Pinyin as a feature of neural machine translation for Chinese speech recognition error correction

Dagao Duan[1], Shaohu Liang, Zhongming Han, Weijie Yang

[1] Beijing Technology and Business University, Haidian, Beijing, China
duandg@th.btbu.edu.cn

Abstract. Text correction after automatic speech recognition (ASR) is an important method to improve the speech recognition system. We regard the speech error correction as a translation task—from the language of bad Chinese to the language of good Chinese. We propose a speech recognition error correction algorithm based on neural machine translation (NMT) model. The algorithm is characterized by Chinese Pinyin coding, using a multilayer convolutional encoder-decoder with attention neural network. In the WeChat speech transcription data set we collected, our model substantially outperforms all prior neural approaches on this data set as well as the strong statistical machine translation-based systems. Our analysis shows the superiority of convolutional neural networks in capturing the local context via attention and thereby improving the coverage in speech transcription errors. By boosting multiple modes, using data augmentation and 3-gram language model tricks, our novel algorithm makes the error rate on the test set decreased by 26.2% on average. Our results show that using a multilayer convolutional encoder-decoder with Pinyin feature is able to achieve state-of-the-art performance in text correction after speech recognition.

Keywords: Automatic Speech Recognition, Neural Machine Translation, Attention Mechanism, Pinyin Encoding, Chinese Error Correct

1 Introduction

In the speech recognition system, the text after the speech recognition is difficult to be understood and there are many errors in the text due to the noisy background environment of the speaker or the phenomenon of swallowing, dragging, accent and dialect of the speaker. Generally speaking, there are four types of text errors, including redundant words (R), missing words (M), bad word selection (S) and mixing errors (Mi). Table 1 lists examples of each type of error. Because of the speaker's dragging, the speech recognition system separated the "pian" vowels into "ping" and "an" syllable, resulting in redundancy errors. Because of the speaker's accent or dialect, the speech recognition system recognizes the "na" vowel "a" as "ei", resulting in the incorrect selection of the Chinese character "那" with "内". Because the speaker speaks fast, or some zero initial Chinese characters, it is easy to join the first Chinese character and the second Chinese syllable together, and the speech recognition system identifies two syllables as one syllable, which leads to deletion errors. For example, "xi’an" is identified as "xian". The last type of error is a combination of the first three types, which will result in mixing errors due to the speaker's non-standard tone and speaking speed. For example, in the last column of table 1, " xué" can be translated as " xuě", and “yǐ yǐ” can be translated as " yǐ", resulting in mixed errors in selection and deletion. Due to the complexity of speech recognition errors in Chinese language, the common methods based on rule matching and simple statistics are difficult to effectively solve practical problems.

Table 1. Error Types

Error type	Bad sentence	Good sentence
Redundant error	有没有平安一点的衣服啊 Do you have anything safe clothes yǒu méi yǒu píng ān yì diǎn de yī fú a	有没有便宜点的衣服啊 Do you have any cheaper clothes yǒu méi yǒu pián yí diǎn de yī fú a
Replacement error	内个人是我妈妈 Inner one women is my mother nèi gè rén shì wǒ mā mā	那个人是我妈妈 That woman is my mother nà gè rén shì wǒ mā mā
Missing error	暑假我去先玩了 I went to play first in the summer vacation shǔ jià wǒ qù xiān wán le	暑假我去西安玩了 I went to xi 'an during the summer vacation shǔ jià wǒ qù xī ān wán le
Mixing error	雪不可以 Snow can not xuě bù kě yǐ	学不可以已 You cannot learn by halves xué bù kě yǐ yǐ

