Non-Autoregressive Chinese ASR Error Correction with Phonological Training

Zheng Fang[1,2], Ruiqing Zhang[3]*, Zhongjun He[3], Hua Wu[3], Yanan Cao[1,2]
[1]Institute of Information Engineering, Chinese Academy of Sciences
[2]School of Cyber Security, University of Chinese Academy of Sciences
[3]Baidu Inc. No. 10, Shangdi 10th Street, Beijing, 100085, China
[1,2]{fangzheng, caoyanan}@iie.ac.cn
[3]{zhangruiqing01, hezhongjun, wu_hua}@baidu.com

Abstract

Automatic Speech Recognition (ASR) is an efficient and widely used input method that transcribes speech signals into text. As the errors introduced by ASR systems will impair the performance of downstream tasks, we introduce a post-processing error correction method, PhVEC, to correct errors in text space. For the errors in ASR result, existing works mainly focus on fixed-length corrections, modifying each wrong token to a correct one (one-to-one correction), but rarely consider the *variable-length* correction (one-to-many or many-to-one correction). In this paper, we propose an efficient non-autoregressive (NAR) method for Chinese ASR error correction for both cases. Instead of conventionally predicting the sentence length in NAR methods, we propose a novel approach that uses phonological tokens to extend the source sentence for *variable-length* correction, enabling our model to generate phonetically similar corrections. Experimental results on datasets of different domains show that our method achieves significant improvement in word error rate reduction and speeds up the inference by 6.2 times compared with the autoregressive model.

1 Introduction

Errors introduced by automatic speech recognition (ASR) usually affect the performance of downstream tasks such as phonetic search, speech translation, etc. In recent years, ASR error correction techniques have been proposed (Anantaram et al., 2018; Mani et al., 2020; Zhao et al., 2021; Leng et al., 2021) to refine the ASR output and correct errors in text space. Without loss of generality, we study Chinese ASR error correction in this paper.

Given the ASR result of an utterance, the goal of error correction is to generate a sentence with the wrongly recognized words corrected. Thus the ASR error correction can be modeled as a machine translation problem under conventional autoregressive sequence-to-sequence (Seq2Seq) framework (Guo et al., 2019; Hrinchuk et al., 2020; Mani et al., 2020). However, the autoregressive models suffer from inefficient decoding since the generation of each target token depends on previously generated characters (Figure 1 (a)). Furthermore, without considering the phonetic similarities, the method is prone to generate corrections with totally different pronunciation, as for the example, the error character ("表", *biao*) should be corrected into ("不要", *bu yao*) but the Seq2Seq model ignores phonological features and corrects it to a phonetically different correction ("不许", *bu xu*).

To speed up prediction, recent studies propose to take non-autoregressive (NAR) methods (Gu et al., 2018; Ren et al., 2020) for error correction, which generates target tokens in parallel. Most NAR approaches make fixed-length predictions that generate same-length output as the source input by directly tagging on the source text (Zhang et al., 2020). Some works further leverage phonological features to correct the ASR errors caused by similar pronounced characters (Zhang et al., 2021a; Cheng et al., 2020). Such methods have made great improvements in correcting simple errors, but **cannot** handle samples with different lengths of source and target, referred to as *variable-length error correction*. See Figure 1 (b) for illustration. The method successfully finds out the erroneous character ("表", *biao*), and substitutes it with its phonological feature "*biao*" for error correction. However, it eventually generates a wrong correction ("标", *biao*) under the constraint of fixed length prediction.

To overcome above constraint while preserving efficient prediction, NAR solutions for *variable-length* prediction are proposed (Leng et al., 2021; Gu et al., 2019). They first build a *length predictor* to estimate fertility, i.e., the number of target tokens

* This work was conducted at Baidu. Corresponding author: Ruiqing Zhang.

Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 5907 - 5917

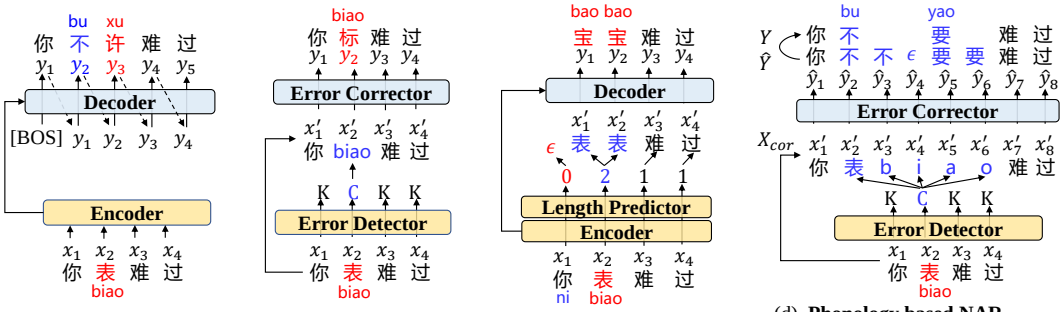


Figure 1: An example of Chinese ASR Error Correction. The source speech "*ni bu yao nan guo* " ("Don't be sad") is incorrectly recognized to "*ni biao nan guo* " ("Your watch is sad"). (a) The Seq2Seq model generates a fluent but incorrect sentence because it neglects the phonetic similarity. (b) The sequence tagging model with phonological features correctly detects the character with error ("K" means keep unchanged and "C" means to be correct), but because it only supports equal-length prediction, it fails to generate a fluent correction with equal length as the input. (c) The NAR model with a length predictor incorrectly predicts the number of target tokens corresponding to the source character "*ni*" as 0, which should be 1. Accordingly, it removes the character and generates an incorrect result. (d) The proposed PhVEC first detects problematic tokens, then leverages their phonological features to generate a *variable-length* intermediate result, and outputs the final result by reducing the repeated characters.

