공학석사학위논문

Pseudo-labeling, Pre-training with Multilingual Untranscribed Speech Corpora and Fine-tuning for Low-resource Speech Synthesis

전사되지 않은 외국어 음성 데이터를 이용한 슈도 레이블링과 사전 학습, 그리고 저-자원 환경에서의 음성합성을 위한 파인 튜닝

2023 년 2 월

서울대학교 대학원

산업공학과

이 연 수

Pseudo-labeling, Pre-training with Multilingual Untranscribed Speech Corpora and Fine-tuning for Low-resource Speech Synthesis

전사되지 않은 외국어 음성 데이터를 이용한 슈도 레이블링과 사전 학습, 그리고 저-자원 환경에서의 음성합성을 위한 파인 튜닝

지도교수 조 성 준

이 논문을 공학석사 학위논문으로 제출함

2023 년 2 월

서울대학교 대학원

산업공학과

이 연 수

이연수의 공학석사 학위논문을 인준함

2023 년 2 월

위 원 장 이 재 욱 (인)

부위원장 조 성 준 (인)

위 원 홍 성 필 (인)

Abstract

Pseudo-labeling, Pre-training with Multilingual Untranscribed Speech Corpora and Fine-tuning for Low-resource Speech Synthesis

Yeonsu Lee
Department of Industrial Engineering
The Graduate School
Seoul National University

Speech is used in many places as a very effective means of information transmission, and speech synthesis with the modern deep learning models has reduced the need for people to record voice, resulting in improved productivity and cost savings. However, training the majority of text-to-speech (TTS) models requires a large amount of speech-text paired data, which makes it challenging to train a TTS model for languages with less text-labeled data. Noting that voices uttered in other languages share similar pronunciations, we propose to leverage multilingual untranscribed foreign speech corpora, which are relatively easy to procure, for training a TTS model. Concretely, first, we extract features of each foreign waveform containing phoneme information by inputting the waveform into the pre-trained wav2vec 2.0 XLSR-53 model and then perform k-means clustering on these features to obtain pseudo phoneme sequence (pseudo-label) that can play the role of text. Next, we pre-train a

TTS model with the speech-pseudo phoneme sequence data. Finally, we fine-tune the pre-trained model with a small speech-text paired dataset of a target language we originally intended to use. Experimental results showed that the pre-trained models with the multilingual data learned faster and achieved lower CER values, confirming that a multilingual untranscribed speech corpora can help train a TTS model.

Keywords: Speech synthesis, Multilingual untranscribed speech corpora, Clustering, Pseudo phoneme sequence, Pre-training, Fine-tuning

Student Number: 2021-24409

Contents

Abstract **i**

Contents **v**

List of Tables **vi**

List of Figures **viii**

Chapter 1 Introduction **1**

Chapter 2 Related Work **5**

2.1 Leveraging untranscribed speech data 5

2.2 Multi-speaker & Multilingual TTS 6

2.3 Fine-tuning a TTS model 7

Chapter 3 Proposed Method **9**

3.1 Pseudo-labeling 9

3.1.1 Extracting feature sequences of waveforms 9

3.1.2 K-means clustering on the features 10

3.1.3 Pseudo-labeling with center ids 11

3.1.4 Algorithm 11

3.2 Pre-training with pseudo phoneme sequence 13

3.2.1 Network architecture . . . 13
3.2.2 Training loss . . . 15
3.2.3 Algorithm . . . 17
3.3 Fine-tuning with a small speech-text paired dataset of a target language 20
3.3.1 Network architecture . . . 20
3.3.2 Training loss . . . 21
3.3.3 Algorithm . . . 21

Chapter 4 Experimental Setting **24**
4.1 Dataset . . . 24
4.1.1 Multilingual untranscribed speech corpora . . . 24
4.1.2 Speech-text paired data of a target language . . . 25
4.2 Training setting . . . 26
4.2.1 Baseline . . . 27
4.2.2 Proposed method . . . 28
4.3 Training details . . . 28

Chapter 5 Experimental Results **30**
5.1 Character Error Rate . . . 30
5.2 Mel spectrogram . . . 32
5.3 Speaker Embedding Cosine Similarity . . . 34

Chapter 6 Conclusion **38**

Bibliography **40**

Appendix **45**

국문초록 **54**

List of Tables

Table 4.1 Statistics of the multilingual dataset 25

Table 4.2 Statistics of the multilingual dataset used for pre-training . . 25

Table 4.3 Details of the first category settings 27

Table 4.4 Details of the second category settings 27

Table 4.5 Details of the third category settings 28

Table 4.6 Details of the baselines 28

Table 4.7 Details of the models trained with the proposed method . . . 29

Table 5.1 CER of the models we trained 31

Table 5.2 SECS of the models we trained 36

List of Figures

Figure 1 Extracting phoneme-related features 10

Figure 2 K-means clustering on the features 10

Figure 3 Center id sequences as pseudo phoneme sequences 11

Figure 4 Overall procedure to get pseudo phoneme sequences 12

Figure 5 Model architecture for pre-training 18

Figure 6 Speaker encoder architecture 19

Figure 7 Overall procedure to pre-train a TTS model with speech-pseudo phoeneme sequence data 19

Figure 8 Model architecture for fine-tuning 22

Figure 9 Overall procedure to fine-tune the pre-trained TTS model with a small speech-text paired data of a target language . . 23

Figure 10 CER as a performance metric of a TTS model 31

Figure 11 CER trends of all models 32

Figure 12 CER trend comparison of the FF and FN setting 33

Figure 13 Mel spectrograms of the ground truth and generated waveforms 34

Figure 14 Procedure to calculate SECS 35

Figure 15 Visualization of the speaker embeddings with T-SNE 37

Figure 16 Frequency of each cluster in the PEW and PIW setting . . . 46

Figure 17 Frequency of each cluster with German corpus in the PIL setting 47
Figure 18 Frequency of each cluster with Dutch corpus in the PIL setting 48
Figure 19 Frequency of each cluster with French corpus in the PIL setting 49
Figure 20 Frequency of each cluster with Spanish corpus in the PIL setting 50
Figure 21 Frequency of each cluster with Italian corpus in the PIL setting 51
Figure 22 Frequency of each cluster with Portuguese corpus in the PIL setting 52
Figure 23 Frequency of each cluster with Polish corpus in the PIL setting 53

Chapter 1

Introduction

Speech is a very effective way to convey information. For instance, while it is risky to read a newspaper when driving a car, many people listen to the news radio when driving. Because of its unique characteristic, voice is used in many places and will be used in more places with the development of IoT technology which need interfaces that connect human with computers.

Recently, with the advance of deep learning-based speech synthesis models [12, 15, 22, 25, 27], high-quality voices which are hard to distinguish from that of humans can be synthesized. It has become more accessible for people to use text-to-speech (TTS) models for generating voices of diverse speakers when making content such as YouTube videos. Furthermore, an AI news anchor with the TTS model can work 24 hours a day, which is especially useful in disaster situations. In summary, TTS models have brought cost-cutting effects and increased productivity. However, for most TTS models to generate high-quality voices, there needs to be a large speech-text paired dataset (transcribed dataset) [16]. Because there are about 7000 languages in the world, the vast majority of which lack such a large dataset [1], there is a need to explore how to make a well-performing TTS model in low-resource scenarios. One common solution to tackle the problem of training a model with insufficient data is to

leverage other data with similar characteristics that are assumed critical for solving the task the model was developed for. For example, when trying to learn a model to synthesize the voice of someone whose speech-text paired dataset is small, pre-training the model with a large, paired dataset of other people's voices in the same language and then fine-tuning it with the small dataset can synthesize better outputs than it would have without the large dataset [5]. Likewise, when there are insufficient paired data of a specific language, it is noteworthy that voices uttered in the other languages share similar pronunciations. For instance, the sentence "Hi" in English can be written as "ᄒᆞ아ᄋᆡ" in Korean. Moreover, the existence of the International Phonetic Alphabet (IPA), which was designed to represent utterances spoken in all languages over the world, supports that there are overlaps in pronunciations of each other languages. Even though there are differences in the distributions of frequencies of use of, lengths of, and combinations of phonemes in each other languages, it may be possible to cover the various distributions using many language data. In this sense, we investigated how to leverage multilingual foreign data when there are insufficient speech-text paired data of a specific language (target language) for training a TTS model.

In this thesis, we propose to pre-train a TTS model (source model) with multilingual foreign speech corpora without corresponding text, which is easier to get than paired dataset. We assume there are speaker ids and language ids for each waveform. Then we fine-tune the model with a small paired dataset of a target language. Specifically, we make 'pseudo phoneme sequences' corresponding to each speech waveform by extracting features of the waveform via the pre-trained wav2vec 2.0 XLSR-53 [7] and k-means clustering, following [1, 16]. Then, we train a TTS model with the

paired dataset consisting of waveforms and pseudo phonemes in the same way the model would be trained with a speech-text paired dataset. Finally, we reinitialize some model parameters and fine-tune the model with target language data. For fine-tuning to be effective and handle multilingual data, we use YourTTS [3] architecture. It is based on VITS [15] where the invertibility of normalizing flow helps the waveform domain to be an input domain so that the knowledge of the pre-trained model can be utilized. Furthermore, it has language embedding to synthesize voice in different languages.

The main contributions of the thesis are three-fold:

(a) We use a cost-effective and high-accessible dataset, which does not need corresponding multilingual texts that are expensive to label and unfamiliar to most people who use 1 or 2 languages.

(b) We observed that a model trained with the proposed method achieved a lower character error rate than a model trained with only target language data, which implies that multilingual untranscribed speech corpora can be effectively used in low-resource scenarios.

(c) We propose a framework that uses a pre-trained multi-speaker and multilingual model that can be adapted on any language, which can be the groundwork for a model that will eliminate the need to train a TTS model from scratch for any voice in any language.

The thesis consists of 6 chapters. In chapter 2, we review the literature concerning how to handle untranscribed speech data and recent TTS models. In chapter 3, we describe the proposed method. In chapter 4, we explain the settings used for

experiments. In chapter 5, we compare models trained with the proposed method and models trained with only target language data. In chapter 6, we summarize and conclude the thesis and suggest possible future research directions of it.

Chapter 2

Related Work

2.1 Leveraging untranscribed speech data

When trying to train a speech synthesis or speech recognition model, it is common to train it with a speech-text paired dataset. However, there were some researches related to using untranscribed speech data for training them. [1] suggested a way to train a fully unsupervised speech recognition model with separate speech and text corpora. The key idea was extracting features containing phoneme information from waveforms and generating phoneme sequences whose distribution made similar to a phoneme distribution of text corpora via adversarial training. They extracted the features from the transformer block 15 of the pre-trained wav2vec2.0 [2] because they found that a phoneme classifier trained with features of that block had shown the best performance. They observed features of the block 15 of wav2vec 2.0 XLSR-53 model [7] contain phoneme information across a range of languages too. [16] proposed using a sizeable untranscribed speech corpus of the same language when training a TTS model in low-resource settings. They also used features of the block 15 of the wav2vec 2.0 to get pseudo phoneme sequences corresponding to each waveform. They performed k-means clustering on the features to get K clusters, each of which became a pseudo phoneme. After transforming the feature sequences into

corresponding pseudo phoneme sequences and pre-training a TTS model with waveforms and pseudo phoneme sequences, they fine-tuned it with a small paired dataset. They found that untranscribed speech data of the same language can improve the model's performance.

2.2 Multi-speaker & Multilingual TTS

Multi-speaker TTS model is a model that can generate the voices of multiple speakers. It needs a speaker embedding vector containing the information for the model to know whose voice to synthesize when it is given a text prompt of a waveform to generate. There are two typical ways to get a speaker embedding. The first is using speaker id embedding with a lookup table which has unique embedding vectors of each speaker [5, 14, 15, 29, 30]. The second is using a speaker encoder module that takes a reference waveform as input and outputs the speaker embedding of the waveform [3, 28]. Because a model with a speaker encoder module can take an arbitrary waveform as input, once the model is trained, it may be possible to generate a voice of an arbitrary speaker, which makes it called a zero-shot TTS model. If a multi-speaker TTS model uses flow-based architecture with a speaker embedding conditioned, it can disentangle the speaker identity (speaker embedding) and the latent representations containing linguistic contents from a waveform by the invertibility of the flow [3, 14, 15]. It makes the TTS model can convert only the speaker identity of a waveform with the linguistic contents unchanged by changing the extracted speaker embedding, which is called voice conversion.

