

[8]*seen* and *unseen* designate whether the word and its pronunciation have appeared in the paired training set.

TABLE III: Word-level results of OOD-free bootstrapping approach, where 'total' is the combination of 'seen' and 'unseen' for each group. Word ACC stands for word-level accuracy of word tokens considering the phones, stresses and syllable boundaries, while Word PACC stands for word-level accuracy of word tokens considering phones only.

(a) LibriSpeech Test-clean

Category		# word tokens	Word ACC (%)	Word PACC (%)	PER (%)	Stress ACC (%)	Syl ACC (%)
ID	seen	51,070	99.9	99.9	0.03	100	100
	unseen	166	81.9	89.2	2.9	91.6	96.4
	total	51,236	99.8	99.9	0.05	99.9	100
OOD	seen	0	–	–	–	–	–
	unseen	498	0	42.2	14.8	0	78.1
	total	498	0	42.2	14.8	0	78.1

(b) LJSpeech

Category		# word tokens	Word ACC (%)	Word PACC (%)	PER (%)	Stress ACC (%)	Syl ACC (%)
ID	seen	211,699	99.8	99.8	0.06	99.9	100
	unseen	1,874	65.3	80.5	4.9	81.8	97.7
	total	213,573	99.5	99.7	0.14	99.7	100
OOD	seen	0	–	–	–	–	–
	unseen	2,425	1.0	41.8	15.4	1.1	81.6
	total	2,425	1.0	41.8	15.4	1.1	81.6

type PACC (99.4%) is close to that reported elsewhere in the literature (99.0% reported in [27]), though they use a different test set.

The ID (unseen) result (word token ACC of 82.1%) shows that the resulting Seq2Seq frontend also performs well on **generalising** to unseen words. Note that word type PACC (88.3%) outperforms those reported in recent literature (e.g. 70.2% (NetTalk) & 77.9% (CMUDict) in [26] and 66.2% in [27]), though they use different test sets.

B. Forced Alignment Method

In this subsection, we show the experimental results of our proposed FA method. We show that the FA method is especially effective in decoding the pronunciation sequence (and hence the pronunciation of OOD words within it) and updating the bootstrapped Seq2Seq frontend (OOD-free bootstrapping approach). Moreover, the resulting FA-Seq2Seq frontend is more robust than the bootstrapped Seq2Seq frontend.

1) Experimental Settings: The Seq2Seq frontend from Section IV-A3 serves as the candidate generator. Hi-Fi TTS [58] is used to construct the *FA dataset* and the test set in this experiment. Hi-Fi TTS is a high-quality multi-speaker English dataset, which consists of speech audio from 11 speakers with different accents. As mentioned in Section III-D, the accent of the speech audio should match that of the bootstrapped Seq2Seq frontend. Since our Seq2Seq frontend is based on the British accent (RPX), we choose the RPX speakers and combine their data to form the *FA dataset*. As a result, 3 British-accent (RPX) speakers (numbered 92, 6097 and 9136) are chosen and the resulting *FA dataset* consists of 102,749 utterances in total ($\sim$ 81.7 hours).

The other 7 American-accent (GAM) speakers' text data in Hi-Fi TTS are merged to form the test set in this experiment, amounting to 219,669 sentences in total. As usual, for ID words, we can regard Festival's predictions as (reasonably reliable) ground truth. For OOD words, we ask a native linguist to correct the pronunciation predicted by Festival, so we also have the ground truth for OOD words this time.

2) Hyper-parameters: There are several important hyper-parameters involved in our FA method. The first one is how to obtain a FA-Seq2Seq frontend. There are three options: 1) Train a brand new Seq2Seq model with all the available data (original dataset plus decoded *FA hypotheses*, 308,805 sentences in total) from scratch; 2) train a new Seq2Seq model only with decoded *FA hypotheses* (102,749 sentences); and 3) finetune the bootstrapped Seq2Seq model with decoded *FA hypotheses* (102,749 sentences). We compare their performance on Dev-clean using their best checkpoints respectively and find that option 1 performs best overall, 3 is slightly worse than 1, and 2 performs much worse compared to options 3 and 1. This is most likely due to the lack of the original bootstrapping dataset in option 2, which confirms the importance of bootstrapping in the first place. In the following sections, we will stick to option 1 unless otherwise stated.

The second hyper-parameter is n, i.e., the length of the candidate list. At first glance, we might expect that the larger n is the better, as it becomes more probable that the correct pronunciation sequence is within the n-best list. However, larger n also means that the LM G will be composed of more candidates of minor differences, making the FA decoding more error-prone. Hence, there is a trade-off between the comprehensiveness and the redundancy of the n-best list. We compare $n = 3, 5, 10, 30$ on Dev-clean and find that when $n = 10$, the best validation performance is achieved in our setting.