aligned to each source token, then up-sampling or dropping some source tokens accordingly for parallel correction. However, the method is fragile because incorrect length prediction will distort the meaning of the sentence and may directly lead to false prediction. See Figure 1 (c) for example, the length predictor produces a wrong alignment (0 2 1 1), indicating that the decoder will drop the first source character and generate two target characters for the second source character. According to the alignment, the intermediate result turns to be a meaningless sentence ("表表难过", means "watch watch sad") and leads to a wrong correction.

To address the issues mentioned above, we propose a novel NAR model named PhVEC (Phonology-based Variable-length Error Correction). The model incorporates the phonological features of error characters to enable *variable-length* prediction. Concretely, PhVEC contains a detection network and a correction network based on a pre-trained language model (Devlin et al., 2019; Sun et al., 2020). The detection network predicts the correctness of each token, and the correction network generates correction result. Instead of using a length predictor to predict the fertility for each source token, we insert the phonological tokens (Chinese Pinyin[1]) to the source sequence as placeholders after each detected erroneous character. During prediction, each source token (either the character token or pinyin token) can generate zero or one target token. With the guidance of the phonological token, the model will generate characters with similar pronunciations. As Figure 1 (d) shows, the pinyin token "*b*" produces target character ("不", *bu*) with the same consonant letter, and "*a o*" produces target character ("要", *yao*) with the same vowel letter. We delete the repeated characters in the final sequence.

We evaluate our methods on multiple datasets with varying degrees of ASR word error rates. Experimental results show that our PhVEC obtains significant improvement (over 10% word error rate reduction) on standard benchmarks compared with existing NAR-based ASR error correction methods at comparable speed. Even compared with the AR baseline Transformer and BART (Lewis et al., 2020) models, PhVEC can still have an 8.55% and 2.39% word error rate reduction while keeping a 6.2x speed-up.

2 Method

Chinese ASR error correction can be formalized as the following task. Given a speech recognition sequence $X = (x_1, x_2, ..., x_n)$ of n Chinese characters, the goal is to correct it into another sequence of m characters $Y = (y_1, y_2, ..., y_m)$. Note that the target sequence length m does not have to be equal to the source sequence length n. There exist three types of ASR errors in transforming from X to Y: substitution, deletion, and insertion.

As illustrated in Figure 1(d), our proposed ASR

[1]In this paper, we ignore the tone of pinyin, and the pinyin of each Chinese character can be represented by one to six English letters.

error correction model is composed of an error detector and an error corrector, both perform a NAR tagging. The error detector takes X as input to predict the correctness of each token. The error corrector takes the combination of the source tokens and the phonological features of problematic tokens as input for correction. During error correction, each token of the extended input can be tagged to a Chinese character or a blank token "ϵ", so that the model can support *variable-length* correction. Since different pinyin letters may generate the same character, we eliminate continuous repetitive characters in the final result. In this paper, we use Chinese pinyin as the phonological feature. The learning of PhVEC is conducted end-to-end, with the error detector and corrector optimized jointly.

2.1 Error Detection Network

The goal of the error detector is to check whether a character x_i $(1 \leq i \leq n)$ is correct or not. We model this task as sequence labeling. We build a sequential binary classifier and use class 1 and 0 to label the problematic characters and correct characters, respectively. The ground-truth detection label $C = (c_1, c_2, ..., c_n)$ is pre-calculated by matching X and Y with edit distance, in which $c_i \in \{0, 1\}$. The prediction result of the error detector is represented by a sequence $C' = (c'_1, c'_2, ..., c'_n)$, and we use p_i to denote the probability of token x_i being predicted to class 1, which can be formalized as follows:

$$p_i = p(c'_i = 1|X) = softmax(f_{dec}(E(\boldsymbol{e_w}))) \quad (1)$$

where $\boldsymbol{e_w} = (e_{x_1}, e_{x_2}, ..., e_{x_n})$ is the token embedding of X, E is a Transformer-based encoder and f_{dec} is a fully-connected layer that maps the sentence representation to a binary sequence. To train the model, we adopt the following cross entropy loss function:

$$\mathcal{L}_{dec} = -\frac{1}{n}\sum_i [c_i \ln p_i + (1 - c_i) \ln(1 - p_i)] \quad (2)$$

2.2 Error Correction Network

After identifying the errors, we introduce the pinyin features of the incorrect characters and feed them into the correction network. Concretely, we use the tool PyPinyin[2] to generate pinyin for each erroneous Chinese character and insert pinyin tokens after the problematic token. The original