2 Related Works

In recent years, with the development of automatic speech recognition, speech recognition has been significantly improved. However, due to noise and speaker accent, the results of speech recognition are still unsatisfactory. Therefore, text correction after speech recognition is an auxiliary task to improve speech recognition. Text error correction is an important task in natural language processing. After Ng et al. organized the CoNLL-2013 open task [1], many methods based on statistics and neural network were proposed, which greatly improved the research on English error correction. Brockett [2] put forward the concept of "translation" in the first time, by putting the wrong sentences "translate" the correct sentence to achieve the purpose of syntax error correction. The use of channel noise statistical machine translation model (SMT) came to correct errors in English. In addition, SMT can be combined with other methods, such as integrating rule-based methods to build complex and efficient systems [3]. Recently, the SMT-based method proposed by Chollampatt [4] has achieved fairly good error correction results. With the development of deep learning, neural machine translation (NMT) as a new paradigm has been put forward and applied to machine translation. Compared with SMT, NMT has greatly improved the translation quality. On the basis of previous work, Yuan [5] applied NMT to English error correction tasks, specifically, they used the classical translation model: a bidirectional RNN encoder and decoder model containing attention mechanism. In order to solve the problem of Out Of Vocabulary (OOV), Ji [6] applied a hybrid NMT model that combines word-level and character-level information. Chollampatt [7] firstly proposed the encoder-decoder model of convolution based on attention mechanism, which can extract local context information better. Later, Junczys [8] demonstrated the similarities between grammatical error correct（GEC) and low-resource neural machine translation, and transformed some low-resource neural machine translation methods into grammatical error correction methods, achieving the best results in GEC tasks. Recently, Grundkiewicz [9] proposed a hybrid system by combining SMT and NMT and achieved a level close to human. Chinese scholars focus on Chinese grammatical error diagnosis (CGED), Yu [10] et al. organized a CGED task, that goal was to develop a computer aided tool for CGED. The diagnosis types included redundant words, missing words, word order error and word choice error. Zheng [11], Xie [12] and Tan yongmei [13] regarded CGED as a sequence labeling problem. Their solution was to classify words in combination with CRF and LSTM. The CGED task prompted researchers to focus on the detection of grammatical errors in computational linguistics. The task still solely concentrates on the detection of the grammatical errors rather than the automatic generation of corrections. This limits its application. Hu yi [14] et al. proposed a method of error correction for search engine query by replacing the parts of the original query that need to be modified, then sorting all the candidate error correction queries according to various judgments to find the most suitable correction. Since most of the search engine queries are phrase errors and keyboard input errors, the speech errors in the sentences cannot be well corrected. Wu Long [15] used phrase-based statistical machine translation model to convert ASR Pinyin outputs to translated Pinyin. Then they used the beam search algorithm based on dynamic tree to convert translated Pinyin to error-corrected Chinese characters. Since one syllable can correspond to multiple Chinese characters， in the process of converting pinyin into Chinese characters, errors may occur.

This paper presents a neural network error correction model based on machine translation. We use word (character) vector as the features of the machine translation model, the Chinese Pinyin coding as an additional feature to join the network model. Since there is a one-to-one correspondence between word (character) vector and word (character). The Chinese Pinyin contains speech information, which better solves the problem of changing from pinyin to Chinese character and correcting the text that was wrong due to speech.

In sum, this paper makes three significant contributions:

1. We subtly employ a convolutional encoder-decoder model to achieve state-of-art performance for Chinese speech recognition error correction. To our knowledge, ours is the first work to use full convolutional neural networks for end-to-end Chinese speech recognition error correction.
2. We add Pinyin as an additional feature to the convolutional encoder-decoder model. Besides, we design 8 kinds of Pinyin coding schemas and detailly report corresponding performance. Our work may guide further development in Pinyin coding.
3. In order to get better results, we propose a data augmentation method. Specifically, we train a reverse generative model, that input are good sentences while labels are bad sentences. The reverse generative model can learn how to transform good

sentences to bad sentences. And we verify 3-gram language model which can further reduce the error rate.

3 Error correction translation model

3.1 Deep CNN error correction model

The neural network error correction model in the continuous vector space after speech recognition sentences $\mathbf{x} = (\mathrm{x}_1, \mathrm{x}_2, \dots, \mathrm{x}_\mathrm{m})$ map to the target sentence after error correction $\mathbf{y} = (\mathrm{y}_1, \mathrm{y}_2, \dots, \mathrm{y}_\mathrm{n})$. Short sentences are filled with a special marker <PAD> to keep the same length, and the end of each sentence is marked with <EOS>. Considering that most speech errors are caused by local words in a sentence, we use convolutional neural network (CNN), which can better extract the local context information of a sentence compared with RNN. Given the window size, the convolutional network glided on the input sequence to calculate the local characteristics of the window size. The wider context information and information between distant of Chinese characters can also be captured by a multi-level convolutional hierarchy. In addition, when predicting target words, we also use an attention mechanism, which assigns weights to source words according to the correlation between source words and target words.