The final hyper-parameter is the LM scaling factor λ. As shown in Equation 9, λ balances the LM and the AM during decoding. A larger λ means the LM (i.e., the bootstrapped Seq2Seq frontend) dominates the decoding, whereas a smaller λ means the AM dominates. Intuitively, we should set a larger λ if the LM is more accurate and a smaller λ if the AM is more accurate. We evaluate different λs on a separate validation set, and the results of this are shown in Figure 4. The best validation performance is achieved in our setting when $\lambda = 50$.

3) Experimental Results: The results on the Hi-Fi TTS test set are shown in Table IV. The fourth column shows the test results of the baseline Seq2Seq frontend and the last column shows the test results of the updated FA-Seq2Seq frontend. The results are grouped into four groups.

The first group corresponds to sentence-level results. The measure "# alignment error" indicates the number of sentences with an alignment error (e.g., word skip or word duplication, which are two common alignment error types for Seq2Seq

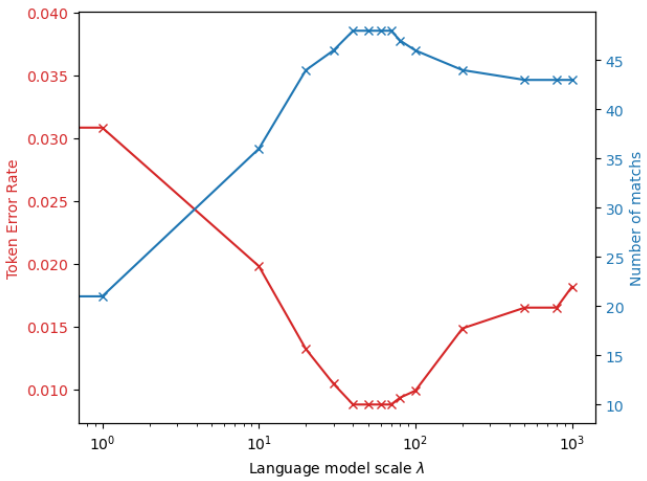


Fig. 4: Token Error Rate and utterance ACC vs. LM scale λ on a golden validation set (best viewed in color)

TABLE IV: Test results on Hi-Fi TTS test set for the baseline Seq2Seq frontend (OOD-free bootstrapping approach, 120k checkpoint) and the FA-Seq2Seq frontend (150k checkpoint). The numbers in parentheses indicate how many word tokens or word types there are for each category. PBER stands for phrase boundary error rate. ACC stands for word-level accuracy considering phones, stresses and syllabification, whereas PACC stands for word-level accuracy considering phones only. PER stands for phone error rate.

			Baseline Seq2Seq	FA-Seq2Seq
Sentence (219K)	# alignment error		138	35
	Sum of length difference		937	60
	PBER (%)		2.95	2.94
ID (seen)	Token (2.2M)	ACC (%)	99.94	99.94
		PER (%)	0.018	0.017
	Type (29k)	ACC (%)	99.1	99.2
		PACC (%)	99.4	99.5
ID (unseen)	Token (2.9k)	ACC (%)	82.1	82.2
		PER (%)	3.26	3.15
	Type (1.6k)	ACC (%)	83.0	83.4
		PACC (%)	88.3	88.8
OOD	Token (982)	ACC (%)	52.6	91.2
		PER (%)	15.7	1.7
	Type (198)	ACC (%)	83.3	87.9
		PACC (%)	86.4	91.4

models). As we can see, the resulting FA-Seq2Seq is much more robust than the baseline (138 vs. 35). Sum of length difference is the sum of differences between the ground truth and the prediction sequence lengths, which correlates well with "# alignment error". The phrase boundary error rates (PBER) are similar for both frontends. Note that here we evaluate the phrase boundaries against Festival's predictions, which may be rather inaccurate, as Festival partly relies on the punctuation to determine the phrase boundaries and we remove all punctuation but apostrophes when preprocessing the input text. We leave formal evaluation of phrase boundaries against ground truth for future work.

The second group corresponds to the *seen* ID words (2.2M word tokens or equivalently 29k word types). Measures based on word tokens and word types are shown here respectively. We can see the two frontend models achieve similar performance under all four metrics.

The third group corresponds to the *unseen* ID words (2,923 word tokens or equivalently 1.6k word types). The same four metrics are shown here. Unsurprisingly, FA-Seq2Seq generalises slightly better than the baseline does according to all four metrics, due to the slightly larger training set.

The fourth group corresponds to the OOD words (982 word tokens or equivalently 198 word types), which is of the most interest to us. Note that these words used to be unseen but have now become seen because of the newly formed FA decoded dataset. The same four metrics are shown here. FA-Seq2Seq frontend surpasses the baseline by a large margin (91.2% vs. 52.6% in word token ACC), indicating that it has successfully decoded and then learned the pronunciation of OOD words from transcribed speech audio and it now has a larger "dictionary size" than the baseline does.