[2]https://github.com/mozillazg/python-pinyin

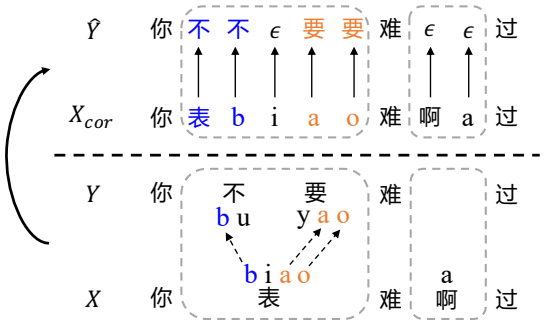


Figure 2: An illustration of generating X_{cor} and $\hat{Y}$ from the training sample (X, Y). We first label pinyin for incorrect characters in X and their counterparts in Y (bottom of the Figure). Then align the pinyin token according to the LCS algorithm (dashed arrows). X_{cor} is obtained by inserting pinyin tokens after the incorrect characters in X. Finally, $\hat{Y}$ is generated according to the alignment. For NULL alignments, such as "*i*", we align a special token "ϵ".

sentence $X = (x_1, x_2, ..., x_n)$ is thus rewritten as $X_{cor} = (x'_1, x'_2, ..., x'_t)$, where $t \geq n$, and $t - n$ is the number of pinyin tokens.

Given X_{cor}, the error corrector performs sequence labeling to predict the correction result $Y' = (y'_1, y'_2, ..., y'_t)$ as follows:

$$p(y'_i = \mathcal{V}_j|X_{cor}) = softmax(f_{crt}E(\boldsymbol{e_{cor}})) \quad (3)$$

where $p(y'_i = \mathcal{V}_j|X_{cor})$ is the conditional probability that x'_i is corrected to $\mathcal{V}_j$, the token with index j in the vocabulary $\mathcal{V}$. $\boldsymbol{e_{cor}}$ is the token embedding of X_{cor} and f_{crt} is a fully-connected layer that maps the hidden states of source tokens to their predicted logit vector of length $|\mathcal{V}|$. Note that, the parameters of the token embedding, the encoder E and the correction network f_{crt} are initialized by pre-trained language models (Devlin et al., 2019; Sun et al., 2020). We share the pre-training model parameters in the error detector and corrector to encode the Chinese characters and pinyin tokens in a shared space, which not only reduces the model size, but also makes the model yield better semantic representation.

The learning objective of the error corrector is to correct X_{cor} to the golden correction Y. However, it is not an easy task because the lengths of X_{cor} and Y may not be the same. Therefore, we rewrite Y to $\hat{Y} = (\hat{y}_1, \hat{y}_2, ..., \hat{y}_t)$ which has the same length as X_{cor}. Concretely, $\hat{Y}$ is constructed from Y with some tokens repeated and inserted according to pronunciation alignment. We compare the pinyin of the incorrect tokens of X with their

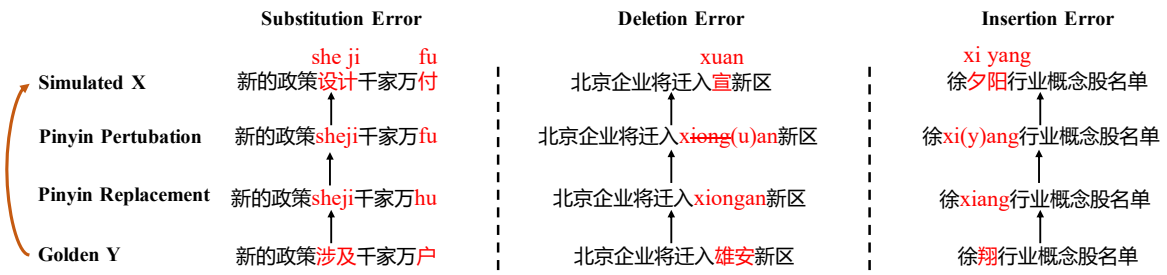


Figure 3: The process of generating training samples. The characters marked in red from bottom to top represent the correct Chinese characters, the pinyin corresponding to the correct Chinese characters, the simulated pinyin disturbance, and the simulated error characters corresponding to the noisy pinyin, respectively.

correct counterparts of Y, and align according to the longest common substring (LCS) algorithm, as illustrated by the dashed arrows in the bottom of Figure 2. Then we rewrite Y to $\hat{Y}$ according to the alignment.

The loss of the error corrector can be defined as follows:

$$\mathcal{L}_{cor} = -\sum_{i=1}^{t} \log(p(y'_i = \hat{y}_i | X_{cor})) \quad (4)$$

Our proposed pinyin alignment provides an explicit clue for the model to learn the correlation between pinyin and Chinese characters. As an alternative, we can also learn the alignment through Connectionist Temporal Classification (CTC) (Graves et al., 2006), which is widely used for *variable-length* alignments, such as in ASR (Audhkhasi et al., 2019), handwriting recognition (Bluche et al., 2014), etc. But we don't use CTC in our model because we assume that aligning according to the phonological features is reliable for ASR error correction, which will reduce the learning difficulty compared with learning a global alignment. We confirm our assumption in experiments.

2.3 Joint Learning and Inference

For each training sample (X, Y), we construct its golden detection result C and correction target $\hat{Y}$ first, then jointly optimize the two modules as follows:

$$\mathcal{L} = \lambda \cdot \mathcal{L}_{dec} + (1 - \lambda) \cdot \mathcal{L}_{cor} \quad (5)$$

where λ is a trade-off parameter.

During the inference stage, we first detect the problematic characters and insert their pinyin into the original sentence, then predict the correction sequence Y' by maximizing the probability of $p(y'_i | X_{cor})$ for each i. The final correction result is generated by removing the blank tokens ϵ and merging adjacent duplicate characters from the prediction of the error corrector, as shown at the top of Figure 1(d). Note that such post-processing is conducted only at locations where X_{cor} is modified relative to X, while the characters predicted as correct by the error detector are kept unchanged in the final prediction.