The deep convolutional neural error correction model uses the encoder-decoder model. The encoder network is used to encode error-prone source statements in a vector space, and the decoder network generates a corrected output statement by using the source code. Our model is based on an encoder and decoder architecture with multi-layer convolution and attention mechanism. Network structure is described in detail below.

Given an input source statement $\boldsymbol{S}$ is composed of m segmentation $\mathrm{s}_1, \mathrm{s}_2, \dots, \mathrm{s}_i \dots, \mathrm{s}_m$ $\mathrm{s}_i \in V_s$, where V_s is the source vocabulary, the last segmentation marker s_m is a special statement end tag, source statement $\boldsymbol{S}$ in the continuous space of the vector $\boldsymbol{s_1}, \boldsymbol{s_2}, \dots, \boldsymbol{s_m}$, $\boldsymbol{s_i} \in \mathbb{R}^d$, $\boldsymbol{s_i} = \mathbf{w}(\boldsymbol{s_i}) + \boldsymbol{P}(\boldsymbol{i})$, where $\mathbf{w}(\boldsymbol{s_i})$ is the embedded segmentation, $\boldsymbol{P}(\boldsymbol{i})$ is the position of $\boldsymbol{s_i}$ in the source statement .These two kinds of embedding are obtained by training the embedding matrix together with other parameters of the network. The encoder and decoder are respectively composed of L layers. The network main frame is shown in figure 1.Source segmentation encode $\boldsymbol{s_1}, \boldsymbol{s_2}, \dots, \boldsymbol{s_m}$, by affine transformation for the first layer coding input $\boldsymbol{h_1^1}, \boldsymbol{h_2^1}, \dots, \boldsymbol{h_m^1}$ $\boldsymbol{h_i^1} \in \mathbb{R}^h$ h is all encoder decoder input/output vector dimensions. The affine transformation is obtained through formula (1).

$$\boldsymbol{h_i^1} = \boldsymbol{W}\boldsymbol{s_1} + \boldsymbol{b} \tag{1}$$

Where, the weight $\boldsymbol{W} \in \mathbb{R}^{\mathrm{h}\times\mathrm{d}}$, bias $\boldsymbol{b} \in \mathbb{R}^{\mathrm{h}}$.

In the first coding layer, there are a number of 2h convolution kernels with $3 \times h$ dimensions in windows size 3 to map the input vector to the eigenvector $f_i^2 \in \mathbb{R}^{2h}$. Adding <PAD> at the beginning and end of the source sentence (as shown in figure 1) is to preserve the output vector of the same length as the source sentence before the convolution operation.

$$\boldsymbol{f_i^2} = conv(\boldsymbol{h_{i-1}^1}, \boldsymbol{h_i^1}, \boldsymbol{h_{i+1}^1}) \tag{2}$$

Where $conv(\cdot)$ represents the convolution operation. Then pass the gate linear unit (GLU):

$$\mathrm{GLU}(\boldsymbol{f_i^2}) = \boldsymbol{f_{i,1:h}^2} \circ \sigma(\boldsymbol{f_{i,h+1:2h}^2}) \tag{3}$$

Where $\mathrm{GLU}(\boldsymbol{f_i^2}) \in \mathbb{R}^{\mathrm{h}}$,∘ Means multiply by elements, $\sigma(\cdot)$ Represents the sigmoid activation function, $\boldsymbol{f_{i,m:u}^2}$ Represents $\boldsymbol{f_i^2}$ from $\boldsymbol{m}$ to $\boldsymbol{u}$ elements. Finally, the input vector of the encoder layer is added as a residual join. The output vector of the l encoder layer is given by formula (4):

$$\boldsymbol{h_i^l} = GLU(\boldsymbol{f_i^l}) + \boldsymbol{h_i^{l-1}} \quad \boldsymbol{i} = 1,2,\dots,\mathrm{m} \tag{4}$$