Multilingual TTS model is a model that can generate voices of multiple languages. [3] used a language embedding in order to let the model know what language

the voice to generate is of. They concatenated a language embedding into a character embedding sequence and used it as input for the text encoder module. The language embedding is also used as input for the duration predictor module that predicts the duration of each character in the waveform domain.

2.3 Fine-tuning a TTS model

If there are insufficient data for training a model, one practical solution is pre-training the model with another similar large dataset and then fine-tuning it with the original small dataset. It works for some TTS models too. [5] proposed the TTS model that can be adapted on an unseen speaker's voice during training with just 20 waveform and text pairs. Especially they only fine-tuned the speaker embedding and the weight matrices for conditional layer normalizations so that they could save ample memory storage. While [5] can only be adapted on speakers whose speech data are transcribed, [29] can be adapted on speakers whose speech data are untranscribed by using the Mel Encoder module. [30] fine-tuned a model pre-trained with a reading-style speech dataset to generate spontaneous speech, which has more dynamic rhythms and filled pauses such as *ah* and *um*, by introducing the FP predictor module and the mixture of expert (MoE) based duration predictor. [16] fine-tuned a model pre-trained with untranscribed speech data of the same language and pseudo phoneme sequences. They pointed out that a general feed-forward TTS model [25, 26, 27] is not appropriate for pre-training with pseudo phoneme sequences. Because the pseudo phoneme embeddings are replaced by randomly initialized character embeddings, which makes features from the model different from those of the pre-training stage, it makes the knowledge from the pre-training stage hard to be

utilized in turn. Thus, they used VITS [15] architecture based on the normalizing flow to exploit the invertibility so that the pseudo phoneme and the character embedding domain became kinds of target domains, and the waveform domain became a kind of input domain in reverse. As the distribution of waveform is unchanged over the pre-training stage and fine-tuning stage, they could fine-tune the model successfully.

Chapter 3

Proposed Method

Our proposed method consists of 3 stages. The first stage makes pseudo phoneme sequences corresponding to each waveform of an multilingual untranscribed dataset, which will play the role of text. The second stage pre-trains a TTS model with those speech-pseudo phoneme sequence data as speech-text paired data. The third stage fine-tunes the TTS model with a small speech-text paired dataset of a target language. We specify details in each section.

3.1 Pseudo-labeling

The overall procedure to get the pseudo phoneme sequences is the same as [16].

3.1.1 Extracting feature sequences of waveforms

When a waveform is given, we extract a feature sequence containing phoneme information of it. Because [1] found the features of the transformer block 15 of the wav2vec 2.0 XLSR-53 helpful in predicting the phoneme sequence of a waveform across a range of languages, we use the feature of the block 15. This process is illustrated in Figure 1.

Figure 1: Extracting phoneme-related features

3.1.2 K-means clustering on the features

We get many features by applying the above procedure to each waveform of the multilingual speech data. Then we perform k-means clustering on that features to get K centers representing the overall feature space related to pronunciations. Clustering can be performed on features of all language speech waveforms without language distinction or performed respectively by language, which is an option for an experimental setting. This process is specified in Figure 2.

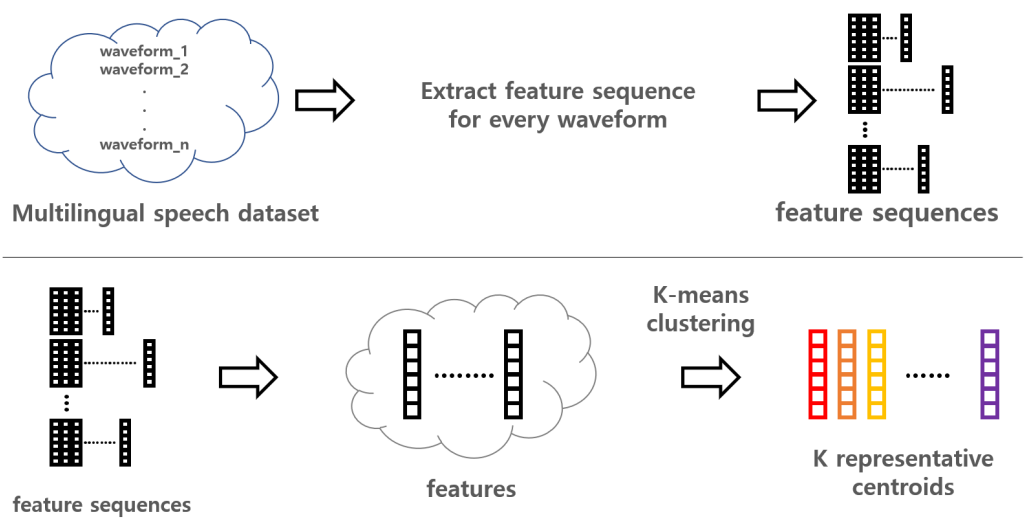

Figure 2: K-means clustering on the features

3.1.3 Pseudo-labeling with center ids

After getting the K centers, we replace each feature sequence with the center sequence by substituting each feature with the nearest center. If the same center appears several times in a row in a center sequence, we delete the repeated parts to make all centers appear once in a row. Finally, we take the center id sequence as the pseudo phoneme sequence. Figure 3 illustrates this process.

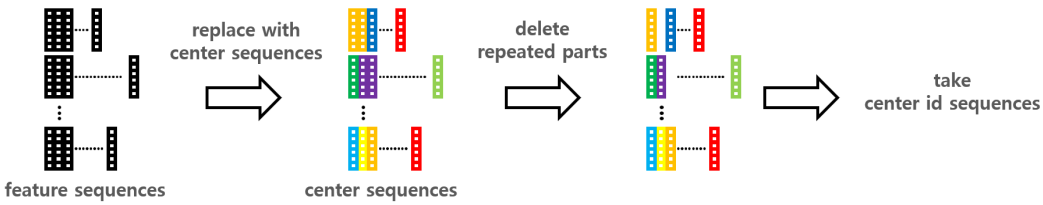

Figure 3: Center id sequences as pseudo phoneme sequences

3.1.4 Algorithm

We specify the overall procedure to get pseudo phoneme sequences in Figure 4.

Algorithm 1 Pseudocode for getting a speech-pseudo phoneme sequence paired dataset.

Input: Untranscribed speech dataset (of all languages or each language) $\mathcal{D}_{\texttt{wav}} = \{(x_{\texttt{wav}})\}$, the number of centers K, Pre-trained wav2vec 2.0 XLSR-53 model h_θ.

```
F ← []
D_fe ← {}
D_pp ← {}
foreach x_wav in D_wav do
    Obtain feature sequence f_wav of the block 15 of h_θ for x_wav
    F.extend(f_wav)
    D_fe.append((x_wav, f_wav))
end for
Get K centers by performing k-means clustering on F
foreach (x_wav, f_wav) in D_fe do
    l ← sequence length of f_wav
    y_pp ← []
    for i in [1..l] do
        f ← ith feature of f_wav
        j ← id of f's nearest center among the K centers
        if y_pp[−1] ≠ j then
            y_pp.append(j)
        end if
    end for
    D_pp.append((x_wav, y_pp))
end for
return D_pp;
```

Figure 4: Overall procedure to get pseudo phoneme sequences

3.2 Pre-training with pseudo phoneme sequence

We pre-train a TTS model with the speech-pseudo phoneme sequence data as though they are speech-text paired data. However, for fine-tuning to be effective, it is recommended to use a model with an invertible mapping structure [10, 18] rather than a typical feed-forward model [25, 26, 27], as mentioned in [16]. In addition, in terms of handling multiple languages, we utilize YourTTS [3] model structure, which is a flow-based model and allows us to handle multilingual data through language embeddings.

3.2.1 Network architecture

Figure 5 illustrates the overall model architecture for pre-training. To explain the model components, the model is a kind of VAE [19] model. There is a posterior encoder that calculates the posterior distribution of latent feature z from the input and a decoder trying to restore z to the input again. However, while the input is a waveform, the posterior encoder itself instead receives the linear spectrogram as the input, which is calculated by applying a closed-form formula to the waveform. The decoder is named HiFi-GAN Decoder because it uses the HiFi-GAN [20] network structure, which was developed as a vocoder that converts a mel spectrogram to the corresponding waveform. Both the posterior encoder and the decoder are conditioned on a speaker embedding containing the speaker information.

There are two main ways to get a speaker embedding. One is to use a lookup table containing unique embedding values for each speaker, and the other is to obtain an embedding vector from a speaker encoder, which is a module that receives a waveform as input. Which to use depends on the experimental setting, and we

use both settings in experiments. In the speaker encoder case, we use a model with a structure different from that of [3]. Given that [6] showed promising results in voice conversion by using the block 1 feature of the wav2vec2.0 as the input to the speaker encoder with the ECAPA-TDNN [8] structure to obtain an embedding relating speaker's timbre, we use that same architecture. The structure of the speaker encoder we used is illustrated in Figure 6.

The prior distribution of latent feature z is modeled using a text encoder and a normalizing flow. To get prior distribution, we first get a sequence of means and variances of normal distributions by the transformer-based text encoder, which takes as input a concatenation of a pseudo phoneme embedding sequence and a language embedding. The normal distribution sequence corresponds to the pseudo phoneme sequence, which means each normal distribution corresponds to each pseudo phoneme. As there is no guarantee that the length of this mean and variance sequence, which is equal to the length of the pseudo phoneme sequence, is the same as the length of the latent z computed by the posterior encoder, we modify the distribution sequence by repeating some normal distributions to match the lengths. Which normal distribution (pseudo phoneme) and how many to repeat is determined by the monotonic alignment search [14]. Then the modified distribution sequence is transformed to be the prior distribution of latent z by the normalizing flow with the rule of change-of-variable. In addition, since the normalizing flow is also conditioned on speaker embedding, it can be seen as conditional vae in that the prior distribution is dependent on a pseudo phoneme sequence, a language embedding, and a speaker embedding.

In [3, 15], the duration predictor is also trained to predict how many times each

character would be repeated by the monotonic alignment search at inference time. However, we do not need the duration predictor as we will not infer with the pre-trained model. Thus, we can skip training the duration predictor, which is why Figure 5 does not have a duration predictor.

Finally, there is a discriminator for adversarial training to reconstruct the input more naturally. The discriminator has a multi-period discriminator structure of Hifi-GAN [20].

3.2.2 Training loss

The loss function to optimize is the same as [3, 15], except for the term for training the duration predictor. Because of the adversarial training, the process of updating the discriminator and the generator sequentially is repeated. In this case, the generator means all modules except the discriminator.

