End-to-End Mispronunciation Detection with Simulated Error Distance

Zhan Zhang, Yuehai Wang, Jianyi Yang

Department of Information and Electronic Engineering, Zhejiang University, China
zhan_zhang@zju.edu.cn, wyuehai@zju.edu.cn, yangjy@zju.edu.cn

Abstract

With the development of deep learning, the performance of the mispronunciation detection model has improved greatly. However, the annotation for mispronunciation is quite expensive as it requires the experts to carefully judge the error for each pronounced phoneme. As a result, the supervised end-to-end mispronunciation detection model faces the problem of data shortage. Although the text-based data augmentation can partially alleviate this problem, we analyze that it only simulates the categorical phoneme error. Such a simulation is inefficient for the real situation. In this paper, we propose a novel unit-based data augmentation method. Our method converts the continuous audio signal into the robust audio vector and then into the discrete unit sequence. By modifying this unit sequence, we generate a more reasonable mispronunciation and can get the vector distance as the error indicator. By training on such simulated data, the experiments on L2Arctic show that our method can improve the performance of the mispronunciation detection task compared with the text-based method.

Index Terms: mispronunciation detection, second language learning, speech recognition

1. Introduction

With the trend of globalization, more and more people need to learn a second language (L2). To offer a flexible education service for these learners, Computer-Assisted Pronunciation Training (CAPT) technologies have been studied extensively in recent years.

A typical CAPT system requires the learner to read a target text, and the system detects mispronunciations of the speech for feedback. The learner can adjust the pronunciation accordingly to improve oral skills. Early CAPT papers [1–3] utilize the Hidden Markov model (HMM) to align the phonemes and then use the posterior probability to judge the correctness. With the improvement of ASR technologies, recognition-based methods [4–6] have also been applied to mispronunciation detection. These methods recognize the pronounced phonemes and align them to the target phonemes to find out the mispronounced ones. However, such methods only consider the ASR task, and the target text is used only for alignment during inference. As a result, the detection task is not directly optimized. To solve this problem, end-to-end mispronunciation detection methods [7–9] propose to use the target text for input and predict the mispronunciation state directly. These methods generally achieve a better performance compared with the ASR-based methods.

Despite the success of end-to-end methods, one main deficiency is that such methods require human-annotated training data. Unfortunately, annotating the mispronunciation is quite hard. The experts need to judge the correctness of each pronounced phoneme carefully. As a result, the dataset for mispronunciation detection is quite scarce. For example, compared with the hundred-hour level Librispeech [10] (a common ASR dataset), the annotated part of L2Arctic [11] (a CAPT dataset) is only a few hours. Thus, data augmentation methods are needed to generate artificial mispronunciations so that the mispronunciation detection model will not overfit.

So far, however, there has been little work that focuses on the data augmentation method for mispronunciation detection. Existing methods generally modify the target phoneme for simulating the mispronunciation. For example, [12] adds the negative samples for a certain phoneme to train the supervised support vector regressor (SVR). [9, 13] analyzes the typical mispronunciation when learning the second language, and designs the phoneme replacement rule to simulate mispronunciations. In summary, these data augmentation methods are text-based. They choose to modify the original text so that the speech and the text does not match each other. For example, the speech of the word “cat” (phonemes are “K AE T”) can be a mispronunciation for the target “sat” by replacing the phoneme “K” with “S”. Here the raw utterance that pronounces “cat” generally comes from the accessible ASR dataset by native speakers (because we cannot view the non-native speech as a standard pronunciation for the transcription).

Although such a text-based data augmentation can partially simulate mispronunciations, it has the following deficiencies. First, the text-based method only generates the categorical phoneme error. This assumes that mispronunciations can be represented by another phoneme in the target language. However, according to the L2 speech learning theory [14], the mispronunciation generally takes place between similar position-sensitive allophones rather than at the more abstract phoneme level. Meanwhile, as analyzed by [15], the pronunciation of second language learners is affected by their native language. The mispronunciation may not be accurately categorized into the phoneme set of the target language [16, 17]. Second, the
text-based data augmentation only uses the existing native utterances for phoneme replacement but cannot generate non-native speech features for training. However, for the inference of the detection model, the input is the non-native speech from the language learners. This conflict makes this augmentation method less efficient.