C. Error Analysis

To better understand the performance and behaviour of the Seq2Seq frontend (including both the baseline and FA-Seq2Seq), we manually analyze the errors made by FA-Seq2Seq in this subsection. Specifically, we focus on the prediction error for *unseen* ID word types on Hi-Fi TTS test set (i.e., the third group in Table IV), as Festival's predictions of these word types are mostly reliable and so they can be regarded as the ground truth. For FA-Seq2Seq, there are 265 prediction errors in total, which we divide into two error supertypes: phone error and non-phone error. Phone errors are those containing phone mismatches and are further divided into five error types:

- Vowel→schwa: exactly one vowel other than schwa (plus its lexical stress) in the ground truth is substituted by a schwa (plus its lexical stress).
- Schwa→vowel: exactly one schwa (plus its lexical stress) in the ground truth is substituted by a vowel other than schwa (plus its lexical stress).
- Vowel: exactly one vowel other than schwa (plus its lexical stress) in the ground truth is substituted by another vowel other than schwa (plus its lexical stress).
- Consonant: exactly one consonant in the ground truth is substituted by another consonant.
- Unrecoverable: other errors not belonging to any of the above four types (e.g., two vowel substitutions in one word), which can be largely regarded as unrecoverable phone errors.

TABLE V: The prediction errors made by FA-Seq2Seq for *unseen* ID word types on Hi-Fi TTS test set (265 in total) and some illustrative examples. Festival's predictions are mostly reliable and so they can be regarded as the ground truth.

Error type	Word	Festival	FA-Seq2Seq	Proportion
Phone error (i.e., errors containing phone mismatches)				
Vowel→schwa	ENDICOTT	1 e n - 0 d i - 0 k o t	1 e n - 0 d i - 0 k @ t	7.5%
	CONVERSES	1 k o n - 0 v @@r s - 0 i z	0 k @ n - 1 v @@r s - 0 i z	
Schwa→vowel	HORRENDOUS	0 h @ - 1 r e n - 0 d @ s	0 h o - 1 r e n - 0 d @ s	5.7%
	ALLYING	0 @ - 1 l ai - 0 i ng	1 a - 0 l ai - 0 i ng	
Vowel	LATHERING	1 l aa - 0 dh @ r - 0 i ng	1 l a - 0 dh @ r - 0 i ng	18.1%
	REPUBLICATION	2 r ii - 0 p uh - 0 b l i k - 1 ei - 0 sh n!	0 r i - 2 p uh - 0 b l i k - 1 ei - 0 sh n!	
Consonant	ROTHENBURG	1 r o - 0 th @ n - 0 b @@r g	1 r o - 0 dh @ n - 0 b @@r g	2.3%
	TRANSFUSE	0 t r a n s - 1 f y uu z	0 t r a n s - 1 f y uu s	
Unrecoverable	INITIATES	0 i - 1 n i - 0 sh iy @ t s	0 i - 1 n i - 0 sh iy - 0 ei t s	34.0%
	ADIRONDACKS	2 a - 0 d i - 1 r o n - 0 d a k s	0 @ - 1 d ai @ n - 0 d a k s	
	GOURMET	1 g ur - 0 m ei	1 g ur - 0 m i t	
Non-phone error (i.e., errors in which the phone subsequence match the ground truth)				
Stress	GERMANTOWN	1 jh @@r - 0 m @ n - 2 t ow n	2 jh @@r - 0 m @ n - 1 t ow n	11.7%
	INTAKING	1 i n - 2 t ei k - 0 i ng	0 i n - 1 t ei k - 0 i ng	
Syllabification	IDEOLOGICALLY	2 ai - 0 d iy @ - 1 l o - 0 jh i - 0 k l iy	2 ai - 0 d iy @ - 1 l o - 0 jh i k - 0 l iy	20.8%
	BOTANIC	0 b @ - 1 t a - 0 n i k	0 b @ - 1 t a n - 0 i k	

Non-phone errors are those errors in which the phone subsequences match and can be further divided into two error types:

- Stress: at least one lexical stress does not match.
- Syllabification: at least one syllable boundary's position does not match.

All the aforementioned error types and some illustrative examples are shown in Table V. We manually checked all the prediction errors and found despite the mispronunciation, these predictions are plausible according to the English letter-to-sound rules (e.g., "TRANSFUSE" is plausibly mispronounced as "0 t r a n s - 1 f y uu s"). This again confirms that Seq2Seq frontends are good at generalising to unseen words. During our manual inspection, we found only "Unrecoverable" errors are severe errors, which may result in the listener not being able to recover the original word from its prediction, accounting for around $1/3$ of the total errors. Also note that a prediction error does not mean it is indeed incorrect, as it might be an alternative pronunciation which is not present in Festival's dictionary (e.g., the prediction of "1 l a - 0 dh @ r - 0 i ng" for LATHERING is an acceptable pronunciation).

D. Discussion and Future Work

In this subsection, we consider the strength and the weakness of the proposed FA method using some illustrative examples, as well as discussing the scaling of the Seq2Seq frontend.