The discriminator loss term is computed as Equation 3.1 borrowed from the LS-GAN [21]. $\mathcal{D}, \mathcal{G}, x, x_{lin}, z, \mu_\phi(x_{lin}), \sigma_\phi(x_{lin})$ denote the discriminator, the generator, an input waveform, the corresponding linear spectrogram, the latent variable, the mean and variance of the posterior distribution computed by the posterior encoder given the waveform, respectively.

$$\mathcal{L}_{adv}(\mathcal{D}) = \mathbb{E}_{(x,z)}[(\mathcal{D}(x) - 1)^2 + (\mathcal{D}(\mathcal{G}(z)))^2] \tag{3.1}$$
$$\text{given } x,\ z \sim q_\phi(z|x_{lin}) = \mathcal{N}(z; \mu_\phi(x_{lin}), \sigma_\phi(x_{lin}))$$

The loss for the generator is the sum of several terms. The first term is for reconstructing an input waveform from the latent variable z, which is computed as

Equation 3.2. $x_{mel}, \mathcal{G}(z)_{mel}$ denote the mel spectrogram of the waveform and the reconstructed waveform from the latent variable, respectively.

$$\mathcal{L}_{recon}(\mathcal{G}) = \mathbb{E}_{(x,z)}[\|x_{mel} - \mathcal{G}(z)_{mel}\|_1] \tag{3.2}$$

The second term is the KL-divergence term making the posterior distribution and the prior distribution of the latent variable similar, which is computed as Equation 3.3. $t, A, c, q_\phi(z|x_{lin}), p_\theta(z|c), f_\theta(z), \mu_\theta(c), \sigma_\theta(c)$ denote text (pseudo phoneme sequence in pre-training stage), an alignment computed from the monotonic alignment search, condition information (not consider a speaker embedding for simplicity), a probability of the posterior distribution and the prior distribution, the inverted value of z by the normalizing flow, mean and variance of the modified normal distribution sequence, respectively.

$$\begin{aligned} \mathcal{L}_{KL}(\mathcal{G}) &= \mathbb{E}_{(x,z,t)}[\log q_\phi(z|x_{lin}) - \log p_\theta(z|c)] \\ p_\theta(z|c) &= \mathcal{N}(f_\theta(z); \mu_\theta(c), \sigma_\theta(c))|\det \frac{\partial f_\theta(z)}{\partial z}| \\ &\text{where } c = [t, A] \end{aligned} \tag{3.3}$$

The third term is for making the generator deceive the discriminator, which is computed as Equation 3.4.

$$\mathcal{L}_{adv}(\mathcal{G}) = \mathbb{E}_{(x,z)}[(\mathcal{D}(\mathcal{G}(z)) - 1)^2] \tag{3.4}$$

The last fourth term is additional feature matching loss, which helps adversarial training, computed as Equation 3.5. $N_l, \mathcal{D}^l(x), \mathcal{D}^l(\mathcal{G}(z))$ denote the number of layers

of the discriminator, the feature of the l-th layer of the discriminator when it takes as input a waveform and the reconstructed waveform, respectively.

$$\mathcal{L}_{fm}(\mathcal{G}) = \mathbb{E}_{(x,z)}[\frac{1}{N_l}\|\mathcal{D}^l(x) - \mathcal{D}^l(\mathcal{G}(z))\|_1] \tag{3.5}$$

Therefore, the final loss for pre-training the generator is computed as Equation 3.6. λs are weights for each term.

$$\mathcal{L}_{gen}(\mathcal{G}) = \lambda_{recon}\mathcal{L}_{recon}(\mathcal{G}) + \lambda_{KL}\mathcal{L}_{KL}(\mathcal{G}) + \lambda_{adv}\mathcal{L}_{adv}(\mathcal{G}) + \lambda_{fm}\mathcal{L}_{fm}(\mathcal{G}) \tag{3.6}$$

3.2.3 Algorithm

We specify the overall procedure to pre-train a TTS model with speech-pseudo phoneme sequence data in Figure 7.

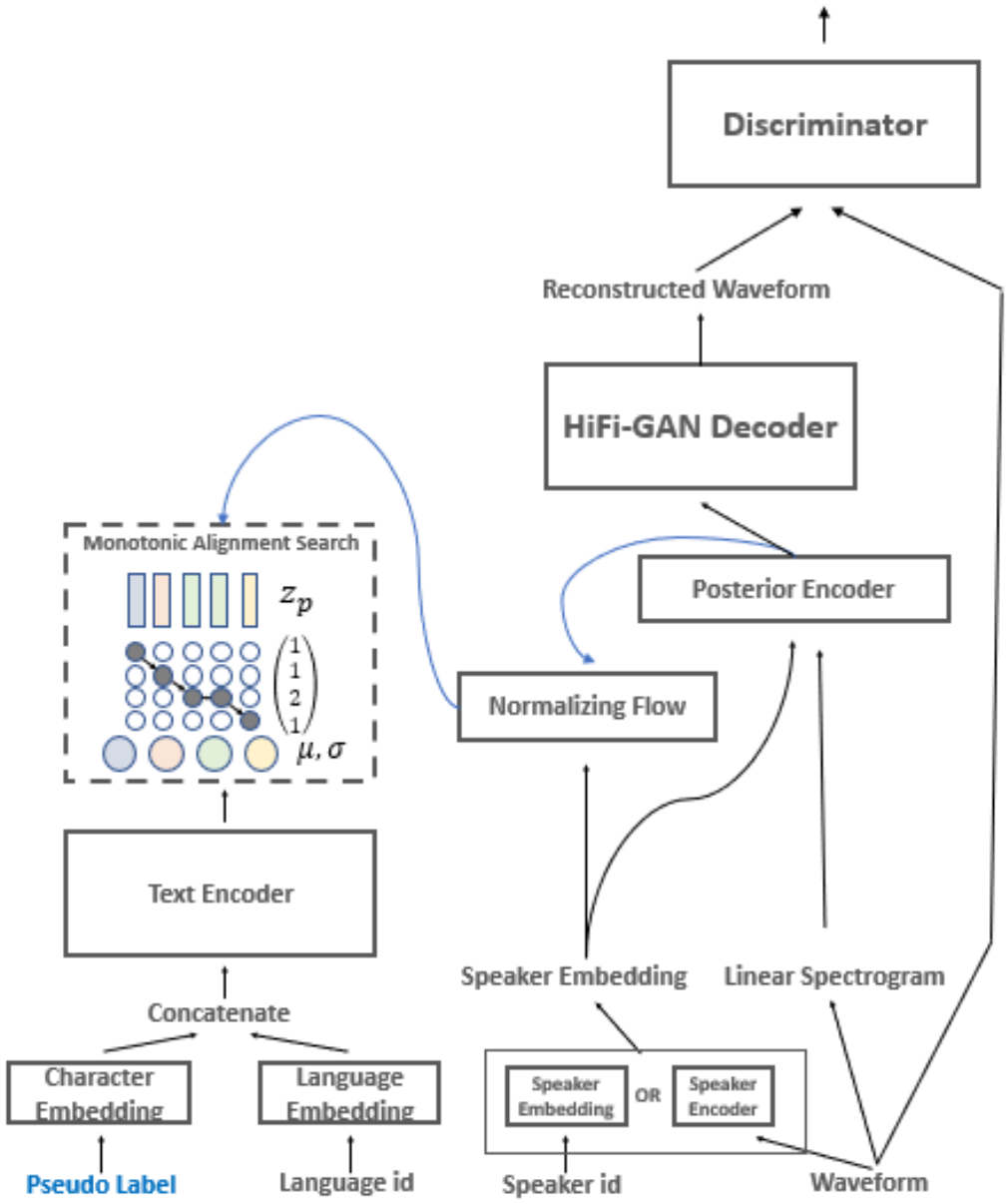

Figure 5: Model architecture for pre-training

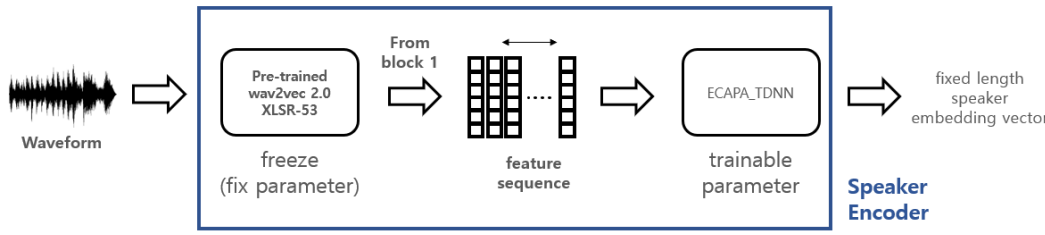

Figure 6: Speaker encoder architecture

Algorithm 2 Pseudocode for pre-training a TTS model with a speech-pseudo phoneme sequences dataset.

Input: Speech-pseudo phoneme sequence dataset $\mathcal{D}_{\text{pp}} = \{(x_{\text{wav}}, y_{\text{pp}})\}$,
TTS model Generator $\mathcal{G}_\theta$, Discriminator $\mathcal{D}_\phi$, mini batch size k.

while not converged **do**
  sample mini batch $\mathcal{B}$ of size k
  $\mathcal{L}_{adv}(\mathcal{D}_\phi) \leftarrow$ compute the loss for discriminator with B and Equation 3.1
  update discriminator with the loss $\mathcal{L}_{adv}(\mathcal{D}_\phi)$

  $\mathcal{L}_{gen}(\mathcal{G}_\theta) \leftarrow$ compute the loss for generator with B and Equation 3.6
  update generator with the loss $\mathcal{L}_{gen}(\mathcal{G}_\theta)$
end while
return $\mathcal{G}_\theta$;

Figure 7: Overall procedure to pre-train a TTS model with speech-pseudo phoeneme sequence data

3.3 Fine-tuning with a small speech-text paired dataset of a target language

We fine-tune the pre-trained model with a small paired dataset of a target language. It is almost the same as the pre-training stage. There are only slight differences.

3.3.1 Network architecture

Figure 8 illustrates the overall model architecture for fine-tuning. The parameters of the posterior encoder and HiFi-GAN Decoder can be frozen or fine-tuned, which is an option for an experimental setting. Although [16] froze the parameters of them, we considered the option of fine-tuning them in that the distribution of waveform of the multilingual dataset may not cover the distribution of waveform of the target language dataset properly.

For speaker embeddings, we train a new lookup table from scratch if we used the setting with a lookup table at pre-training. If we used the setting with a speaker encoder at pre-training, we can freeze or fine-tune the parameter of the speaker encoder, which is also an option for an experimental setting.

We train new lookup tables for text character and target language embedding from scratch as they have to learn unique embedding for each input that has never been seen in the pre-training stage. We also train a new text encoder from scratch that takes the concatenation of two embeddings as input. Meanwhile, we fine-tune the normalizing flow module in that it receives features extracted from a waveform as input by its invertibility.

There is a duration predictor in the fine-tuning stage. As mentioned before, the duration predictor predicts how many times each character would be repeated, which

is the duration sequence, by the monotonic alignment search at inference time. We train it from scratch.

We train a new discriminator from scratch as well.

3.3.2 Training loss

The loss function to optimize is exactly the same as [3, 15]. We use the same loss function for the discriminator as Equation 3.1. The loss function for the generator is also the same, except that now it has the term for the duration predictor. The loss term for the duration predictor is computed as Equation 3.7. $u, v, d, q_\phi(u, v|d, t), p_\theta(d - u, v|t)$ denote a random variable for variational dequantization [11], a random variable for variational data augmentation [4], the ground truth duration sequence obtained from the monotonic alignment search, and distributions of the random variables for variational lower bound, respectively.

$$\mathcal{L}_{dp}(\mathcal{G}) = \mathbb{E}_{q_\phi(u,v|d,t)}[\log p_\theta(d - u, v|t) - \log q_\phi(u, v|d, t)] \tag{3.7}$$

Thus, the final loss for the generator is computed as Equation 3.8, where λs are weights for each term.

$$\mathcal{L}_{gen}(\mathcal{G}) = \lambda_{recon}\mathcal{L}_{recon}(\mathcal{G}) + \lambda_{KL}\mathcal{L}_{KL}(\mathcal{G}) + \lambda_{adv}\mathcal{L}_{adv}(\mathcal{G}) + \lambda_{fm}\mathcal{L}_{fm}(\mathcal{G}) + \lambda_{dp}\mathcal{L}_{dp}(\mathcal{G}) \tag{3.8}$$

3.3.3 Algorithm

We specify the overall procedure to fine-tune the pre-trained TTS model with a small speech-text paired data of a target language in Figure 9.