To solve these problems, we propose a data augmentation that can simulate a more realistic mispronunciation for the end-to-end models. We utilize the unsupervised Wav2Vec2 model [18] and the k-Means clustering model [19] to convert the original continuous speech into audio vectors and then into discrete acoustic units. As the training process for these two models does not need text annotations, these two models can be trained on both native and non-native utterances. Thus, the unit can better model the accentuated pronunciation compared with the former phoneme-based method. After converting the continuous speech into discrete units, we can easily replace the raw unit with another unit to simulate mispronunciations. Moreover, we can get the vector distance between the original unit and the replaced one as the error indicator. By using the proposed data augmentation method, we show that the detection performance is improved under different settings of the trainset.

2. Method

2.1. Text-based data augmentation

Before describing our method, we first introduce the end-to-end mispronunciation detection model and the text-based data augmentation in detail. The workflow is shown in Fig.1.

End-to-end mispronunciation detection. We denote an annotated mispronunciation as the triplet $A = \{a_1,...,a_{26}\}$, where the audio feature is $A = \{a_1,...,a_{26}\}$, the target phoneme is $P_{tgt} = \{T_{tgt}^1,...,T_{tgt}^L\}$, and the state error is $E = \{e_1,...,e_L\}$. The length of the audio and phoneme sequence is $T$ and $L$, correspondingly. The annotator may also label the canonically pronounced phoneme of the speech. We denote it as $P$. We use the Transformer model [20] for this task. The end-to-end mispronunciation detection model uses the audio feature $A$ for the encoder input and the target phoneme $P_{tgt}$ for the decoder input. The model directly outputs the binary error state $E$ for mispronunciation detection. The binary cross-entropy (BCE) loss is applied for training.

$$l_e = BCE(\hat{E}, E),$$

If the pronounced phoneme $P$ is available, the model also predicts the pronounced phoneme as the auxiliary ASR task to avoid overfitting. Cross-entropy (CE) loss is applied.

$$l_{asr} = CE(\hat{P}, P).$$

The loss function for the whole model is

$$l = l_e + l_{asr}$$

Text-based data augmentation. The text-based data augmentation method starts from the accessible ASR dataset of native speakers. For a native utterance, we can view the pair of the phoneme annotation $P$ and the audio $A$ as correct pronunciations (i.e., for the triplet $(A, P_{tgt}, E)$, $P_{tgt} = P$, and $E$ is all 0). The text-based data augmentation method modifies part of the original phoneme $P$ and sets the modified version as the new target phoneme. Thus, the original speech can be a mispronunciation for the new target phoneme. The error state for the modified phoneme is set as 1. A sample utterance that pronounces “cat” is also shown in Fig.1.

![Figure 2: We convert the continuous audio signal into audio vectors and then into the discrete audio unit sequence $U$ for further processing. We use the clustered centroid to represent each unit. By replacing $u$ with $u'$, we obtain the vector distance $d$ as the error indicator. For example, we replace the original unit indexed 26 with the new 77. Thus, the first unit is mispronounced. The vector distance 3.12 reflects the error degree.](image)

![Figure 3: The proposed method and the unit-based data augmentation. We use the discrete audio unit for the encoder input. For the same native utterance that pronounces “cat”, we modify its original audio unit to simulate mispronunciation for the phoneme “c”.](image)
Such a simulation is more reasonable compared with the former text-based method.

Unit replacement. After converting the continuous audio signal into \( U \), we randomly replace part of \( U \) to obtain the new unit sequence \( U^r = \{ u^r_1, ..., u^r_T \} \) and the distance sequence \( D = \{ d_1, ..., d_T \} \) for simulating the mispronunciation.1 A sample is shown in Fig.2. The centroid distance \( d \) between \( u^r \) and \( u \) in the vector subspace (we use the Euler distance) reflects their similarity. A bigger distance suggests that the new \( u^r \) deviates more from the original pronunciation and thus the original phone. As a result, this distance reflects the error degree.

For simplicity, we call it “the error distance” in the following discussion.

As discussed in the introduction part, the mispronunciation generally takes place between similar pronunciations. In other words, for each original unit \( u \), the other unit that has a smaller distance will be more likely to be chosen for replacement. To achieve such a target, we first sort the \( k - 1 \) units by their distance to \( u \) (in the ascending order). Then, we choose the unit that ranked the \( j \)th for replacement. Formally, we denote the sorted unit sequence as \( S^u = \{ s^u_1, ..., s^u_{k-1} \} \) and the corresponding distance as \( S^d = \{ s^d_1, ..., s^d_{k-1} \} \). Then \( u^r \) and \( d \) are

\[
\begin{align*}
    &u^r = s^u_j \\
    &d = s^d_j.
\end{align*}
\]

We use a norm distribution to decide \( j \).

\[
j = \min(k - 1, \left\lfloor \frac{(k - 1)|m|}{3} \right\rfloor),
\]

where \( m \sim N(0, \sigma^2) \). Here \( \sigma \) controls the diversity of \( u^r \). A low \( \sigma \) means \( u^r \) is more likely to be chosen from the units that have low distance of \( u \). A higher \( \sigma \) means more units that are far from \( u \) will be chosen.