1) Decoding the Pronunciation of OOD Words: To investigate the effectiveness of our FA method in decoding the pronunciation from transcribed speech audio, we first focus on OOD words. Three positive examples are shown in Table VI. It is possible that there are several training sentences in the *FA dataset* containing the target word. Here, for each example word, we only show one such training sentence and its 4 most probable candidates according to the LM scores ($\lambda \log \Pr(\boldsymbol{y}|\boldsymbol{x})$, shown in the second to last column). The baseline Seq2Seq model will select the candidate with the largest (scaled) LM score as the prediction. Instead, the FA method will select the candidate with the largest sum of the scaled LM score and the AM score ($\log \Pr(\boldsymbol{y}|\boldsymbol{z})$, shown in the last column), highlighted in bold in Table VI.

2) Improving Homograph Disambiguation: In addition to correctly decoding the pronunciation of OOD words, we observe the FA method also has the potential to improve homograph disambiguation using transcribed speech audio. The last example "READ" in Table VI is a homograph which is a *seen* ID word. From this example, we see that *seen* ID homographs can potentially benefit from our method as well. To confirm this, we analyze all the newly introduced "READ"s in the FA dataset, which is summarized in Table VII. Over all the newly introduced 179 "READ"s, the 1-best candidate accuracy is 69.8%. The percentage of the n-best list ($n = 10$) containing the correct pronunciation is 100%. For the negative examples, the FA method corrects them most of the time (85.5%). For the positive examples, the FA method always leaves the correct pronunciation untouched (error rate of 0).

Furthermore, we analyze the impact of adding these FA-corrected pronunciations into the training set of FA-Seq2Seq. The baseline Seq2Seq contains 1,719 tokens of "READ" in its training set, while FA-Seq2Seq contains 1,897 after the augmentation. We manually check all differences between the baseline Seq2Seq prediction and the FA-Seq2Seq prediction

TABLE VI: Some positive examples of the proposed FA method (decoded with a 10-best candidate list and $\lambda = 50$ in Equation 9). The first three examples are OOD words while the last example is an ID seen homograph. Only one training sentence containing this word and its first 4 candidates (with truncation) are shown here to save space. The closest matching candidate with regard to Equation 9 (i.e. FA hypothesis) is highlighted in bold.

Word	Baseline (incorrect)	FA-Seq2Seq (correct)	Context & Hypotheses	$\lambda\times$LM score	AM score
BLEAR	1 b l eir	1 b l i@	...BY A BLEAR EYED OLD WOMAN		
			1 b ai + 1 ei + 1 b l eir + 1 ai d + 1 ou lw d + 1 w u - 0 m @ n _B	−21.9	−60076
			1 b ai + 1 ei + 1 b l eir _B 1 ai d + 1 ou lw d + 1 w u - 0 m @ n _B	−78.8	−60130
			1 b ai + 1 ei + 1 b l i@ + 1 ai d + 1 ou lw d + 1 w u - 0 m @ n _B	−111.1	−59447
			1 b ai + 1 ei + 1 b l i@ _B 1 ai d + 1 ou lw d + 1 w u - 0 m @ n _B	−164.4	−59483
INVAL -IDED	2 i n - 0 v @ - 1 l ai d - 0 i d	0 i n - 1 v a - 0 l i d - 0 i d	...THEY HAD LEFT INVALIDED		
			1 dh ei + 1 h a d + 1 l e f t + 1 i n - 0 v @ - 2 l ai d - 0 i d _B	−35.0	−99629
			1 dh ei + 1 h a d + 1 l e f t + 2 i n - 0 v @ - 1 l ai d - 0 i d _B	−51.3	−99576
			1 dh ei + 1 h a d + 1 l e f t + 0 i n - 1 v a - 0 l i d - 0 i d _B	−104.5	−99521
			1 dh ei + 1 h a d + 1 l e f t + 1 i n - 0 v @ - 0 l ai d - 0 i d _B	−272.3	−99612
QUAD -ROON	0 k w @ - 1 d r uu n	0 k w o - 1 d r uu n	...BEAUTIFUL QUADROON GIRL		
			1 b y uu - 0 t i - 0 f u lw + 0 k w @ - 1 d r uu n + 1 g @@r lw _B	−34.6	−79619
			1 b y uu - 0 t i - 0 f u lw + 0 k w o - 1 d r uu n + 1 g @@r lw _B	−60.7	−79460
			1 b y uu - 0 t i - 0 f u lw + 1 k w o - 0 d r uu n + 1 g @@r lw _B	−87.9	−79578
			1 b y uu - 0 t i - 0 f u lw + 0 k w @ - 1 d ur n + 1 g @@r lw _B	−233.1	−79888
READ	1 r ii d	1 r e d	...SEEMED TO HAVE HEARD OR READ ...		
			1 s ii m d + 0 t uu + 1 h a v + 1 h @@r d _B 1 oo + 1 r ii d	−76.4	−224833
			1 s ii m d + 0 t uu + 1 h a v + 1 h @@r d _B 1 oo + 1 r ii d	−91.7	−224809
			1 s ii m d + 0 t uu + 1 h a v + 1 h @@r d _B 1 oo + 1 r ii d	−94.5	−224796
			1 s ii m d + 0 t uu + 1 h a v + 1 h @@r d _B 1 oo + 1 r e d	−96.7	−224346