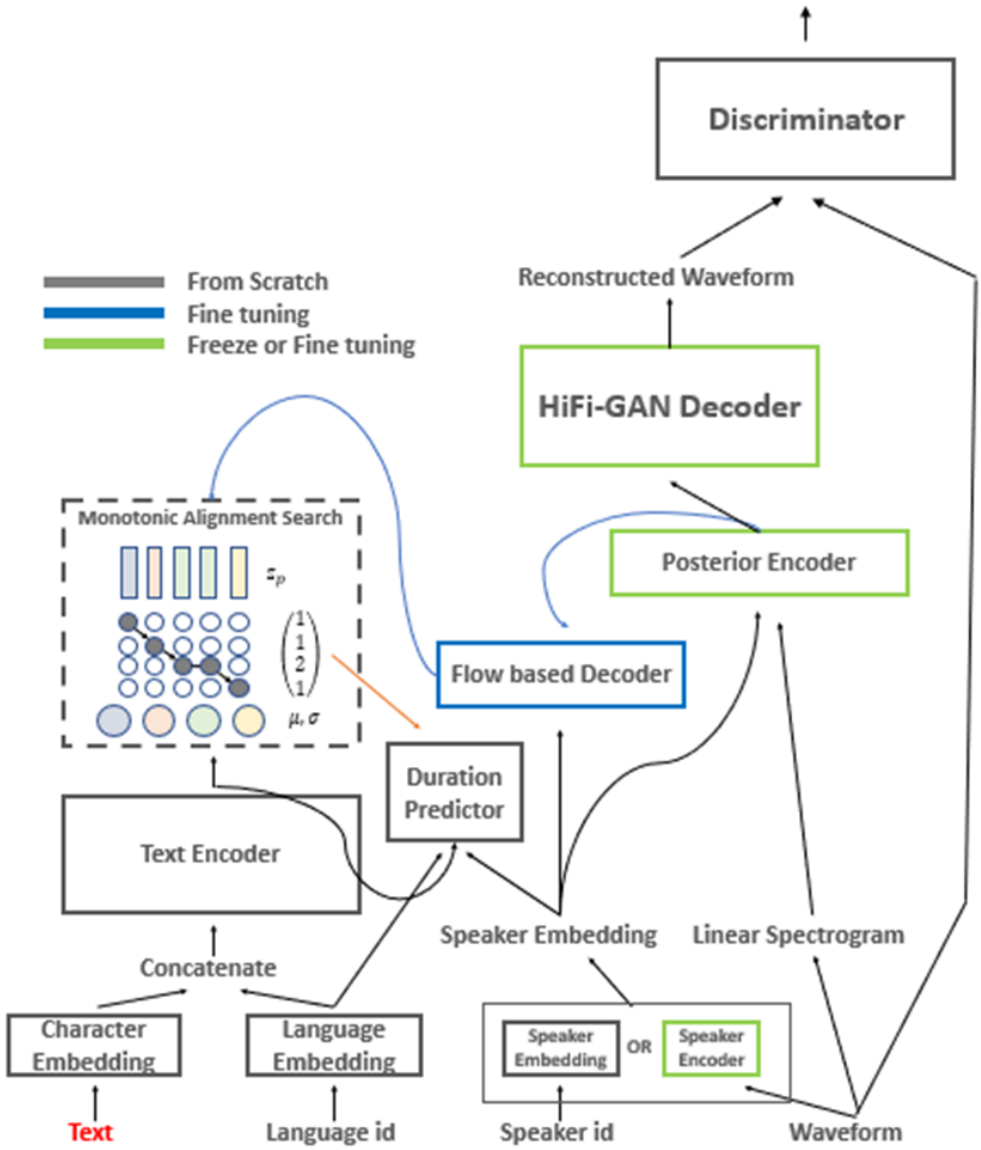

Figure 8: Model architecture for fine-tuning

Algorithm 3 Pseudocode for fine-tuning the TTS model with a speech-text paired dataset of a target language.

Input: Speech-text paired dataset of a target language $\mathcal{D}_{\texttt{tl}} = \{(x_{\texttt{wav}}, y_{\texttt{text}})\}$, pre-trained TTS model Generator $\mathcal{G}_\theta$, pre-trained Discriminator $\mathcal{D}_\phi$, mini batch size k, parameter index set θ_s, θ_f.

reinitialize all the parameters of $\mathcal{D}_\phi$
reinitialize the parameters of $\mathcal{G}_\theta$ whose index is in θ_s
freeze the parameters of $\mathcal{G}_\theta$ whose index is in θ_f
add Duration predictor module to TTS model $\mathcal{G}_\theta$
while not converged **do**
    sample mini batch $\mathcal{B}$ of size k
    $\mathcal{L}_{adv}(\mathcal{D}_\phi) \leftarrow$ compute the loss for discriminator with B and Equation 3.1
    update discriminator with the loss $\mathcal{L}_{adv}(\mathcal{D}_\phi)$

    $\mathcal{L}_{gen}(\mathcal{G}_\theta) \leftarrow$ compute the loss for generator with B and Equation 3.8
    update generator with the loss $\mathcal{L}_{gen}(\mathcal{G}_\theta)$
end while
return $\mathcal{G}_\theta$;

Figure 9: Overall procedure to fine-tune the pre-trained TTS model with a small speech-text paired data of a target language

Chapter 4

Experimental Setting

4.1 Dataset

This section describes the dataset used in the experiments. There are two main datasets used: the first is a multilingual foreign language dataset consisting only of speech waveforms for pre-training, and the second is a small speech-text paired dataset of a target language.

4.1.1 Multilingual untranscribed speech corpora

We used a part of the Multilingual LibriSpeech (MLS) [23] dataset. The MLS dataset itself consists of a total of eight languages: English, German, Dutch, French, Spanish, Italian, Portuguese, and Polish, with 44.5 k hours of voice and text data in English and a total of 6 k hours of voice and text data in the remaining languages. However, since we used English as the target language, seven language data excluding English were used to prevent a pre-trained model from seeing English data in the pre-training stage. In addition, since we had limited computing resources and only speech waveforms were needed, a total of about 63 hours of data consisting of about 9 hours of speech data per language were used as the full multilingual speech corpora data. The sample rate of all waveforms is 16kHz. The statistics of the multilingual

Table 4.1: Statistics of the multilingual dataset

Language	Num of waveforms	Total time (hr)	Num of speakers
German	2194	9.00	27
Dutch	2153	9.00	21
French	2167	9.00	30
Spanish	2110	9.00	29
Italian	2173	9.00	30
Portuguese	2116	9.00	23
Polish	2173	9.00	9

Table 4.2: Statistics of the multilingual dataset used for pre-training

Language	Num of waveforms	Total time (hr)	Num of speakers
German	1990	8.17	27
Dutch	1936	8.08	21
French	1955	8.14	30
Spanish	1889	8.07	29
Italian	1929	7.98	30
Portuguese	1919	8.16	23
Polish	1959	8.13	9

dataset are described in Table 4.1.

A complete dataset of 63 hours was used for obtaining pseudo phoneme sequences. However, when pre-training with pseudo phoneme sequences, only about 90% of the total data were used for pre-training, and the remaining data were used to check whether overfitting occurred. The statistics of the dataset used for pre-training are described in Table 4.2.

4.1.2 Speech-text paired data of a target language

We used English as the target language and used a part of the LibriTTS dataset [31]. To handle low-resource scenarios, a total of about 100 minutes of speech-text

paired dataset was used for fine-tuning the model, which was 10 minutes per speaker consisting of a total of 10 speakers. The ids of the speakers were 3003, 2204, 8080, 3922, 7982, 3307, 5935, 3032, 3274, and 1271, consisting of five women and five men. Although the original sample rate of LibriTTS is 22 kHz, we reduced it to 16 kHz to match it with the sample rate of the previously used dataset for pre-training.

4.2 Training setting

The setting of how to train a model is divided into three categories to explain. The first is about the model architecture for pre-training and whether to perform clustering for pseudo-labeling by language. The second is about the model architecture for fine-tuning. The third is about whether to freeze the parameters of some modules at the fine-tuning stage.

The first category has settings called PEW, PIW, and PIL, respectively. For the model architecture, we can use a lookup table or speaker encoder for speaker embeddings and choose whether to use a language-specific text encoder with language-specific relative positional encoding in these settings. The details of the settings are described in Table 4.3.

The second category has settings called TE and TI, respectively. Similarly, we can use a lookup table or speaker encoder in these settings. The details of the settings are described in Table 4.4.

The third category has settings called FF and FN, respectively. We can choose whether to freeze the parameters of some modules, which are the posterior encoder, decoder, and speaker encoder if the model has it. The details of the settings are

Table 4.3: Details of the first category settings

	Description
PEW	Perform clustering on the whole corpora to get pseudo phoneme sequence and pre-train a model with the speaker encoder
PIW	Perform clustering on the whole corpora to get pseudo phoneme sequence and pre-train a model with the lookup table
PIL	Perform clustering on each language's corpus to get pseudo phoneme sequence and pre-train a model with the lookup table and the language-specific text encoder

Table 4.4: Details of the second category settings

	Description
TE	Use the model architecture with the speaker encoder at the fine-tuning stage
TI	Use the model architecture with the lookup table at the fine-tuning stage

described in Table 4.5.

4.2.1 Baseline

The critical point of the experiments is to compare whether the model with pre-training is better than the model without the pre-training. We call a model without pre-training as a baseline. The baseline can also have a lookup table or speaker encoder for speaker embeddings. Thus, there are two baselines: BASE_TE and BASE_TI, which are named after the second category settings. The details of the baselines are described in Table 4.6.

Table 4.5: Details of the third category settings

	Description
FF	Freeze the parameters of posterior encoder, decoder, (speaker encoder)
FN	Do not freeze the parameters of posterior encoder, decoder, (speaker encoder)

Table 4.6: Details of the baselines

	Description
BASE_TE	Baseline model with the speaker encoder - trained only with target language data
BASE_TI	Baseline model with the lookup table - trained only with target language data

4.2.2 Proposed method

There are several combinations of the settings. For example, we can pre-train a model with the PEW setting and then fine-tune it with the TE and FF settings. We call the model trained with this combination of the settings PEW_TE_FF. Similarly, if we pre-train a model with the PIW setting and then fine-tune it with the TI and FN settings, we call this model PIW_TI_FN. All the models trained with our proposed method are described in Table 4.7.

4.3 Training details

In order to obtain pseudo phoneme sequences, K-means clustering must be performed on a large number of features. We used the FAISS library [13] for efficient processing. We used 128 for the K value for k-means clustering. Therefore, there are

Table 4.7: Details of the models trained with the proposed method

	Description
PEW_TE_FF	Model pre-trained with PEW and fine-tuned with TE and FF
PEW_TE_FN	Model pre-trained with PEW and fine-tuned with TE and FN
PIW_TI_FF	Model pre-trained with PIW and fine-tuned with TI and FF
PIW_TI_FN	Model pre-trained with PIW and fine-tuned with TI and FN
PIL_TI_FF	Model pre-trained with PIL and fine-tuned with TI and FF
PIL_TI_FN	Model pre-trained with PIL and fine-tuned with TI and FN

a total of 129 tokens consisting of 128 pseudo phonemes plus 1 pad token for the whole multilingual speech corpora in the PEW and PIW settings. Likewise, there are a total of 897 tokens consisting of 128 pseudo phonemes per language plus 1 pad token for the whole multilingual speech corpora in the PIL settings.

Hyper-parameters for the network architecture, such as the dimension of speaker embedding, are the same as that of YourTTS implementation of coqui [1]. All the experiments except pre-training with the PIL setting used batch size 16 and the Adam optimizer [17] with $\beta_1 = 0.9$, $\beta_2 = 0.999$ and learning rate 10^{-4}. We used batch size 21 for pre-training with the PIL setting for the language-specific positional encoding. Baselines are trained with the target language dataset for 180k steps, and the models trained with our proposed method are pre-trained for 350k steps, then fine-tuned for 180k steps. [2]

[1] The code is publicly available at `https://github.com/coqui-ai/TTS`

[2] The code is publicly available at `https://github.com/l22ys/multilingual_unlabeled_pretrain_tts`

Chapter 5

Experimental Results

5.1 Character Error Rate

We used Character Error Rate (CER) as a metric of the performance of a TTS model. CER indicates a kind of distance between two text sentences, which is computed as Equation 5.1. N, S, D and I denote the length of one sentence (reference text), the required number of substitutions, deletions and insertions at the character level to transform another sentence (prediction text) into the reference text, respectively. The lower the CER, the closer the two sentences, and when the two sentences are the same, the CER becomes 0.

$$\text{CER} = \frac{S + D + I}{N} \tag{5.1}$$

Figure 11 describes how we used CER as a metric of the performance of a TTS model. An input text for a TTS model is used as a reference text, and the generated waveform is inputted into a high-performance automatic speech recognition model (ASR) to get recognized text. The recognized text is used as the prediction text. Assuming that the ASR model recognizes sentences as accurately as humans, the CER will be lower as the speech synthesis model generates proper waveform. Therefore,

Table 5.1: CER of the models we trained

	CER(%)
BASE_TE	22.35
BASE_TI	27.07
PEW_TE_FF	16.69
PEW_TE_FN	**10.87**
PIW_TI_FF	14.30
PIW_TI_FN	11.89
PIL_TI_FF	14.48
PIL_TI_FN	12.53

we regard a TTS model with a lower CER as a better model.