3 Training Data Generation

We follow the common practice in error correction (Zhang et al., 2020; Takahashi et al., 2020; Leng et al., 2021) to synthesize the training corpus with simulated ASR errors. The simulated ASR results are generated by replacing the Chinese characters or words of clean sentences into problematic ones. Specifically, we generate the noisy text in three steps: (i) sampling some candidate characters and replace them with pinyin. (ii) adding noise to the original pinyin and generate new valid ones (iii) producing new Chinese characters or words based on the updated pinyin.

In the first step, the candidate words are obtained from a confusion set (Wang et al., 2019) that contains words prone to be mis-recognized. In the second step, we design three strategies of pinyin perturbation for simulating substitution errors, deletion errors and insertion errors, as shown in Figure 3. Note that when generating deletion and insertion errors, some random letters may be inserted to construct valid pinyin sequence. In the last step, we replace the noisy pinyin with corresponding word candidates, and select the sentence ranked highest by a n-gram language model as the final simulated ASR result.

4 Experiments

We carry out experiments on two Chinese ASR error correction datasets. We use word error rate (WER) and word error reduction rate (WERR) to evaluate our error correction performance.

4.1 Data Settings

We first train an ASR model on AISHELL-1 (Bu et al., 2017). The training set contains 150 hours of Mandarin speech, along with corresponding transcripts, mainly including news in Finance, Technology, Sports, etc. The trained ASR model then transcribes the speech to generate the paired data for evaluation, as listed in Table 1. AISHELL-1 dev and test sets contain 15 hours of speech, corresponding to 21K sentence pairs. MAGICDATA[3] contains 43 hours of Mandarin speech, including interactive Q&A, daily instructions, etc.

The training set of ASR error correction is constructed from 3 million web-crawled sentences, along with their noisy version with simulated errors. For the methods that only support fixed-length correction, the pseudo ASR input is generated with similar-pronounced substitution errors only. For the training data of methods that support *variable-length* correction, we sample 2.7 million samples from the fixed-length training data and generate simulated insertion and deletion errors for the remaining 0.3M sentences.

4.2 Training Details

Consistent with previous work (Leng et al., 2021), we use the ESPnet (Watanabe et al., 2018) toolkit to train an ASR model on AISHELL-1 training set. Conformer architecture (Gulati et al., 2020) and SpecAugment (Park et al., 2019) are also used to improve the ASR performance. For the ASR error correction network, we take "chinese-bert-wwm" (Cui et al., 2019) as a pre-trained model to initialize the encoders of our PhVEC and other pretraining-based methods. We set the learning rate to 5e-5 and the probability of dropout to 0.1. The loss balancing parameter λ in joint learning is set to 0.5, and the AdamW (Loshchilov and Hutter, 2019) is utilized as optimizer. More details can be found in the Appendix A.

[3]https://openslr.org/68/

Dataset	AISHELL-1	MAGICDATA
Dev	14,329	6,756
Test	7,176	15,131
Avg Length	14.42	9.78
Type	News	Command

Table 1: Statistical results on experimental test datasets.

4.3 Baselines

We compare our PhVEC with the autoregressive Transformer (Vaswani et al., 2017), BART[4] (Lewis et al., 2020) and recent state-of-the-art non-autoregressive methods as follows: **Levenshtein Transformer (LevT)** (Gu et al., 2019) supports *variable-length* correction by iteratively performing deletion, insertion and substitution under NAR framework. **FastCorrect** (Leng et al., 2021) implements *variable-length* correction with a length predictor that estimates the number of target tokens each source token should be converted to, then repeating/droping the source tokens to generate a variable length sequence as the input of error correction. **MLM-phonetics** (Zhang et al., 2021a) implements a fixed-length error correction by pre-training a language model with phonological features integrated. **BERT** directly fine-tunes the standard masked language model to generate fixed-length corrections. **BERT+CTC** implements *variable-length* error correction with a CTC layer built on top of BERT, which upsamples each source token twice and learns the source-target alignments automatically with a CTC loss.

4.4 Overall Results

The comparison results on two benchmark datasets are shown in Table 2. We observe that:

- Our proposed PhVEC outperforms all the other methods on the evaluation datasets. It achieves about 20% WERR for the four datasets with varying levels of ASR performance. PhVEC greatly exceeds the two existing methods supporting *variable-length* correction, LevT, and FastCorrect, and become the first method that surpasses the autoregressive Transformer and BART models, with the WERR improved by 8.55% and 2.39% on average, respectively.