The last layer encoder for each of the output vector $\boldsymbol{h_i^L} \in \mathbb{R}^h$ is obtained by linear mapping eventually encoder output vector $\boldsymbol{e_i} \in \mathbb{R}^{\mathrm{d}}$, as shown in (5):

$$\boldsymbol{e_i} = \mathbf{W_e}\boldsymbol{h_i^L} + \boldsymbol{b_e} \quad \boldsymbol{i} = 1,2,\dots,\mathrm{m} \tag{5}$$

Consider the decoding process of the decoder, Given $n-1$ generated target segment, to decode the target segment $\boldsymbol{t_\mathrm{n}}$ at time step n. First, <PAD> is added at the beginning of the target output, followed by <s>, They are embedded as $\mathbf{t_{-2}}, \mathbf{t_{-1}}, \mathbf{t_0}, \mathbf{t_1}, \dots, \mathbf{t_{n-1}}$ vector, the method of embedding is the same as that of source statement segmentation embedding. Each embedded $\mathbf{t_j} \in \mathbb{R}^{\mathrm{d}}$ is linearly mapped to $\boldsymbol{g_j^1} \in \mathbb{R}^{\mathrm{h}}$ as input to the first decoding layer. Each decoding layer, like the encoding layer, contains a nonlinear gate unit GLU and a convolution operation with a window size 3, as shown in equation (6):

$$\mathbf{y_j^l} = \mathrm{GLU}\left(\mathrm{conv}(\mathbf{g_{j-3}^{l-1}}, \mathbf{g_{j-2}^{l-1}}, \mathbf{g_{j-1}^{l-1}})\right) \mathrm{j} = 1,2,\dots,\mathrm{n} \tag{6}$$

Where $\boldsymbol{g_j^{l-1}}$ represents the output vector of the previous decoding layer. $\boldsymbol{y_j^l}$ represents the decoding state of the l layer decoder at the j time step, and the number and size of convolution kernels are the same as that of the encoder. Each layer decoder has an attention module that computes the attention at the time step n predicted by the $\boldsymbol{l}$ layer decoder. The decoder state $\boldsymbol{y_n^l} \in \mathbb{R}^{\mathrm{h}}$ is linearly mapped to a d-dimensional vector, as shown in equation (7):

$$\boldsymbol{z_n^l} = \boldsymbol{W_Z}\boldsymbol{y_n^l} + \boldsymbol{b_z} + \mathbf{t}_{\boldsymbol{n-1}} \tag{7}$$

Where the weight is $\boldsymbol{W_Z} \in \mathbb{R}^{\mathrm{d\times h}}$ and the bias is $\boldsymbol{b_z} \in \mathbb{R}^{\mathrm{d}}$, $\mathbf{t}_{\boldsymbol{n-1}}$ Represents the target segmentation embedding of the previous time step, Attention weight $\alpha_{n,i}^{l}$ is obtained by dot product operation of output vectors $\boldsymbol{z_n^l}$ and $\boldsymbol{e_1}, \dots, \boldsymbol{e_m}$ of the encoder output and then normalized through softmax, as shown in equation (8):

$$\alpha_{n,i}^{l} = \frac{\exp(\boldsymbol{e_i^T z_n^l})}{\sum_{k=1}^{m} \exp(\boldsymbol{e_k^T z_n^l})} \quad i = 1,2,\dots,m \tag{8}$$

The context vector $\boldsymbol{x_n^l}$ weighted the encoder output vector and source shard embedding by attention weight to obtain formula (9). The additional source shard embedding can better retain the source shard information without being diluted after multi-layer coding.

$$\boldsymbol{x_n^l} = \sum_{\boldsymbol{i=1}}^{\boldsymbol{m}} \alpha_{n,i}^{l}(\boldsymbol{e_i} + \boldsymbol{s_i}) \tag{9}$$