Training with the error distance. The workflow for using the unit-based data augmentation method is shown in Fig.3. To use the discrete unit for the encoder input, we use an embedding layer to map the unit to its corresponding audio vector (i.e., the audio vector of the centroid). Note that the distance sequence \( D = \{ d_1, ..., d_T \} \) is at the unit level. To get the phoneme-level error distance, we first find the alignment between the unit and the phoneme by using the force alignment tool MFA [22]. Then, we average the corresponding sub-sequence of \( D \) for each phoneme to get \( D^p = \{ d^p_1, ..., d^p_T \} \). For our method, we force the model to predict the error distance by using the mean squared error (MSE) loss.

\[
l_{ed} = MSE(\hat{D}^r, D^p)
\]

We should note that the proposed error distance \( D^p \) is more expressive than the binary error state \( E \) used by the text-based method. For Eq.1, \( l_e \) only requires the model to judge whether the speech matches the target phoneme or not. In contrast, \( l_{ed} \) forces the model to output the exact distance so that the model needs to discriminate different degrees of mispronunciation.

Finally, we combine the unit-based data augmentation and the former ASR task for training.

\[
l = l_{ed} + l_{asr}.
\]

1We also trained a Transformer-based text-to-speech (TTS) model [21] to demonstrate the effect of unit replacement. We set the discrete unit as the input “text”. Audio samples are available in https://zju-zhan-zhang.github.io/e2e-sed/

When calculating \( l_{ed} \), we apply the unit-based data augmentation to replace \( U \). When calculating \( l_{asr} \), we leave the original unit untouched.

3. Experiments

We use two datasets in our experiments. The first is a native ASR dataset called Librispeech [10]. The second is the accented dataset L2Arctic [11] for mispronunciation detection. L2Arctic contains 3599 annotated utterances that have the phoneme-level mispronunciation label. The other 23268 utterances are not annotated. We split the annotated utterances into 8:1:1 as the train-set, val-set, and test-set for experiments.

3.1. Audio to unit

We use the pretrained Wav2Vec2-XLS-R model [23] to extract the audio vector. This model is trained on multiple languages, and thus it can extract robust audio features even for the non-native speech from language learners. The input for Wav2Vec2 is the 16kHz raw waveform, and we use the output of the 8th layer as the extracted audio vector. We conduct the ABX test on the valset of L2Arctic to decide the hyper-parameter \( n \).

The ABX score is the metric for testing the discriminability of the unsupervised feature2. Given the feature \( X \) and the feature \( A \) of the same allophone, if the discriminability of such a feature is good, the distance between \( X \) and \( A \) should be smaller than between \( X \) and the other feature \( B \) that belongs to the different allophone. Here the ABX test is divided into two situations. For the within-speaker task, \( A \) and \( B \) belong to the same speaker. For the across-speaker task, \( A \) and \( B \) belong to the same speaker, while \( X \) belongs to another speaker. We show the performance of the 80-dim Fbank feature for comparison. The results suggest the extracted audio vector is robust for discriminating the allophone, and we choose \( n = 8 \) as it gives the best result.

To ensure the units can present both native and non-native pronunciations, we train the k-Means model on the audio vectors of Librispeech and the unlabeled part of L2Arctic. We conduct the phoneme recognition task to choose the cluster number \( k \). We use the Transformer-based ASR backbone [26] to test the phoneme error rate (PER) on Librispeech. For the Transformer, we use 12 encoder layers and 12 decoder layers in our experiments. We set the attention dimension to \( d_{mod} = 512 \), attention heads to \( h = 4 \), and the feed-forward dimension to \( d_{ff} = 1024 \). For the converted unit sequence \( U/k = 4096 \) can get a similar PER performance on Librispeech test-clean compared with using the 80-dim Fbank feature and thus is chosen for the experiments.