[4]https://huggingface.co/fnlp/bart-base-chinese

Method	AISHELL-1		MAGICDATA		Latency(ms/sent)
	Dev	TEST	Dev	TEST	TEST
	WER↓ (WERR↑)	WER↓ (WERR↑)	WER↓ (WERR↑)	WER↓ (WERR↑)	GPU
No correction	4.46 (-)	4.83 (-)	13.82 (-)	13.51 (-)	-
Transformer	3.80 (14.80%)	4.08 (15.53%)	12.09 (12.52%)	12.19 (9.77%)	149.5 (1×)
BART	3.62 (18.83%)	3.81 (21.12%)	11.36 (17.80%)	11.33 (16.14%)	180.9 (0.8×)
LevT (Max iter=1)	4.37 (2.02%)	4.73 (2.07%)	13.81 (0.07%)	13.91 (-2.96%)	54.0 (2.8×)
FastCorrect	3.89 (12.78%)	4.16 (13.87%)	-	-	21.2 (7.1×)
BERT	3.71 (16.82%)	3.98 (17.60%)	11.67 (15.56%)	11.79 (12.73%)	14.1 (10.6×)
BERT + CTC	3.78 (15.25%)	4.01 (16.98%)	12.13 (12.23%)	12.11 (10.36%)	14.5 (10.3×)
MLM-phonetics	3.64 (18.39%)	3.90 (19.25%)	11.59 (16.14%)	11.68 (13.55%)	23.7 (6.3×)
PhVEC (Ours)	**3.52** (**21.08%**)	**3.62** (**25.05%**)	**11.19** (**19.03%**)	**11.04** (**18.28%**)	24.1 (6.2×)

Table 2: Performance comparison of our method and other baselines on development set and test set. WER denotes word error rate (%), and WERR is word error reduction rate. We also test the inference speed of the correction models on NVIDIA V100 GPU and the test batch size is set to 1 sentence to match the online serving environment.

- Both PhVEC and MLM-phonetics introduce phonological features into the model but PhVEC performs better. We attribute this improvement to two aspects: one is its effective solution to *variable-length* errors, and the other is the effective use of pinyin splitting strategy. Different from Zhang et al. (2021a), we use pinyin features in letter granularity, instead of treating each pinyin as one token. This facilitates flexible insertion and enhances the correlation between similar pronounced characters and their corresponding phonological features.

- Pre-training significantly promotes correction performance. For the autoregressive models, the pre-trained BART has lower WER compared to the vanilla Transformer model. Moreover, the fine-tuned BERT achieves comparable performance with Transformer, indicating that strong language modeling will greatly facilitate NAR methods for error correction. Among the pretraining-based methods, PhVEC still performs the best, demonstrating the effectiveness of leveraging pinyin tokens for *variable-length* correction.

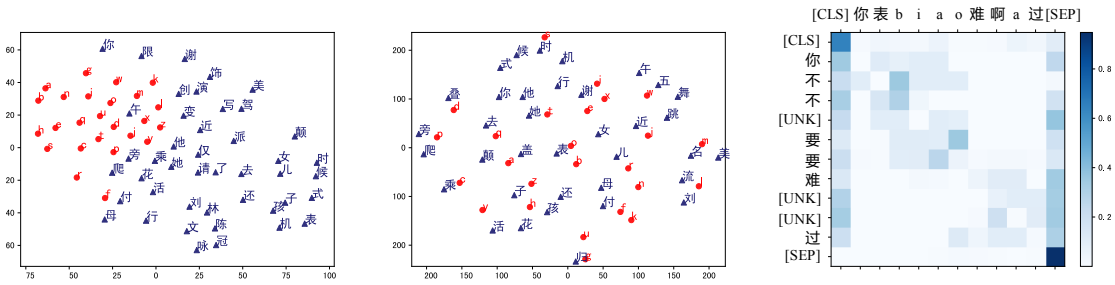
- Adding the CTC layer does not bring obvious advantages for error correction. In particular, BERT+CTC is inferior to BERT, and prones to generate incorrect alignments for the fixed-length correction samples. This might be because, for ASR error correction, the alignment between most words of the source and target is definite. BERT+CTC upsamples each source token twice and dictates the model to learn the alignment, which actually increases the difficulty of learning. It is worth noting that BERT+ CTC outperforms BERT on the *variable-length* error correction samples, which will be introduced later in Table 3.

- PhVEC accelerates to 6.2 times that of Transformer, and further reduces the WER by 8-10%. Even compared with the BART model, we still have 2-3% WER reduction and speed up the inference by 7.5 times, which proves the efficiency and effectiveness of PhVEC. Moreover, we want to emphasize that the original FastCorrect used NVIDIA V100 and P40 GPU to test the model delay respectively. To ensure the consistency of the results, we reimplement all baseline methods and test the inference speed on NVIDIA V100 GPU.

It is also notable that the training data of our method PhVEC includes 3M sentence pairs, which is consistent with that of Transformer, BART, LevT, BERT, and BERT+CTC, but FastCorrect and MLM-phonetics use additional 400M and 300M pairs sentences for pre-training, respectively.

4.5 Analyses

We further analyze the performance of our model on *variable-length* datasets and conduct ablation



(a). **BERT-tSNE** (b). **PhVEC-tSNE** (c). **PhVEC-attention**

Figure 4: (a) and (b) are t-SNE visualization of pinyin letter and Chinese character embeddings, which correspond to BERT and PhVEC models respectively. (c) is the self-attention of the 3^{rd} encoder layer of the error corrector. The horizontal axis represents X_{cor}, the input of the error corrector, and the vertical tokens are the model output, in which [UNK] is for the blank token ϵ.