The context vector $\boldsymbol{x_n^l}$ is linearly mapped to $\boldsymbol{c_n^l} \in \mathbb{R}^{\mathrm{h}}$, and the output $\boldsymbol{g_n^l}$ of the l-layer decoder is obtained by adding $\boldsymbol{c_n^l}, \boldsymbol{y_n^l}$ and the previous layer's output vector$\boldsymbol{g_n^{l-1}}$, as shown in equation (10):

$$\boldsymbol{g_n^l} = \boldsymbol{y_n^l} + \boldsymbol{c_n^l} + \boldsymbol{g_n^{l-1}} \tag{10}$$

The final decoding layer output vector $\boldsymbol{g_n^L}$ is linearly mapped to $\boldsymbol{d_n} \in \mathbb{R}^{d}$.The Dropout [17] is used in each layer. The probability that the decoder output vector is finally mapped into the vector with the number of shards to get the target shard through softmax function, as shown in formula (11) and (12):

$$\boldsymbol{o_n} = \boldsymbol{W_o d_n} + \boldsymbol{b_o} \tag{11}$$

Where $\boldsymbol{W_o} \in \mathbb{R}^{|V_t|\times d}$, $\boldsymbol{b_o} \in \mathbb{R}^{|V_t|}$.

$$p(t_n = w_i | t_1, \dots, t_{n-1}, \boldsymbol{S}) = \frac{\exp(o_{n,i})}{\sum_{k=1}^{|V_t|} \exp(o_{n,k})} \tag{12}$$

w_i means the i-th token in the target token V_t.
We use the negative logarithmic likelihood loss function to train the model:

$$\mathcal{L} = -\frac{1}{\mathrm{N}} \sum_{\mathrm{i=1}}^{\mathrm{N}} \frac{1}{\mathrm{T_i}} \sum_{\mathrm{j=1}}^{\mathrm{T_i}} \log\left(\mathrm{p}\left(\mathrm{t_{i,j}} \middle| \mathrm{t_{i,1}}, \dots, \mathrm{t_{i,j-1}}, \mathbf{S}\right)\right) \tag{13}$$

Where N is the number of training instances in batch processing, T_i is the number of segmentations in the i-th sentence, and $t_{i,j}$ is the j-th target segmentation in the target sentence of the i-th training instance.

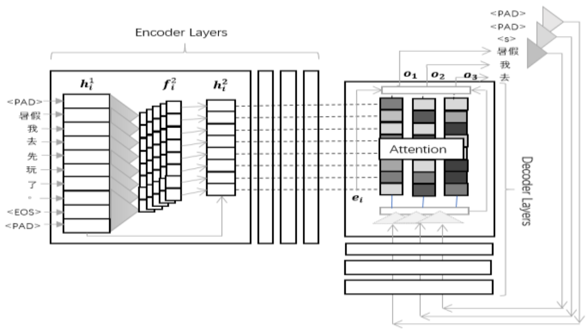

Fig. 1. deep CNN architecture of four-layer encoder-decoder network

3.2 Pinyin encoding

In Chinese speech recognition system, Chinese Pinyin as the main feature for speech to text conversion, Speech recognition error is also caused by Chinese pinyin error. There is evidence that the use of pinyin as an additional feature in Chinese-English translation of neural networks has an effect on translation performance [18]. So we put the Chinese Pinyin as feature of phonetic error correction. To study the impact of different pinyin coding schemes on error correction models, we designed different segmentation granularity: character segmentation, word segmentation. Different Pinyin coding schemes are designed according to different segmentation granularity. It be named as CPinyinCh, CPinyinChT, CInitials, CInitialsT, WPinyinCh WPinyinChT, WInitials, WInitialsT. these design in detail are:

CPinyinCh: Based on Chinese characters, each character of the pinyin string is encoded

CPinyinChT: Based on Chinese characters, each character of a pinyin string with tones is encoded.

CInitials: Based on Chinese characters, the phonetic alphabet is divided into initials and finals for encoding respectively.