To visualize these \( k \) units, we plot the co-occurrence between the phonemes and the units as suggested in [18]. The re-

<table>
<thead>
<tr>
<th>Feature</th>
<th>Within</th>
<th>Across</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fbank</td>
<td>19.58</td>
<td>28.43</td>
<td>24.01</td>
</tr>
<tr>
<td>XLS-R (n = 6)</td>
<td>12.31</td>
<td>11.24</td>
<td>11.77</td>
</tr>
<tr>
<td>XLS-R (n = 8)</td>
<td>11.93</td>
<td>10.76</td>
<td>11.35</td>
</tr>
<tr>
<td>XLS-R (n = 10)</td>
<td>13.00</td>
<td>11.16</td>
<td>12.08</td>
</tr>
<tr>
<td>XLS-R (n = 12)</td>
<td>13.36</td>
<td>11.19</td>
<td>12.28</td>
</tr>
<tr>
<td>XLS-R (n = 14)</td>
<td>13.24</td>
<td>11.11</td>
<td>12.18</td>
</tr>
</tbody>
</table>

When calculating \( l_{asr} \), we leave the original unit untouched.

Table 1: ABX scores on the valset
result is shown in Fig.4. We can see that these units can generally represent each phoneme. Their co-occurrences are also related to the pronunciation. For example, the units that have high co-occurrence for the phoneme “S” also appear for the phoneme “Z”, suggesting their pronunciations are similar. Such a phenomenon can also be observed especially for the vowels.

### 3.2. Mispronunciation Detection

We use the hierarchical evaluation structure defined in [27] to divide the detection results into true acceptance (TA), false rejection (FR), false acceptance (FA), and true rejection (TR). As the model should make a good balance of detecting the wrong pronunciations and accepting the correct ones, we use the F1 score as the main metric.

For mispronunciation detection, we compare the following methods.

- The ASR-based method. We use Librispeech for pretraining on the ASR task. Then, we fine-tune the model to recognize the pronounced phoneme on the trainset of L2Arctic. For inference, the misalignment between the recognized phoneme and the target phoneme is judged as an error.
- The pure end-to-end method that does not use data augmentation. We still use Librispeech for the ASR pretraining task (Eq.2). Then we fine-tune the model on L2Arctic using Eq.3 to predict the error state directly.
- The end-to-end method that uses the text-based data augmentation. We use Librispeech for simulating the mispronunciations. The loss function is Eq.3. Then we fine-tune the model on L2Arctic (still using Eq.3).
- The proposed method. We use Librispeech for simulating the mispronunciations and use Eq.8 to train the model. Then we fine-tune the model on L2Arctic using Eq.3. Note that the output layer for predicting $\tilde{D}_p$ is fine-tuned to predict $\hat{E}$.

For the ASR-based methods, we use the CRNN-CTC model [4] and the attention-based Transformer model (the structure is the same as the aforementioned Transformer for phoneme recognition) for experiments. The other end-to-end models also use the same Transformer setting. To predict $\hat{P}$, we append a linear layer after the decoder. To predict $\hat{E}$ or $\tilde{D}_p$, we append a convolutional module. This module is constructed by two stacked Conv1D (kernel size is 3, stride is 1, and ReLU as the activation function) layers.

We use the warm-up learning schedule for the pretraining task. Next, we fix the learning rate as $\text{lr} = 10^{-4}$ for fine-tuning. For the end-to-end methods, we set the threshold as 0.5 to convert $\hat{E}$ into the binary error state. When adding the text-based data augmentation, we randomly replace 20% of the original phoneme $P$. For our method, we randomly replace 20% of the original unit $U$. We also decide the hyper-parameter $\sigma$ by choosing the model that has the highest F1 on the valset. The results are shown in Tab.2. We choose $\sigma = 0.5$ for our method.

Finally, to test the performance of the data augmentation methods, we use different fractions of the full L2Arctic trainset to fine-tune the model. We show the F1 scores in Tab.3. Note that the annotation is not compulsory for the ASR-based methods. The corresponding F1 scores that do not use annotations are shown in the 0% part. As we can see from Tab.3, due to data shortage, the performance for the pure end-to-end method is worse than the ASR-based methods for the 20% and 40% cases. When using more data for training, the F1 score for the end-to-end method improves. Adding the data augmentation also solves this data-shortage problem. The results show that our method can get a better detection performance compared with the text-based data augmentation (TextAug).

### 4. Conclusions

In this paper, we propose a novel data augmentation method to train the end-to-end mispronunciation detection model. By modifying the converted unit sequence, we generate a more reasonable mispronunciation and can get the vector distance as the error indicator. By training on such simulated data, the experiments on L2Arctic show that our method can improve the performance of the mispronunciation detection task compared with the text-based method.
5. References