TABLE VII: Newly introduced homograph "READ" in the FA dataset. See Section IV-D for a detailed analysis. (decoded with a 10-best candidate list)

# word tokens	1-best ACC (%)	Within n-best list (%)	FA correction ACC (%)	FA mis-correction error rate (%)
179	69.8	100	85.5	0

for "READ" tokens in the test set. Among all the 728 "READ" tokens in the test set, the two models agree in the prediction 658 times and disagree in the prediction 70 times. Among all the 70 differences, the baseline Seq2Seq is correct 33 times (47.1%), while FA-Seq2Seq is correct 37 times (52.9%). We suspect the relatively small improvement is due to the relatively small number of newly introduced "READ" tokens (179 vs. 1,719). We leave formal evaluation of the improved homograph disambiguation performance for future work.

3) Scaling up the Seq2Seq frontend: Currently, the total number of unique ID word types in the unlabeled training set ($\sim$51k) is only a small fraction of the dictionary size ($\sim$116k[9]). For better initial memorization and generalisation in practice, we can opt for a larger unlabelled dataset which at least covers all the ID word types. To further improve the initial homograph disambiguation generalisation ability, we shall find or collect an *FA dataset* which focuses on homographs. To enlarge the initial "dictionary size", we shall find or collect an *FA dataset* containing the OOD words of interest to us. We leave the exploration of significantly scaling up for future work, as here we aim only to present our basic approach.

4) Limitations of the FA method: Despite the success of our proposed FA method, we outline that it has some limitations which could result in the method failing to decode the pronunciation from transcribed speech audio. First, the proposed FA method assumes the candidate generator is diverse enough, so that the correct pronunciation of the sentence is included within the n-best list. This is true for many OOD words and most *seen* ID homographs. However, sometimes this assumption may not be met. Most negative examples of OOD words are indeed cases where this assumption has failed. Often, increasing n cannot solve this issue. For instance, the correct pronunciation "1 sh o m - 0 b r @" of the loanword "CHAMBRE" is so uncommon according to English letter-to-sound rules that it is not within the n-best candidate list. Finding an effective solution to this problem will be the subject of our future work. Second, as mentioned in Section III-D, our FA method is HMM-based, which can hinder its speech recognition accuracy. In future work, we shall investigate other

[9]Note that the number of dictionary entries in `unilex-rpx` is 166.6k, in which some entries share the same word type and pronunciation, only differing in the POS tag. These similar entries are regarded as one unique word type in this work.

more advanced implementations of Equation 9.

V. Conclusion

To facilitate the initial building of an integrated Seq2Seq frontend, this paper first describes how we bootstrap from a pre-existing pipeline-based frontend, to overcome the lack of paired training data. Our experimental results demonstrate the effectiveness of this approach, showing that the resulting Seq2Seq frontend achieves impressive performance. However, a limitation of this bootstrapping approach is that the Seq2Seq frontend's performance is upper-bounded by the pipeline's performance. To overcome this limitation, we propose the FA method, which updates the bootstrapped Seq2Seq frontend using other training sources. This method only requires speech audio and corresponding text, without input from expert linguists. Our experimental results show the effectiveness of the FA method in updating the Seq2Seq frontend. It offers an efficient way to enlarge the "dictionary size" and potentially improves the homograph disambiguation performance of the Seq2Seq frontend.

Acknowledgments

This work was supported in part by the UKRI Centre for Doctoral Training in Natural Language Processing, funded by the UKRI (grant EP/S022481/1), the University of Edinburgh, School of Informatics and School of Philosophy, Psychology & Language Sciences and Huawei. For the purpose of open access, the author has applied a Creative Commons Attribution (CC BY) licence to any Author Accepted Manuscript version arising from this submission.

References

[1] Y. Wang, R. Skerry-Ryan, D. Stanton, Y. Wu, R. J. Weiss, N. Jaitly, Z. Yang, Y. Xiao, Z. Chen, S. Bengio, Q. Le, Y. Agiomyrgiannakis, R. Clark, and R. A. Saurous, "Tacotron: Towards end-to-end speech synthesis," in *Proc. Interspeech 2017*, 2017, pp. 4006–4010.

[2] Y. Yasuda, X. Wang, and J. Yamagishi, "Investigation of learning abilities on linguistic features in sequence-to-sequence text-to-speech synthesis," *Computer Speech & Language*, vol. 67, p. 101183, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0885230820301169

[3] J. Taylor and K. Richmond, "Analysis of pronunciation learning in end-to-end speech synthesis," in *Proc. Interspeech 2019*, 2019, pp. 2070–2074.

[4] Y. Yasuda, X. Wang, S. Takaki, and J. Yamagishi, "Investigation of enhanced Tacotron text-to-speech synthesis systems with self-attention for pitch accent language," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 6905–6909.