Method	AISHELL-1	MAGICDATA
No correction	15.02 (-)	21.38 (-)
Transformer	13.90 (7.5%)	20.68 (3.3%)
BART	12.08 (19.6%)	17.30 (19.1%)
LevT	14.34 (4.5%)	23.11 (-8.1%)
BERT	14.59 (2.9%)	20.08 (6.1%)
BERT+CTC	12.98 (13.6%)	19.67 (8.0%)
MLM-phonetics	14.48 (3.6%)	20.12 (5.9%)
PhVEC (Ours)	**11.34** (**24.5%**)	**16.06** (**24.9%**)

Table 3: The WER↓(WERR↑) evaluated on subsets that containing only the *variable-length* correction samples. The number of such samples in AISHELL-1 and MAGICDATA is 163 and 1490, respectively.

Strategy	Example	AISHELL-1	MAGICDATA
No correction	[表]	4.83 (-)	13.51 (-)
OneToken	[表][biao]	4.08 (15.5%)	12.11 (10.4%)
Initial&Final	[表][b][iao]	3.91 (19.0%)	11.68 (13.5%)
Letters-only	[b][i][a][o]	3.85 (20.3%)	11.59 (14.2%)
Letters (Ours)	[表][b][i][a][o]	**3.62** (**25.1%**)	**11.04** (**18.3%**)

Table 4: The WER↓(WERR↑) of different phonological features evaluated on the AISHELL-1 and MAGICDATA test sets.

studies to dissect the factors affecting the effectiveness of our method.

4.5.1 *Variable-length* scenario

To evaluate the effectiveness of our model in dealing with *variable-length* errors, we extract the samples containing insertion and deletion errors from the test sets of AISHELL-1 and MAGICDATA, and evaluate the above methods on this subset. As shown in Table 3, PhVEC shows significant advantages over other methods, achieving over 20% WERR on both datasets. Both fined-tuned BERT and MLM-phonetics perform weakly because of the inherent fixed-length limitation of their generation. The CTC layer brings significant improvement to BERT by enabling it with *variable-length* prediction, but there is still a large gap between its performance and that of PhVEC. The correction generated by LevT is not stable, even degrades the WERR by 8.1% WERR in the MAGICDATA subset. This might be because its length prediction result is inaccurate. The average length difference between the input and the golden correction is only 1.08, while the average length gap between the prediction of LevT and the golden correction is 1.42. This indicates that the prediction of LevT will produce more length differences than the original ASR input, resulting in its performance degradation.

4.5.2 Different manners of leveraging phonological features

In our model, we split the pinyin of each Chinese character into tokens letter by letter and add them after each detected problematic character, e.g., rewrite the detected error character "[表]" (biao) to 5 tokens [表][b][i][a][o] in the intermediate *variable-length* result. Here we explore the impact of leveraging pinyin with different strategies: 1) *OneToken*: taking the pinyin of each character as one token as in Zhang et al. (2021a); 2) *Initial&Final*: divide each pinyin into an initial ([b]) and a final ([iao]), according to the phonological portion of Chinese; 3) *Letters-only*: remove the original problematic character from the rewritten sentence and use its pinyin for substitution. The first two settings focus on different pinyin granular-

ities, and the third one wants to check whether the original characters detected as errors are useful for error correction.

Table 4 shows that: (1) Compared with *Letters*, both *OneToken* and *Initial&Final* degrades the performance. This is because the granularity of the phonological features used in the two methods are coarser than that of *Letters*, which reduces the probability of aligning each phonological feature to more Chinese characters. (2) The *Letters-only* performs inferior to *Letters*, demonstrating the effectiveness of keeping the original characters in X_{cor}. This might be because, some wrong detection results tag "C" to some correct words. Removing these original words when adding pinyin tokens makes the model almost impossible to recover the correct ones. This always happens to rare words in some named entities.

4.5.3 Characters Embedding and Alignment

To qualitatively examine whether PhVEC learns meaningful representations, we dive into the learned embedding and encoder for visualization. We first investigate whether the model learns the relationship between pinyin letters and Chinese characters. Concretely, we visualize the learned embedding of pinyin letter and Chinese character in a two-dimensional space by applying the t-SNE algorithm (van der Maaten and Hinton, 2008). Figure 4b shows that the embedding of Chinese characters and their corresponding pinyin letters get closer after training with PhVEC. Then we visualize the self-attention of the encoder in the error corrector. Figure 4c shows that the corrected words in the output pay much attention to their corresponding pinyin tokens, for example, "不(bu)" is highly aligned to "b", and "要 (yao)" paid much attention to "a" and "o", which indicates that the phonological features provide an obvious prompt for error correction.

5 Related Work

5.1 Chinese Spelling Error Correction (CSC)

As a close related area to ASR error correction, CSC has been widely explored in recent years. Zhang et al. (2020) proposes a soft-masked BERT model that first predicts the error probability of each character, and then uses the probabilities to perform a soft-masked word embedding for correction. As a remedy of soft-masked BERT, Zhang et al. (2021a) incorporates phonological knowledge into pre-training and proposes to fuse phonological feature in error correction. Cheng et al. (2020) builds a Graph Convolution Network on top of BERT, which reflects the phonological similarity among Chinese tokens. However, these methods are designed to produce corrections of the same length as the input, but incapable of handling *variable-length* correction that includes errors of substitution, deletion and insertion.

5.2 Autoregressive (AR) Error Correction

To correct *variable-length* errors, a large number of Seq2Seq AR models have been proposed. Zhang et al. (2019) uses a Transformer-based model for Chinese ASR error correction. Wang et al. (2019) incorporates a copy mechanism into Seq2Seq framework to copy the corrections directly from a prepared confusion set for the erroneous words. With the popularity of pre-training, Zhao et al. (2021) uses a pre-trained BART (Lewis et al., 2020) to initialize the correction network. Although these AR models are able to deal with various types of errors in ASR, they can not satisfy the latency requirements for online services, especially for some real-time scenarios like simultaneous translation (Zhang et al., 2021b).