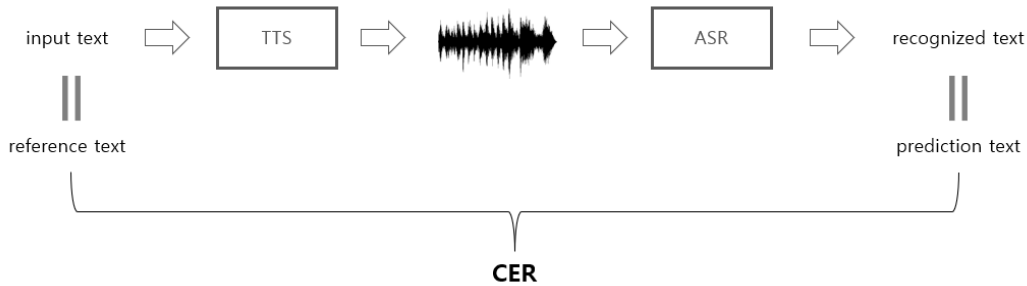

Figure 10: CER as a performance metric of a TTS model

We calculated the average CER for n sentences that had not been seen in the training stage for every model we trained. To recognize the generated text, we used a pre-trained ASR model Whisper [24] from OpenAI.[3] The CER of the models at the 180k step are shown in Table 5.1. Figure 11 shows the trends of the CER according to the number of training steps of all models.

Table 5.1 and Figure 11 tell us that the average CER values of the models trained with our proposed method were lower and decreased faster than those of the mod-

[3]The code is publicly available at `https://github.com/openai/whisper`

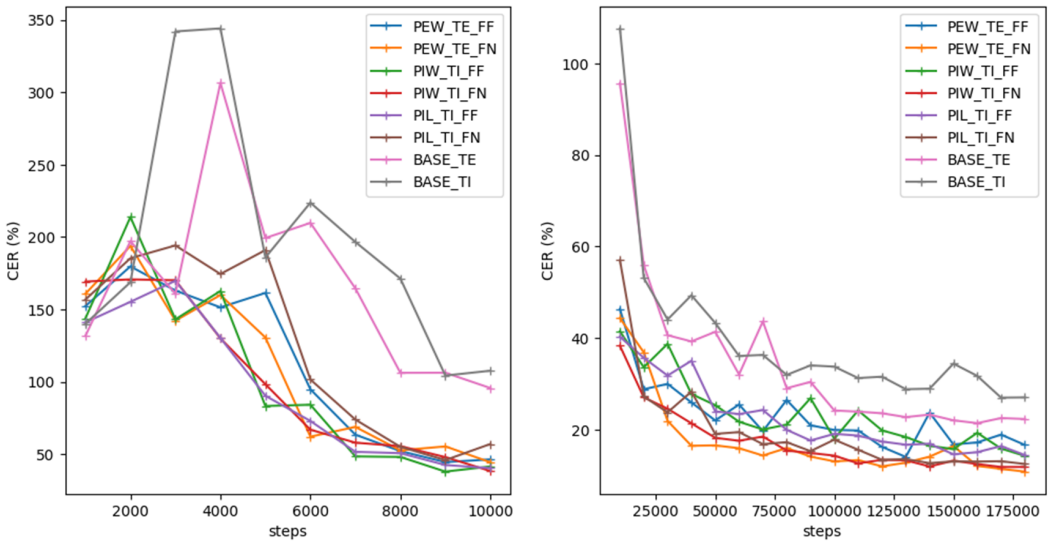

Figure 11: CER trends of all models

els only trained with the target language dataset, which means the effectiveness of multilingual untranscribed speech corpora for speech synthesis. Furthermore, Figure 12 shows us that the FN setting, in which we did not freeze the parameters of some modules, achieved better performance than the FF setting. It may validate our assumption that the multilingual dataset might not fully cover the distribution of waveform of the target language dataset.

5.2 Mel spectrogram

We observed that pre-training helped the models learn faster than those without pre-training. To take a closer look at the differences in the generated waveforms in the early stages of learning, we looked at the mel spectrogram of the waveform generated by each model trained for 10k steps, receiving text sentences they had not seen during training as input. A mel spectrogram represents a two-dimensional image

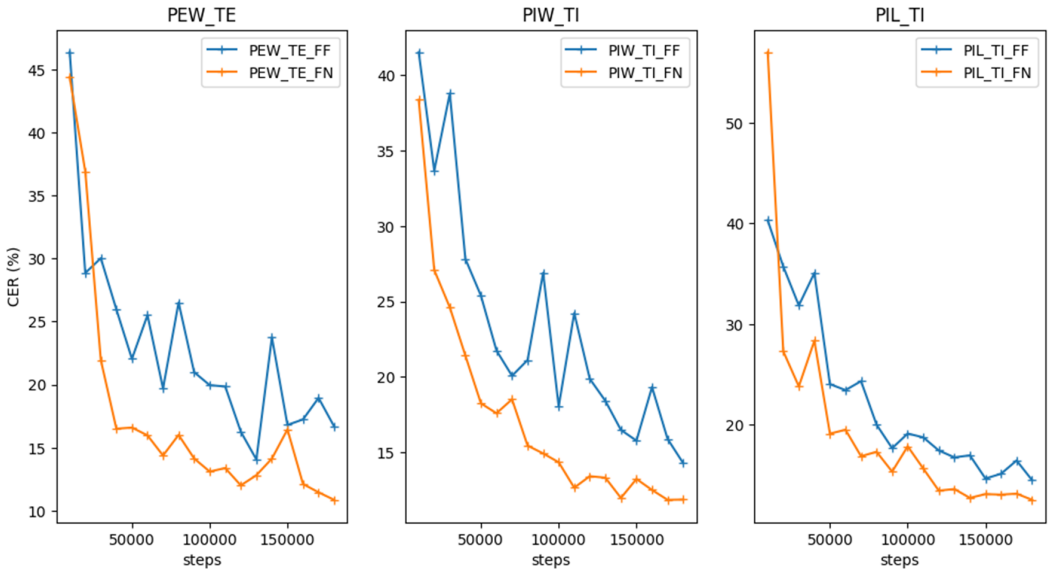

Figure 12: CER trend comparison of the FF and FN setting

of a waveform, with the horizontal axis representing the time axis and the vertical axis representing the frequency axis, and visualizing how strongly components in a specific frequency range exist at each time range.

Figure 13 shows us the mel spectrograms of the ground truth and generated waveforms. We found that while the mel spectrograms of the baselines are blurry, the mel spectrograms of the models with pre-training tended to have somewhat sharply separated frequency components compared to the baselines. It seems to have been influenced by the ability of the posterior encoder and the decoder to reconstruct multilingual speech waveforms, which helped the pre-trained model to adapt fast to reconstruct the waveforms of the target language dataset.

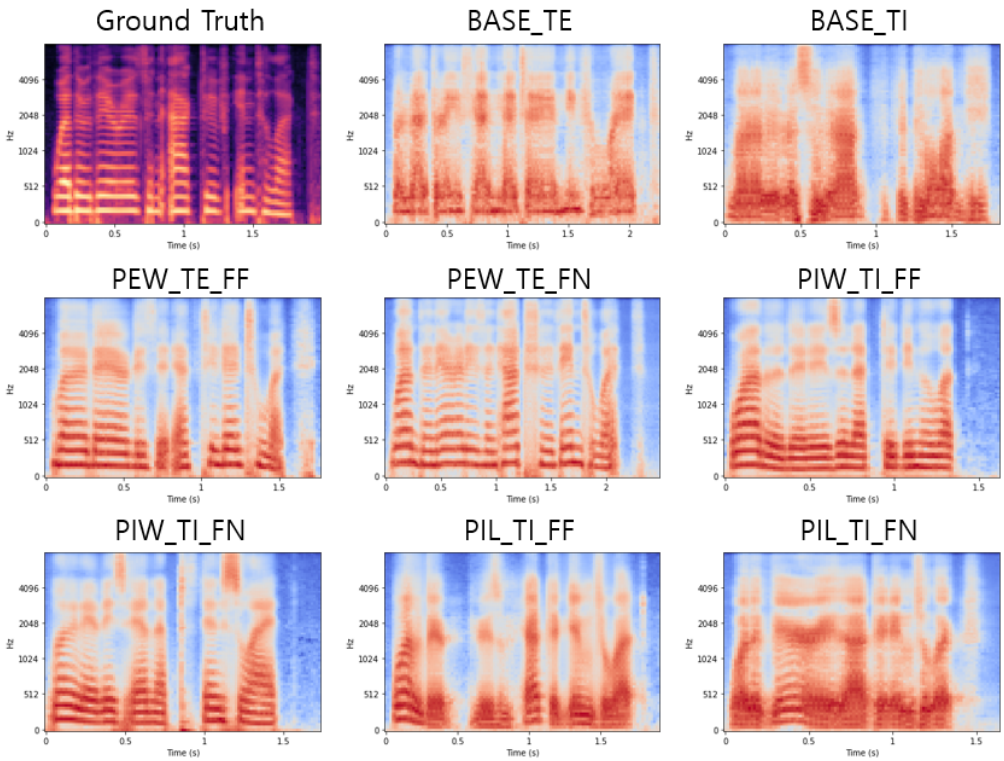

Figure 13: Mel spectrograms of the ground truth and generated waveforms

5.3 Speaker Embedding Cosine Similarity

When evaluating the performance of a TTS model, it is also possible to evaluate whether a waveform can be generated in the intended voice. It would be difficult to say that a model is a good model if it generates another person's voice rather than the intended one. Speaker Embedding Cosine Similarity (SECS) was used as a metric to evaluate this aspect. SECS is computed as Equation 5.2. e_a, e_b denote the speaker embedding vector of waveform a and b, respectively. SECS ranges from -1 to 1, and the larger the SECS of e_a, e_b, the more similar the traits of the speakers of waveform a and b are, which people use to distinguish who speaks. Because

speaker embedding e_a, e_b should contain information for determining the owner of the waveform, we used a pre-trained model for speaker verification task to extract speaker embeddings from waveforms.[4]

$$\text{SECS}(e_a, e_b) = \frac{e_a \cdot e_b}{\|e_a\|_2 \|e_b\|_2} \tag{5.2}$$

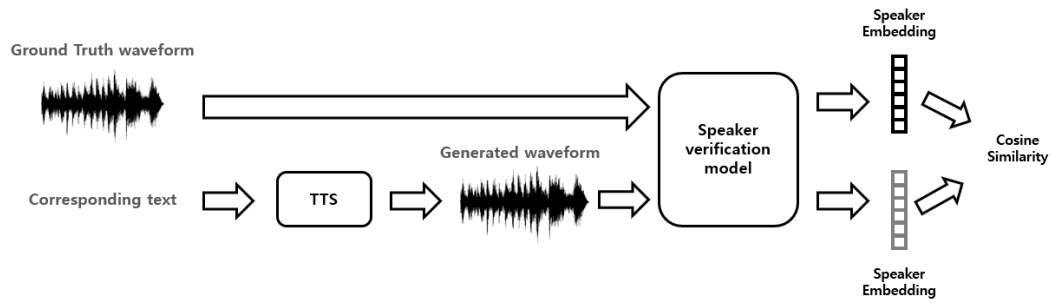

Figure 14: Procedure to calculate SECS

Figure 14 illustrates the procedure to calculate SECS as a metric of the performance of a TTS model. As we did for CER, we calculated the average SECS for n sentences that had not been seen in the training stage for every model we trained. A higher SECS means that the model is better at generating a waveform of the intended speaker's voice.