[5] J. Shen, Y. Jia, M. Chrzanowski, Y. Zhang, I. Elias, H. Zen, and Y. Wu, "Non-attentive Tacotron: Robust and controllable neural TTS synthesis including unsupervised duration modeling," *ArXiv*, vol. abs/2010.04301, 2020.

[6] Y. Ren, C. Hu, X. Tan, T. Qin, S. Zhao, Z. Zhao, and T.-Y. Liu, "FastSpeech 2: Fast and high-quality end-to-end text to speech," in *International Conference on Learning Representations*, 2021.

[7] I. Elias, H. Zen, J. Shen, Y. Zhang, Y. Jia, R. Skerry-Ryan, and Y. Wu, "Parallel Tacotron 2: A non-autoregressive neural TTS model with differentiable duration modeling," in *Proc. Interspeech 2021*, 2021, pp. 141–145.

[8] K. Park and J. Kim, "g2pE," https://github.com/Kyubyong/g2p, 2019.

[9] M. Bernard and H. Titeux, "Phonemizer: Text to phones transcription for multiple languages in Python," *Journal of Open Source Software*, vol. 6, no. 68, p. 3958, 2021. [Online]. Available: https://doi.org/10.21105/joss.03958

[10] J. Fong, J. Taylor, K. Richmond, and S. King, "A comparison of letters and phones as input to sequence-to-sequence models for speech synthesis," in *Proc. 10th ISCA Speech Synthesis Workshop*, 2019, pp. 223–227. [Online]. Available: http://dx.doi.org/10.21437/SSW.2019-40

[11] T. Hayashi, R. Yamamoto, K. Inoue, T. Yoshimura, S. Watanabe, T. Toda, K. Takeda, Y. Zhang, and X. Tan, "Espnet-TTS: Unified, reproducible, and integratable open source end-to-end text-to-speech toolkit," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 7654–7658.

[12] X. Tan, T. Qin, F. K. Soong, and T.-Y. Liu, "A survey on neural speech synthesis," *ArXiv*, vol. abs/2106.15561, 2021.

[13] A. Conkie and A. M. Finch, "Scalable multilingual frontend for TTS," *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6684–6688, 2020.

[14] J. Pan, X. Yin, Z. Zhang, S. Liu, Y. Zhang, Z. Ma, and Y. Wang, "A unified sequence-to-sequence front-end model for Mandarin text-to-speech synthesis," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 6689–6693.

[15] H.-Y. Kim, J.-H. Kim, and J.-M. Kim, "Fast bilingual grapheme-to-phoneme conversion," in *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Industry Track*. Hybrid: Seattle, Washington + Online: Association for Computational Linguistics, Jul. 2022, pp. 289–296. [Online]. Available: https://aclanthology.org/2022.naacl-industry.32

[16] D. Jurafsky and J. H. Martin, *Speech and Language Processing (2nd Edition)*. USA: Prentice-Hall, Inc., 2009.

[17] R. Sproat and N. Jaitly, "Rnn approaches to text normalization: A challenge," *ArXiv*, vol. abs/1611.00068, 2016.

[18] H. Zhang, R. Sproat, A. H. Ng, F. Stahlberg, X. Peng, K. Gorman, and B. Roark, "Neural models of text normalization for speech applications," *Comput. Linguist.*, vol. 45, no. 2, p. 293–337, Jun. 2019. [Online]. Available: https://doi.org/10.1162/coli_a_00349

[19] J. Zhang, J. Pan, X. Yin, C. Li, S. Liu, Y. Zhang, Y. Wang, and Z. Ma, "A hybrid text normalization system using multi-head self-attention for Mandarin," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 6694–6698.

[20] C. Mansfield, M. Sun, Y. Liu, A. Gandhe, and B. Hoffmeister, "Neural text normalization with subword units," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Industry Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 190–196. [Online]. Available: https://aclanthology.org/N19-2024

[21] W. Jiang, J. Li, M. Chen, J. Ma, S. Wang, and J. Xiao, "Improving neural text normalization with partial parameter generator and pointer-generator network," in *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 7583–7587.

[22] K. Yao and G. Zweig, "Sequence-to-sequence neural net models for grapheme-to-phoneme conversion," *ArXiv*, vol. abs/1506.00196, 2015.

[23] K. Rao, F. Peng, H. Sak, and F. Beaufays, "Grapheme-to-phoneme conversion using long short-term memory recurrent neural networks," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015, pp. 4225–4229.

[24] S. Toshniwal and K. Livescu, "Jointly learning to align and convert graphemes to phonemes with neural attention models," *2016 IEEE Spoken Language Technology Workshop (SLT)*, pp. 76–82, 2016.

[25] H. Sun, X. Tan, J.-W. Gan, H. Liu, S. Zhao, T. Qin, and T.-Y. Liu, "Token-level ensemble distillation for grapheme-to-phoneme conversion," in *Proc. Interspeech 2019*, 2019, pp. 2115–2119.