5.3 Non-Autoregressive (NAR) Error Correction

NAR models are designed for fast generation speed compared with their AR counterpart by producing all tokens in a target sequence in parallel, which is widely explored in machine translation (Gu et al., 2019; Xu and Carpuat, 2021), ASR (Fan et al., 2021), TTS (Ren et al., 2019) etc. Recently, some studies (Gu et al., 2019; Leng et al., 2021) propose to apply NAR models to *variable-length* ASR error correction based on a length predictor, which first estimates the length of the target correction, then rewrites the input sentence by dropping or repeating some tokens according to the length estimated, finally performs a sequence labeling on the rewritten sentence to achieve correction. However, it is difficult to predict the target length of an incorrect sentence directly, even for humans, while the accuracy of length prediction is closely related to the performance of error correction.

6 Conclusion

In this paper, we propose a non-autoregressive Chinese ASR error correction network with phonolog-

ical training. Our method first detects the problematic characters, then adds the phonological features of them to adjust the input length, thus generating a *variable-length* sequence for error correction. The phonological features enable our model to produce similar-pronounced corrections, and support *variable-length* correction in a non-autoregressive mode. Experiments show that our method is superior to the autoregressive method while maintaining a 6.2x speed-up. As a future work, we plan to extend PhVEC to other languages and use corresponding phonological tokens to correct the variable length errors caused by pronunciation.

Acknowledgements

This research is supported by the National Key Research and Development Program of China (NO.2017YFC0820700) and National Natural Science Foundation of China (No.61902394). We thank all authors for their contributions and all anonymous reviewers for their constructive comments.

References

C. Anantaram, Amit Sangroya, Mrinal Rawat, and Aishwarya Chhabra. 2018. Repairing ASR output by artificial development and ontology based learning. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018*, pages 5799–5801. ijcai.org.

Kartik Audhkhasi, George Saon, Zoltán Tüske, Brian Kingsbury, and Michael Picheny. 2019. Forget a bit to learn better: Soft forgetting for ctc-based automatic speech recognition. In *Interspeech 2019, 20th Annual Conference of the International Speech Communication Association, Graz, Austria, 15-19 September 2019*, pages 2618–2622. ISCA.

Théodore Bluche, Hermann Ney, and Christopher Kermorvant. 2014. A comparison of sequence-trained deep neural networks and recurrent neural networks optical modeling for handwriting recognition. In *Statistical Language and Speech Processing - Second International Conference, SLSP 2014*, volume 8791 of *Lecture Notes in Computer Science*, pages 199–210. Springer.

Hui Bu, Jiayu Du, Xingyu Na, Bengu Wu, and Hao Zheng. 2017. Aishell-1: An open-source mandarin speech corpus and a speech recognition baseline. In *Oriental COCOSDA 2017*, page Submitted.

Xingyi Cheng, Weidi Xu, Kunlong Chen, Shaohua Jiang, Feng Wang, Taifeng Wang, Wei Chu, and Yuan Qi. 2020. Spellgcn: Incorporating phonological and visual similarities into language models for chinese spelling check. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020*, pages 871–881. Association for Computational Linguistics.

Yiming Cui, Wanxiang Che, Ting Liu, Bing Qin, Ziqing Yang, Shijin Wang, and Guoping Hu. 2019. Pre-training with whole word masking for chinese BERT. *CoRR*, abs/1906.08101.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019*, pages 4171–4186. Association for Computational Linguistics.

Ruchao Fan, Wei Chu, Peng Chang, Jing Xiao, and Abeer Alwan. 2021. An improved single step non-autoregressive transformer for automatic speech recognition. *CoRR*, abs/2106.09885.

Alex Graves, Santiago Fernández, Faustino J. Gomez, and Jürgen Schmidhuber. 2006. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Machine Learning, Proceedings of the Twenty-Third International Conference (ICML 2006)*, volume 148 of *ACM International Conference Proceeding Series*, pages 369–376. ACM.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor O. K. Li, and Richard Socher. 2018. Non-autoregressive neural machine translation. In *6th International Conference on Learning Representations, ICLR 2018*. OpenReview.net.

Jiatao Gu, Changhan Wang, and Junbo Zhao. 2019. Levenshtein transformer. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019*, pages 11179–11189.

Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, and Ruoming Pang. 2020. Conformer: Convolution-augmented transformer for speech recognition. In *Interspeech 2020, 21st Annual Conference of the International Speech Communication Association*, pages 5036–5040. ISCA.

Jinxi Guo, Tara N. Sainath, and Ron J. Weiss. 2019. A spelling correction model for end-to-end speech recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2019, Brighton, United Kingdom, May 12-17, 2019*, pages 5651–5655. IEEE.

Oleksii Hrinchuk, Mariya Popova, and Boris Ginsburg. 2020. Correction of automatic speech recognition with transformer sequence-to-sequence model. In *2020 IEEE International Conference on Acoustics,*

Speech and Signal Processing, ICASSP 2020, pages 7074–7078. IEEE.