The average SECS values of all models are shown in Table 5.2. Although the PEW_TE_FN model achieved the highest SECS, we observed that whether a model is pre-trained with the multilingual dataset was irrelevant to the performance at generating the intended voice. Instead, it seemed important whether the model had a speaker encoder, considering that top-3 models all had the speaker encoder.

Additionally, we visualized speaker embeddings on 2-dimensional space using T-

[4]The model is publicly available at `https://huggingface.co/speechbrain/spkrec-ecapa-voxceleb`

Table 5.2: SECS of the models we trained

	SECS
BASE_TE	0.47
BASE_TI	0.16
PEW_TE_FF	0.28
PEW_TE_FN	**0.53**
PIW_TI_FF	0.11
PIW_TI_FN	0.11
PIL_TI_FF	0.12
PIL_TI_FN	0.18

SNE in Figure 15. Each point indicates a speaker embedding of the corresponding waveform, and the color of the point indicates the speaker of its waveform. We found that the 2-dimension area was divided by gender for ground truth waveforms. Likewise, we could find a similar tendency for the BASE_TE, PEW_TE_FF, and PEW_TE_FN, which had the speaker encoder. However, this was not the case for other models where some speakers existed whose embeddings were closer to those of someone of a different gender. We interpreted these results to mean the model with the speaker encoder was more able to capture the speaker information, which aligns with the result from SECS.

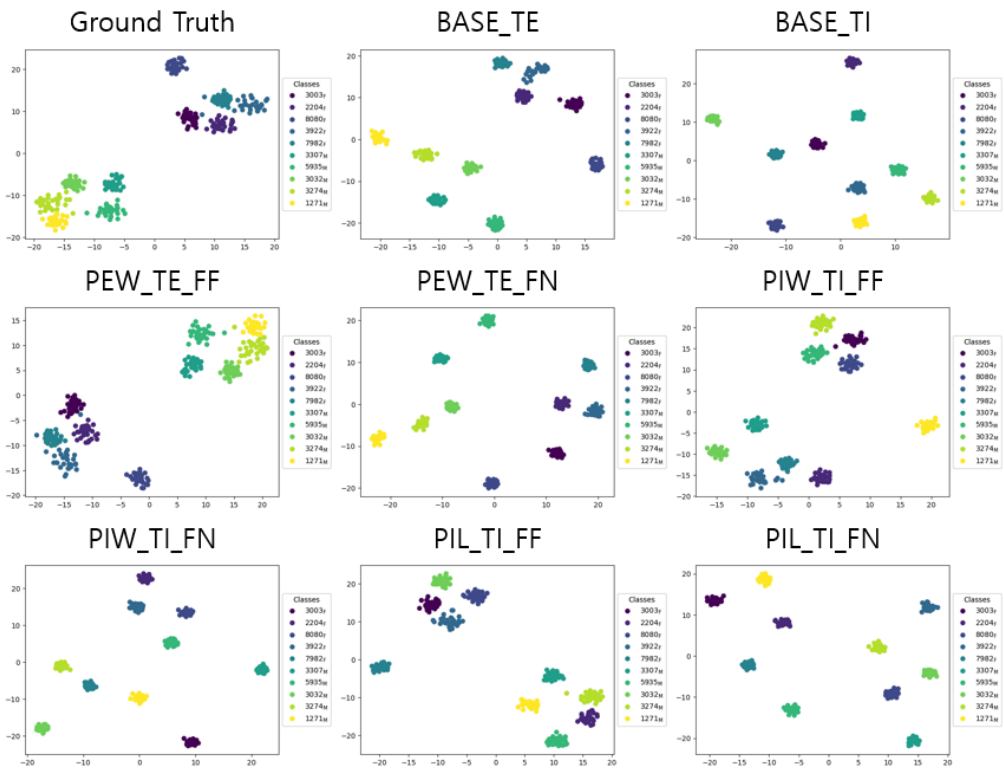

Figure 15: Visualization of the speaker embeddings with T-SNE

Chapter 6

Conclusion

We examined the effectiveness of pre-training a TTS model with multilingual untranscribed speech corpora when there is a small speech-text paired dataset of a target language. Concretely, first, we obtained pseudo phoneme sequences corresponding to each waveform that would play the role of text in the pre-training stage. We then pre-trained a TTS model, which had the normalizing flow module to exploit its invertibility, using multilingual speech data and pseudo phoneme sequences. Finally, we fine-tuned the model with the small paired dataset of the target language. Experimental results showed us that the model pre-trained with the multilingual dataset could learn faster and achieve better CER than the model not using the multilingual dataset, which confirmed the possibility that waveforms of foreign languages can be helpful in training a TTS model for a target language.

As our proposed method uses multilingual untranscribed speech corpora, it is meaningful in that people who are unfamiliar with the writing systems of those foreign languages can use our method. Furthermore, our method can use speech data from any language regardless of whether it uses logogram or phonogram, which makes it easy to procure a large dataset for pre-training. Another thing worth noting is that we use a multi-speaker and multilingual pre-trained model, which can be

adapted on any voice in any language. In natural language processing, models pre-trained with multilingual data, such as m-BERT [9], have been successfully used as starting points of models for downstream tasks in many languages. From this point of view, our method provides a possible direction for how to make a pre-trained model that can be used as a starting point for a TTS model for any voice in any language, which eliminates the need to train the TTS model from scratch.

However, although leveraging the multilingual dataset is more helpful than when not used, there is a limitation that the quality of a generated waveform is much lower than that of a waveform generated from a model trained with a lot of paired data. The waveforms generated from the models trained with our method sound rather noisy and robotic. Considering that there are models that generate almost perfect waveforms indistinguishable from human speech if sufficient paired data exist, improvements are needed to produce waveforms that are much more natural than now.

Since the method proposed in this thesis use pre-training, it is affected by how clearly the pre-trained model can reconstruct the foreign language waveform. Considering that a waveform was not so fully reconstructed at the pre-training stage, it would be possible to increase the model's expressivity by increasing the size of the model and the dataset. In addition, a model using a large multilingual dataset should be able to cover a variety of acoustic conditions because each waveform can be recorded in different environments. To this end, as attempted in [5], it would be possible to model elements that explicitly represent acoustic condition information. We leave these explorations for future work.

Bibliography

[1] A. Baevski, W.-N. Hsu, A. Conneau, and M. Auli. Unsupervised speech recognition. *Advances in Neural Information Processing Systems*, 34:27826–27839, 2021.

[2] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations. *Advances in Neural Information Processing Systems*, 33:12449–12460, 2020.

[3] E. Casanova, J. Weber, C. D. Shulby, A. C. Junior, E. Gölge, and M. A. Ponti. Yourtts: Towards zero-shot multi-speaker tts and zero-shot voice conversion for everyone. In *International Conference on Machine Learning*, pages 2709–2720. PMLR, 2022.

[4] J. Chen, C. Lu, B. Chenli, J. Zhu, and T. Tian. Vflow: More expressive generative flows with variational data augmentation. In *International Conference on Machine Learning*, pages 1660–1669. PMLR, 2020.

[5] M. Chen, X. Tan, B. Li, Y. Liu, T. Qin, S. Zhao, and T.-Y. Liu. Adaspeech: Adaptive text to speech for custom voice. *arXiv preprint arXiv:2103.00993*, 2021.

[6] H.-S. Choi, J. Lee, W. Kim, J. Lee, H. Heo, and K. Lee. Neural analysis and

synthesis: Reconstructing speech from self-supervised representations. *Advances in Neural Information Processing Systems*, 34:16251–16265, 2021.

[7] A. Conneau, A. Baevski, R. Collobert, A. Mohamed, and M. Auli. Unsupervised cross-lingual representation learning for speech recognition. *arXiv preprint arXiv:2006.13979*, 2020.

[8] B. Desplanques, J. Thienpondt, and K. Demuynck. Ecapa-tdnn: Emphasized channel attention, propagation and aggregation in tdnn based speaker verification. *arXiv preprint arXiv:2005.07143*, 2020.

[9] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[10] L. Dinh, D. Krueger, and Y. Bengio. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

[11] J. Ho, X. Chen, A. Srinivas, Y. Duan, and P. Abbeel. Flow++: Improving flow-based generative models with variational dequantization and architecture design. In *International Conference on Machine Learning*, pages 2722–2730. PMLR, 2019.

[12] M. Jeong, H. Kim, S. J. Cheon, B. J. Choi, and N. S. Kim. Diff-tts: A denoising diffusion model for text-to-speech. *arXiv preprint arXiv:2104.01409*, 2021.

[13] J. Johnson, M. Douze, and H. Jégou. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 7(3):535–547, 2019.

[14] J. Kim, S. Kim, J. Kong, and S. Yoon. Glow-tts: A generative flow for text-to-speech via monotonic alignment search. *Advances in Neural Information Processing Systems*, 33:8067–8077, 2020.

[15] J. Kim, J. Kong, and J. Son. Conditional variational autoencoder with adversarial learning for end-to-end text-to-speech. In *International Conference on Machine Learning*, pages 5530–5540. PMLR, 2021.

[16] M. Kim, M. Jeong, B. J. Choi, S. Ahn, J. Y. Lee, and N. S. Kim. Transfer learning framework for low-resource text-to-speech using a large-scale unlabeled speech corpus. *arXiv preprint arXiv:2203.15447*, 2022.

[17] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[18] D. P. Kingma and P. Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. *Advances in neural information processing systems*, 31, 2018.

[19] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[20] J. Kong, J. Kim, and J. Bae. Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis. *Advances in Neural Information Processing Systems*, 33:17022–17033, 2020.

[21] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. Paul Smolley. Least squares generative adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2794–2802, 2017.

[22] V. Popov, I. Vovk, V. Gogoryan, T. Sadekova, and M. Kudinov. Grad-tts: A diffusion probabilistic model for text-to-speech. In *International Conference on Machine Learning*, pages 8599–8608. PMLR, 2021.

[23] V. Pratap, Q. Xu, A. Sriram, G. Synnaeve, and R. Collobert. Mls: A large-scale multilingual dataset for speech research. *arXiv preprint arXiv:2012.03411*, 2020.

[24] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever. Robust speech recognition via large-scale weak supervision. *OpenAI Blog*, 2022.

[25] 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. *arXiv preprint arXiv:2006.04558*, 2020.

[26] Y. Ren, Y. Ruan, X. Tan, T. Qin, S. Zhao, Z. Zhao, and T.-Y. Liu. Fastspeech: Fast, robust and controllable text to speech. *Advances in Neural Information Processing Systems*, 32, 2019.

[27] J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, R. Skerrv-Ryan, et al. Natural tts synthesis by conditioning wavenet on mel spectrogram predictions. In *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 4779–4783. IEEE, 2018.

[28] Y. Wu, X. Tan, B. Li, L. He, S. Zhao, R. Song, T. Qin, and T.-Y. Liu. Adaspeech 4: Adaptive text to speech in zero-shot scenarios. *arXiv preprint arXiv:2204.00436*, 2022.

[29] Y. Yan, X. Tan, B. Li, T. Qin, S. Zhao, Y. Shen, and T.-Y. Liu. Adaspeech 2: Adaptive text to speech with untranscribed data. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6613–6617. IEEE, 2021.

[30] Y. Yan, X. Tan, B. Li, G. Zhang, T. Qin, S. Zhao, Y. Shen, W.-Q. Zhang, and T.-Y. Liu. Adaspeech 3: Adaptive text to speech for spontaneous style. *arXiv preprint arXiv:2107.02530*, 2021.

[31] H. Zen, V. Dang, R. Clark, Y. Zhang, R. J. Weiss, Y. Jia, Z. Chen, and Y. Wu. Libritts: A corpus derived from librispeech for text-to-speech. *arXiv preprint arXiv:1904.02882*, 2019.