[26] S. Yolchuyeva, G. Németh, and B. Gyires-Tóth, "Transformer Based Grapheme-to-Phoneme Conversion," in *Proc. Interspeech 2019*, 2019, pp. 2095–2099.

[27] M. Řezáčková, J. Švec, and D. Tihelka, "T5G2P: Using text-to-text transfer transformer for grapheme-to-phoneme conversion," in *Proc. Interspeech 2021*, 2021, pp. 6–10.

[28] L. Dong, Z.-Q. Guo, C.-H. Tan, Y.-J. Hu, Y. Jiang, and Z.-H. Ling, "Neural grapheme-to-phoneme conversion with pre-trained grapheme models," in *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022, pp. 6202–6206.

[29] S. Ö. Arık, M. Chrzanowski, A. Coates, G. Diamos, A. Gibiansky, Y. Kang, X. Li, J. Miller, A. Ng, J. Raiman, S. Sengupta,

and M. Shoeybi, "Deep voice: Real-time neural text-to-speech," in *Proceedings of the 34th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, D. Precup and Y. W. Teh, Eds., vol. 70. PMLR, 06–11 Aug 2017, pp. 195–204. [Online]. Available: https://proceedings.mlr.press/v70/arik17a.html

[30] R. A. Clark, K. Richmond, and S. King, "Multisyn: Open-domain unit selection for the Festival speech synthesis system," *Speech Communication*, vol. 49, no. 4, pp. 317 – 330, 2007. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167639307000398

[31] W. Ping, K. Peng, A. Gibiansky, S. O. Arik, A. Kannan, S. Narang, J. Raiman, and J. Miller, "Deep Voice 3: 2000-speaker neural text-to-speech," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=HJtEm4p6Z

[32] J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, R. Skerry-Ryan, R. A. Saurous, Y. Agiomyrgiannakis, and Y. Wu, "Natural TTS synthesis by conditioning Wavenet on MEL spectrogram predictions," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 4779–4783.

[33] R. Sproat, "Boring problems are sometimes the most interesting," *Computational Linguistics*, vol. 48, no. 2, pp. 483–490, Jun. 2022. [Online]. Available: https://aclanthology.org/2022.cl-2.8

[34] T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, Sep. 2015, pp. 1412–1421. [Online]. Available: https://www.aclweb.org/anthology/D15-1166

[35] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 1715–1725. [Online]. Available: https://www.aclweb.org/anthology/P16-1162

[36] Y. Jia, H. Zen, J. Shen, Y. Zhang, and Y. Wu, "PnG BERT: Augmented BERT on phonemes and graphemes for neural TTS," in *Proc. Interspeech 2021*, 2021, pp. 151–155.

[37] B. Peters, J. Dehdari, and J. van Genabith, "Massively multilingual neural grapheme-to-phoneme conversion," in *Proceedings of the First Workshop on Building Linguistically Generalizable NLP Systems*. Copenhagen, Denmark: Association for Computational Linguistics, Sep. 2017, pp. 19–26. [Online]. Available: https://aclanthology.org/W17-5403

[38] M. Yu, H. D. Nguyen, A. Sokolov, J. Lepird, K. M. Sathyendra, S. Choudhary, A. Mouchtaris, and S. Kunzmann, "Multilingual grapheme-to-phoneme conversion with byte representation," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 8234–8238.

[39] K. Vesik, M. Abdul-Mageed, and M. Silfverberg, "One model to pronounce them all: Multilingual grapheme-to-phoneme conversion with a transformer ensemble," in *Proceedings of the 17th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*. Online: Association for Computational Linguistics, Jul. 2020, pp. 146–152. [Online]. Available: https://aclanthology.org/2020.sigmorphon-1.16

[40] J. Zhu, C. Zhang, and D. Jurgens, "ByT5 model for massively multilingual grapheme-to-phoneme conversion," in *Proc. Interspeech 2022*, 2022, pp. 446–450.

[41] B. Thompson and A. Alshehri, "Improving Arabic diacritization by learning to diacritize and translate," in *Proceedings of the 19th International Conference on Spoken Language Translation (IWSLT 2022)*. Dublin, Ireland (in-person and online): Association for Computational Linguistics, May 2022, pp. 11–21. [Online]. Available: https://aclanthology.org/2022.iwslt-1.2

[42] J. Route, S. Hillis, I. Czeresnia Etinger, H. Zhang, and A. W. Black, "Multimodal, multilingual grapheme-to-phoneme conversion for low-resource languages," in *Proceedings of the 2nd Workshop on Deep Learning Approaches for Low-Resource NLP (DeepLo 2019)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 192–201. [Online]. Available: https://aclanthology.org/D19-6121

[43] A. T. Rutherford, F. Peng, and F. Beaufays, "Pronunciation learning for named-entities through crowd-sourcing," in *Proc. Interspeech 2014*, 2014, pp. 1448–1452.