Yichong Leng, Xu Tan, Linchen Zhu, Jin Xu, Renqian Luo, Linquan Liu, Tao Qin, Xiang-Yang Li, Ed Lin, and Tie-Yan Liu. 2021. Fastcorrect: Fast error correction with edit alignment for automatic speech recognition. In *35th Conference on Neural Information Processing Systems, NeurIPS 2021*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020*, pages 7871–7880.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019*. OpenReview.net.

Anirudh Mani, Shruti Palaskar, Nimshi Venkat Meripo, Sandeep Konam, and Florian Metze. 2020. ASR error correction and domain adaptation using machine translation. In *2020 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2020, Barcelona, Spain, May 4-8, 2020*, pages 6344–6348. IEEE.

Daniel S. Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D. Cubuk, and Quoc V. Le. 2019. Specaugment: A simple data augmentation method for automatic speech recognition. In *Interspeech 2019, 20th Annual Conference of the International Speech Communication Association*, pages 2613–2617. ISCA.

Yi Ren, Jinglin Liu, Xu Tan, Zhou Zhao, Sheng Zhao, and Tie-Yan Liu. 2020. A study of non-autoregressive model for sequence generation. *arXiv preprint arXiv:2004.10454*.

Yi Ren, Yangjun Ruan, Xu Tan, Tao Qin, Sheng Zhao, Zhou Zhao, and Tie-Yan Liu. 2019. Fastspeech: Fast, robust and controllable text to speech. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019*, pages 3165–3174.

Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Hao Tian, Hua Wu, and Haifeng Wang. 2020. Ernie 2.0: A continual pre-training framework for language understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8968–8975.

Yujin Takahashi, Satoru Katsumata, and Mamoru Komachi. 2020. Grammatical error correction using pseudo learner corpus considering learner's error tendency. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop, ACL 2020, Online, July 5-10, 2020*, pages 27–32. Association for Computational Linguistics.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017*, pages 5998–6008.

Dingmin Wang, Yi Tay, and Li Zhong. 2019. Confusionset-guided pointer networks for chinese spelling check. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019*, pages 5780–5785. Association for Computational Linguistics.

Shinji Watanabe, Takaaki Hori, Shigeki Karita, Tomoki Hayashi, Jiro Nishitoba, Yuya Unno, Nelson Enrique Yalta Soplin, Jahn Heymann, Matthew Wiesner, Nanxin Chen, Adithya Renduchintala, and Tsubasa Ochiai. 2018. Espnet: End-to-end speech processing toolkit. In *Interspeech 2018, 19th Annual Conference of the International Speech Communication Association, Hyderabad, India, 2-6 September 2018*, pages 2207–2211. ISCA.

Weijia Xu and Marine Carpuat. 2021. EDITOR: an edit-based transformer with repositioning for neural machine translation with soft lexical constraints. *Trans. Assoc. Comput. Linguistics*, 9:311–328.

Ruiqing Zhang, Chao Pang, Chuanqiang Zhang, Shuohuan Wang, Zhongjun He, Yu Sun, Hua Wu, and Haifeng Wang. 2021a. Correcting chinese spelling errors with phonetic pre-training. In *Findings of the Association for Computational Linguistics: ACL/IJCNLP 2021*, pages 2250–2261. Association for Computational Linguistics.

Ruiqing Zhang, Xiyang Wang, Chuanqiang Zhang, Zhongjun He, Hua Wu, Zhi Li, Haifeng Wang, Ying Chen, and Qinfei Li. 2021b. BSTC: A large-scale Chinese-English speech translation dataset. In *Proceedings of the Second Workshop on Automatic Simultaneous Translation*.

Shaohua Zhang, Haoran Huang, Jicong Liu, and Hang Li. 2020. Spelling error correction with soft-masked BERT. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 882–890. Association for Computational Linguistics.

Shiliang Zhang, Ming Lei, and Zhijie Yan. 2019. Investigation of transformer based spelling correction model for ctc-based end-to-end mandarin speech recognition. In *Interspeech 2019, 20th Annual Conference of the International Speech Communication Association, Graz, Austria, 15-19 September 2019*, pages 2180–2184. ISCA.

Yun Zhao, Xuerui Yang, Jinchao Wang, Yongyu Gao, Chao Yan, and Yuanfu Zhou. 2021. BART based semantic correction for mandarin automatic speech recognition system. *CoRR*, abs/2104.05507.

A Experimental Details

A.1 Structure and parameters of ASR model

The ASR model is an end-to-end encoder-attention-decoder model with a 12-layer conformer encoder and a 6-layer conformer decoder, which is trained with cross-entropy loss on decoder output and an auxiliary CTC loss on encoder output. For the hyper-parameters of the ASR model, we take the beam search decoding with beam size to be 10, conformer kernel size to be 15, ctc weight to be 0.6, lm weight to be 0.3.

A.2 Balance the objective of detection and correction

We explore the impact of the weighting strategy that balances the two objectives in fine-tuning. Table 5 presents the results of PhVEC in different values of hyper-parameter λ. Specifically, a larger λ value means a higher weight on error detection, and the highest F1 score is obtained when λ is 0.5.

λ	AISHELL-1		MAGICDATA	
	Dev(WER↓)	TEST(WER↓)	Dev(WER↓)	TEST(WER↓)
0.2	3.61	3.75	11.42	11.37
0.5	3.52	3.62	11.19	11.04
0.8	3.74	3.83	11.55	11.47

Table 5: Impact of Different Values of λ.