Appendix

A. K-means clustering results

In the PEW and PIW settings, we performed k-means clustering on the features of the whole speech corpora. Figure 16 shows us the frequency of each cluster in the PEW and PIW settings. The horizontal axis represents the id of each cluster, and the vertical axis represents how many features are mapped into each cluster. We can notice that the features are quite evenly distributed across the clusters, which means 128 is a fairly appropriate number for clustering. In the PIL setting, k-means clustering was performed for each language's corpus. Figure 17, Figure 18, Figure 19, Figure 20, Figure 21, Figure 22, and Figure 23 show similar results.

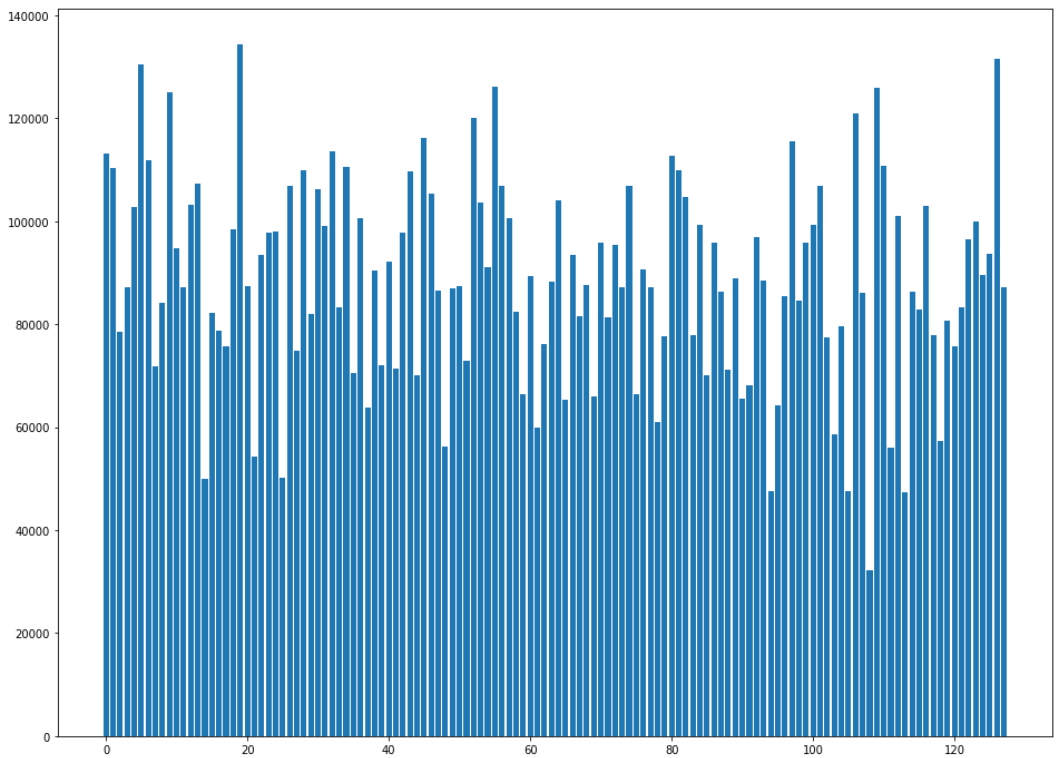

Figure 16: Frequency of each cluster in the PEW and PIW setting

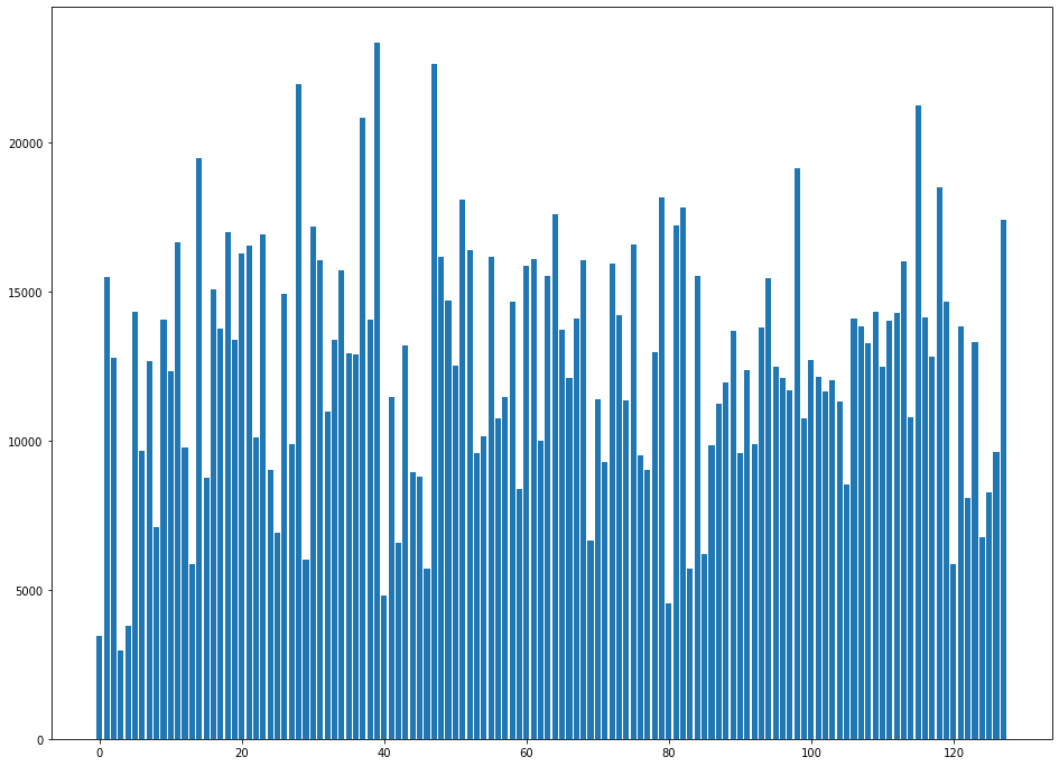

Figure 17: Frequency of each cluster with German corpus in the PIL setting

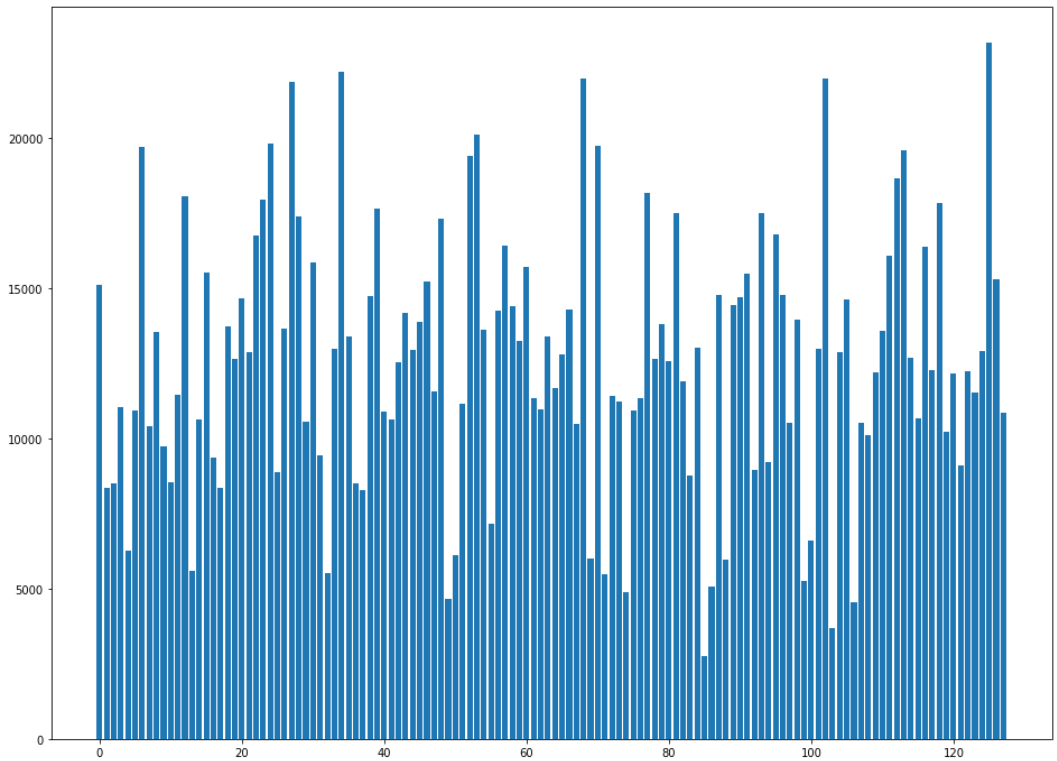

Figure 18: Frequency of each cluster with Dutch corpus in the PIL setting

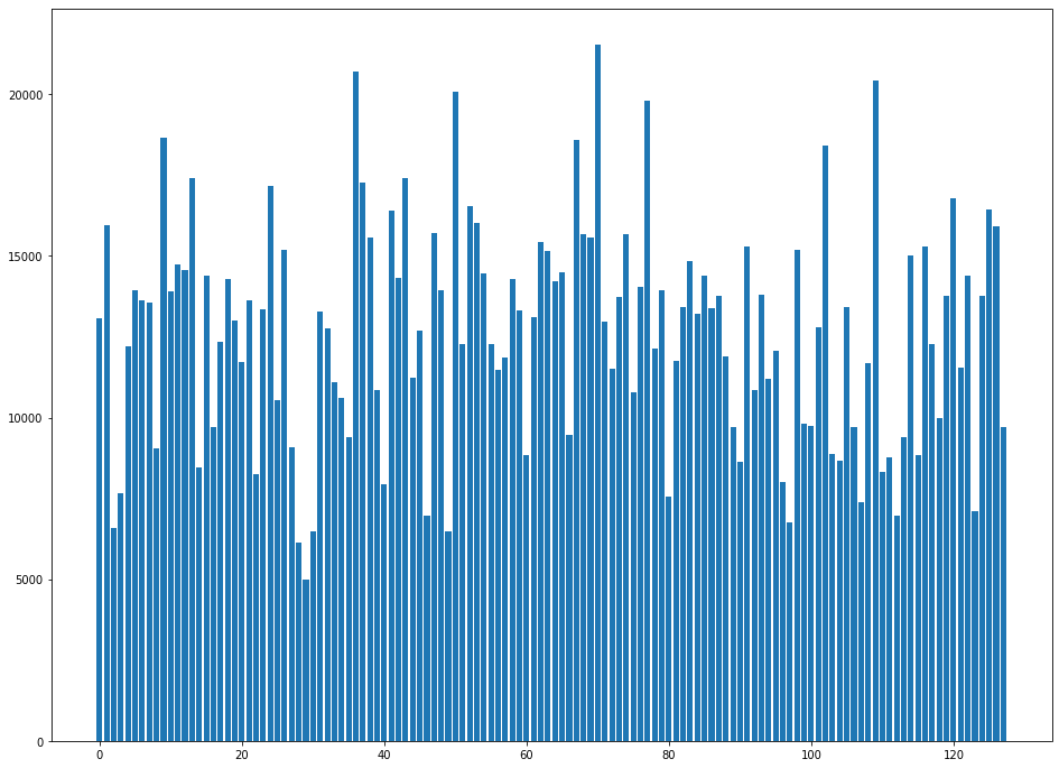

Figure 19: Frequency of each cluster with French corpus in the PIL setting

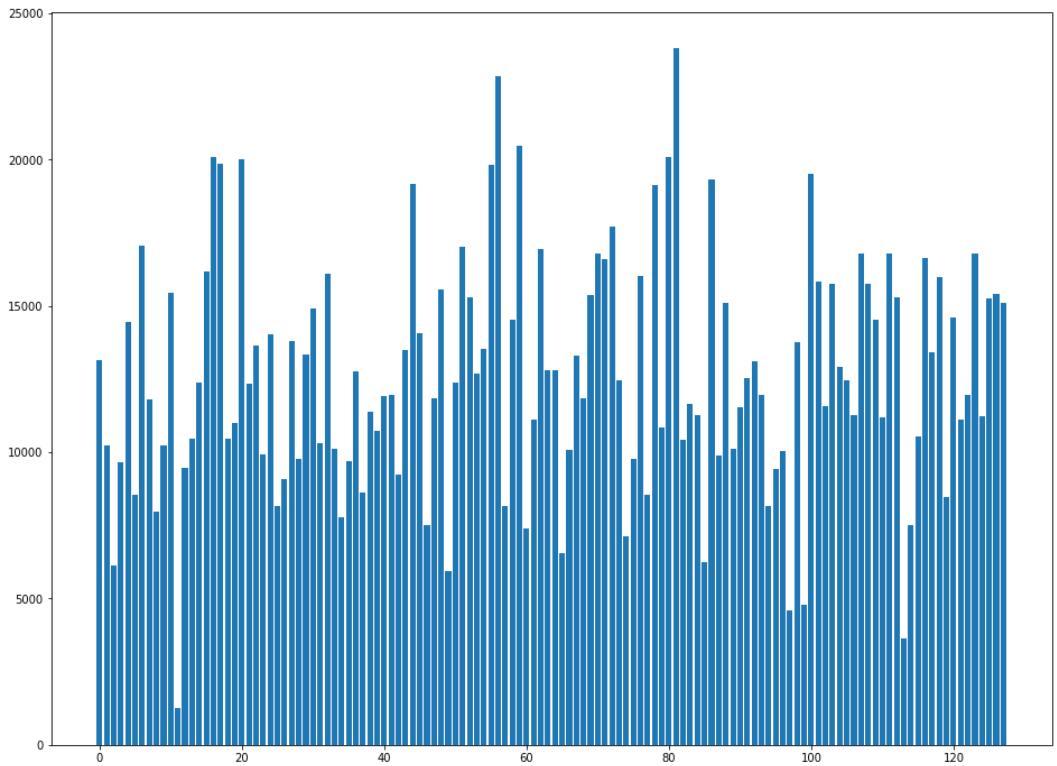