[44] Z. Kou, D. Stanton, F. Peng, F. Beaufays, and T. Strohman, "Fix it where it fails: Pronunciation learning by mining error corrections from speech logs," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015, pp. 4619–4623.

[45] A. Bruguier, F. Peng, and F. Beaufays, "Learning personalized pronunciations for contact name recognition," in *Proc. Interspeech 2016*, 2016, pp. 3096–3100.

[46] A. Bruguier, D. Gnanapragasam, L. Johnson, K. Rao, and F. Beaufays, "Pronunciation learning with RNN-Transducers," in *Proc. Interspeech 2017*, 2017, pp. 2556–2560.

[47] Çaglar Gülçehre, O. Firat, K. Xu, K. Cho, L. Barrault, H.-C. Lin, F. Bougares, H. Schwenk, and Y. Bengio, "On using monolingual corpora in neural machine translation," *ArXiv*, vol. abs/1503.03535, 2015.

[48] D. Bahdanau, J. Chorowski, D. Serdyuk, P. Brakel, and Y. Bengio, "End-to-end attention-based large vocabulary speech recognition," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016, pp. 4945–4949.

[49] A. Sriram, H. Jun, S. Satheesh, and A. Coates, "Cold fusion: Training Seq2Seq models together with language models," in *Proc. Interspeech 2018*, 2018, pp. 387–391.

[50] A. Kannan, Y. Wu, P. Nguyen, T. N. Sainath, Z. Chen, and R. Prabhavalkar, "An analysis of incorporating an external language model into a sequence-to-sequence model," *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5828, 2018.

[51] S. Young, G. Evermann, M. Gales, T. Hain, D. Kershaw, X. Liu, G. Moore, J. Odell, D. Ollason, D. Povey, A. Ragni, V. Valtchev, P. Woodland, and C. Zhang, *The HTK Book (version 3.5a)*, 12 2015.

[52] M. Mohri, F. Pereira, and M. Riley, "Weighted finite-state transducers in speech recognition," *Computer Speech & Language*, vol. 16, no. 1, pp. 69–88, 2002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0885230801901846

[53] C. Allauzen, M. Riley, J. Schalkwyk, W. Skut, and M. Mohri, "OpenFst: A general and efficient weighted finite-state transducer library," in *Implementation and Application of Automata*, J. Holub and J. Žďárek, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 11–23.

[54] V. Panayotov, G. Chen, D. Povey, and S. 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.

[55] K. Ito and L. Johnson, "The LJ speech dataset," https://keithito.com/LJ-Speech-Dataset/, 2017.

[56] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: http://arxiv.org/abs/1412.6980

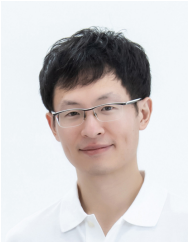
[57] G. Klein, Y. Kim, Y. Deng, J. Senellart, and A. Rush, "OpenNMT: Open-source toolkit for neural machine translation," in *Proceedings of ACL 2017, System Demonstrations*. Vancouver, Canada: Association for Computational Linguistics, Jul. 2017, pp. 67–72. [Online]. Available: https://www.aclweb.org/anthology/P17-4012

[58] E. Bakhturina, V. Lavrukhin, B. Ginsburg, and Y. Zhang, "Hi-Fi multi-speaker English TTS dataset," in *Proc. Interspeech 2021*, 2021, pp. 2776–2780.

VI. Biography Section

Siqi Sun is a doctoral student at the UKRI Centre for Doctoral Training in Natural Language Processing, University of Edinburgh. Before joining the University of Edinburgh, he worked as a research engineer in the speech industry. He received an MSc (Scientific Computing) from University of Pennsylvania, an ME (Computer System Architecture) and a BE (Computer Science) from Shanghai University. His research focuses on speech synthesis, natural language processing and machine learning.

Korin Richmond is a Reader in Speech Technology (UK Associate Professor) at the Centre for Speech Technology Research, University of Edinburgh. With experience in human language and speech technology going back to 1991, he received an MA (Linguistics and Russian, 1995), MSc (Cognitive Science and Natural Language Processing, 1997) and PhD (titled "Estimating Articulatory Parameters from the Acoustic Speech Signal", 2002) from the University of Edinburgh. He has held a teaching post at the University of Edinburgh since 2016, and has diverse research interests including: speech synthesis; pronunciation modelling and lexicography; speech therapy; articulatory data and modelling). He has over 100 publications in speech technology and processing (h-index 32; i10-index 59). He is an IEEE Senior Fellow and has served two 3-year terms as a member of the IEEE Speech and Language Technical Committee.



Hao Tang is a Lecturer in the School of Informatics at the University of Edinburgh. He was a postdoctoral associate working at Massachusetts Institute of Technology, and he completed his PhD at Toyota Technological Institute at Chicago. His research focuses on speech processing and machine learning. He has received best paper awards in Interspeech and ICASSP for his work on self-supervised learning, low-resource automatic speech recognition, and automatic sign language recognition.