Figure 20: Frequency of each cluster with Spanish corpus in the PIL setting

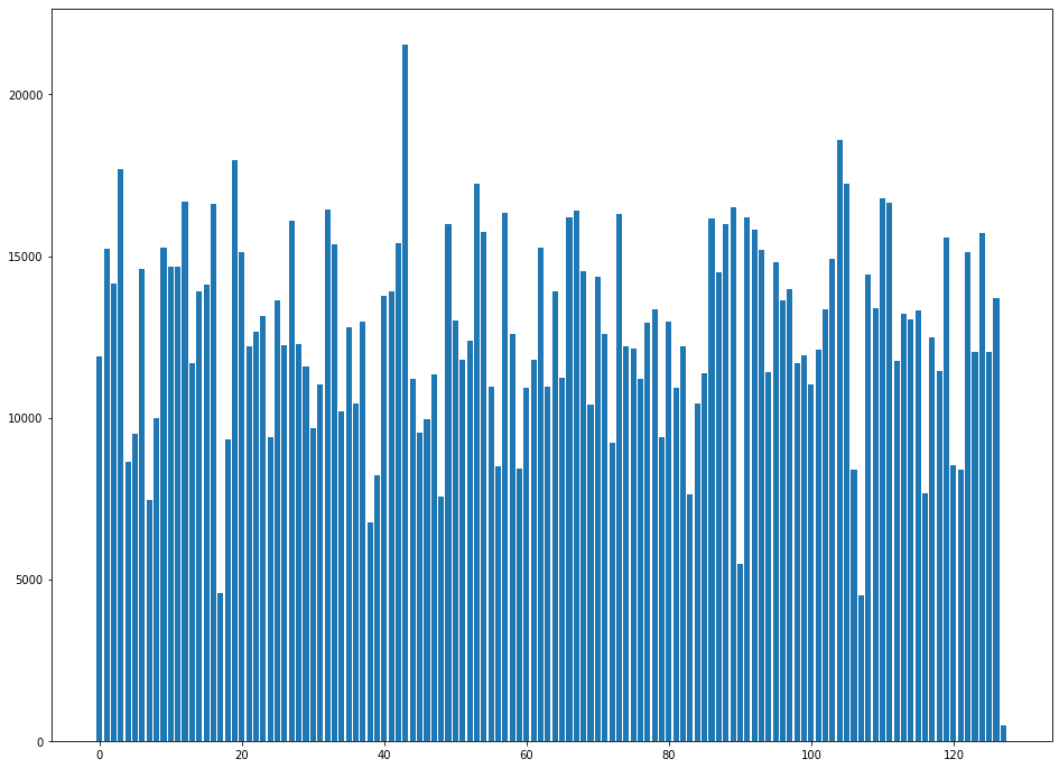

Figure 21: Frequency of each cluster with Italian corpus in the PIL setting

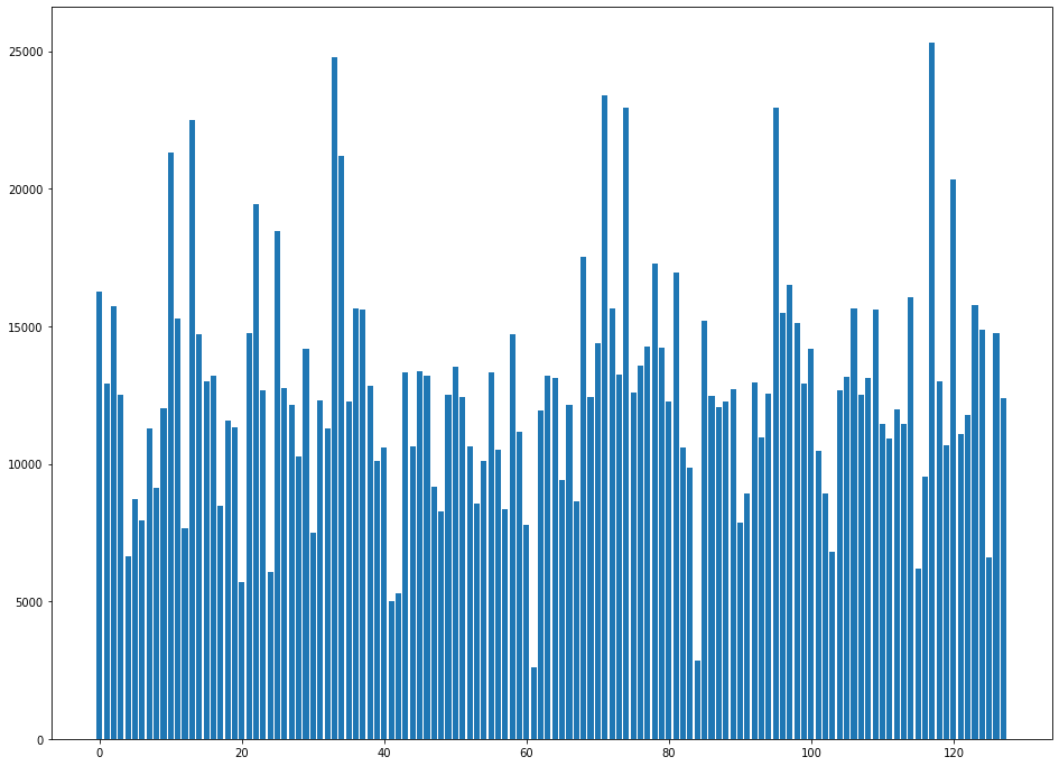

Figure 22: Frequency of each cluster with Portuguese corpus in the PIL setting

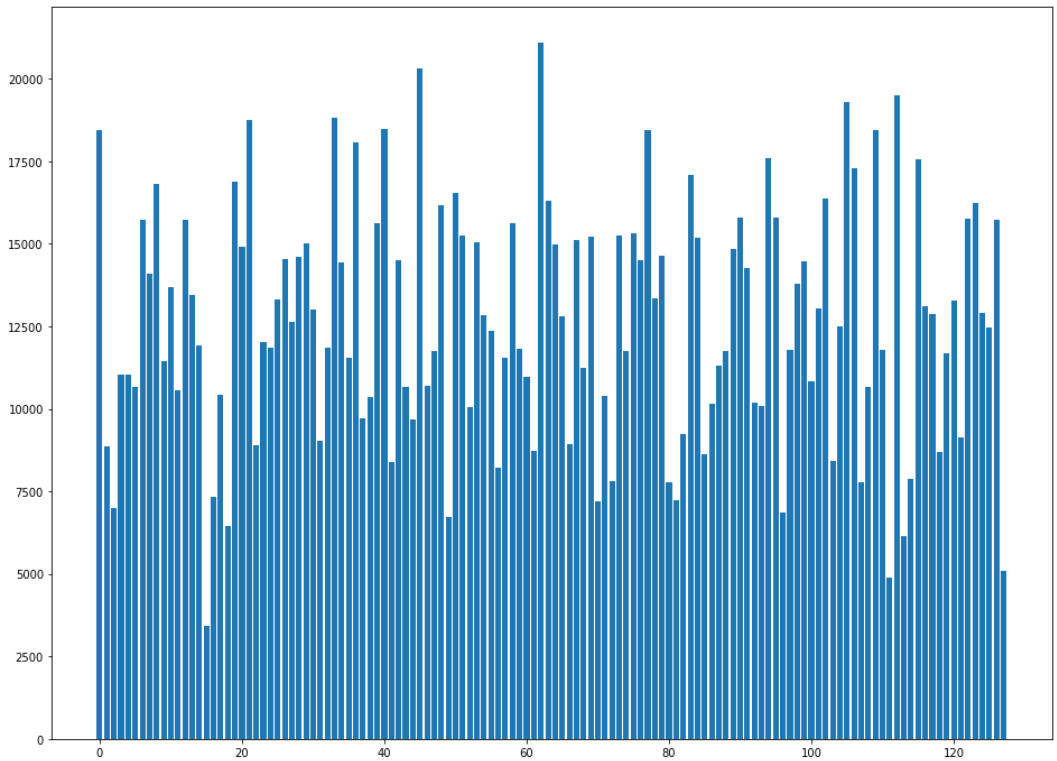

Figure 23: Frequency of each cluster with Polish corpus in the PIL setting

국문초록

음성은 매우 효과적인 정보전달 수단으로 굉장히 많은 곳에서 사용되고 있으며, 음성 합성 기술은 사람이 직접 음성을 녹음할 필요를 줄여주어 생산성의 향상과 비용 절감 효과를 가져왔다. 하지만 현재 대다수의 음성합성 모델을 학습시키기 위해서는 많은 양의 음성-텍스트 데이터가 필요하고, 이는 텍스트로 전사된 데이터가 적은 언어권에서 음성합성 모델을 만드는 것을 어렵게 한다. 본 연구에서는 언어가 다르더라도 발화되는 음성들은 비슷한 발음을 공유하는 경우가 있다는 것에 주목하여, 비교적 확보하기 더 용이한 전사되지 않은 외국어 음성 데이터셋을 음성합성 모델을 학습하는 데에 활용하는 방안을 제안한다. 구체적으로는 먼저 외국어 음성 데이터셋에 wav2vec 2.0 XLSR-53 모델을 적용해 발음정보를 담은 벡터들을 추출하고, 이 벡터들을 대상으로 k-means 클러스터링을 진행해 텍스트 역할을 수행할 슈도 레이블을 구한다. 그 다음 외국어 음성-슈도 레이블을 활용해 음성합성 모델을 사전 학습하고, 마지막으로 다시 원래 학습하고자 했던 타겟 언어의 음성-텍스트 데이터를 활용해 모델을 추가로 학습한다. 실험을 통해서 외국어 음성 데이터를 활용해 사전 학습된 모델이 더 빠르게 더 낮은 CER 값을 달성하는 것을 확인함으로써, 음성으로만 구성된 외국어 데이터셋이 음성합성 모델을 학습하는 데에 도움이 될 수 있다는 것을 확인하였다.

주요어: 음성합성, 전사되지 않은 외국어 음성 데이터, 클러스터링, 슈도 레이블, 사전 학습, 파인 튜닝

학번: 2021-24409