CInitialsT: Based on Chinese characters, the pinyin with tone is divided into initial vowel and final vowel to be coded separately.

WPinyinCh: Token the sentence (using jieba) and encode each character of the word's pinyin string.

WPinyinChT： Token the sentence, and encode each character of the word's pinyin string with tone.

WInitials: Token the sentence, the phonetic strings of words are coded separately.

WInitialsT: Token the sentence, the phonetic strings of words with tones are coded separately.

The following are specific examples to illustrate different coding schemes:

Table 2. Pinyin coding

Source sentence：内个人是我妈妈。 Token：内 个人 是 我 妈妈 。
CPinyinCh： n e i g e r e n s h i w o m a m a 。
CPinyinChT：n e i 4 g e 4 r e n 2 s h i 4 w o 3 m a 1 m a 1 。
CInitials： n ei g e r en sh i w o m a m a 。
CInitialsT：n ei 4 g e 4 r en 4 sh i 4 w o 3 m a 1 m a 1 。
WPinyinCh： n e i | g e r e n|s h i | w o | m a m a 。
WPinyinChT：n e i 4 | g e 4 r e n 2 | s h i 4 | w o 3 | m a 1 m a 1 。
WInitials： n ei | g e r en | sh i | w o | m a | m a 。
WInitialsT：n ei 4 | g e 4 r en 4 | sh i 4 | w o 3 | m a 1 | m a 1 。

We spliced the embedded $\mathbf{w}(\boldsymbol{s_i})$ of $\boldsymbol{s_i}$ segmentation mentioned in 3.1 with the pinyin coding. The pinyin vector obtained by different pinyin coding schemes is denoted as $\mathbf{Pin}(\boldsymbol{s_i})$, the neural network input is $\boldsymbol{s_i} = [\mathbf{w}(\boldsymbol{s_i}); \mathbf{Pin}(\boldsymbol{s_i})] + \boldsymbol{P}(\boldsymbol{i})$, where $\boldsymbol{P}(\boldsymbol{i})$ is location coding.

4 Experiments

4.1 Datasets

We used two datasets, one is the WeChat speech transcription data set and the other is the public Chinese grammar correction (CGEC) data set. WeChat speech transcriptional data set collects WeChat speech and transcribes it into text through an open source speech recognition system ASRT[1]. The transcribed text is corrected by manual combination with recording. Chinese grammar correction data set is the data set given by NLPCC2018 Shared Task2 [19], which is the first parallel expected data set for Chinese grammar correction. Since Chinese grammatical errors contain spelling errors, the proposed model can be tested.

Among them, the WeChat transcribed data set contains 455,821 sentence pairs, which are randomly scrambled, and the data set is divided into three parts. The Validation set contained 60,772 sentence pairs, the training set contained 364,663 sentence pairs, and the test set contained 30,386 sentence pairs.

4.2 Evaluation

In the speech transcription data set of WeChat, because the errors mainly occurred in the local words of the sentence, we used the editing distance to measure. In other

[1] https://github.com/nl8590687/ASRT_SpeechRecognition

words, the sentences predicted by the model are converted into the same as tags after at least n times of deletion, addition and substitution of characters. The edit distance is n. For the Chinese grammar error correction data set, the source sentence may have multiple [2]correct error correction results. Therefore, edit distance is not suitable for syntax correction. We use the MaxMatch(M2) score as the evaluation index [20]. M2 is widely used in the evaluation of grammar correction. The main idea is to calculate the matching degree of the model output and tags at the phrase level. A sentence is first segmented before evaluation is carried out on a set of sentences. The metrics measured at the testing stage are: Precision, Recall and $F_{0.5}$.

4.3 Experimental configuration and training details

In the CNN error correction model, we extended the fairseq[2] model based on pytorch. The specific parameters are as follows: the encoder and decoder use a stack of 5 layers of convolutional networks with windows size of 3. Each layer of the decoder has an attention mechanism. The number of layers is set according to the performance of the network in the validation set. The output of encoder and decoder of each layer is 512 dimensions, dropout is 0.2. We train the model on 4 NVIDIA 1080Ti GPUs. The training epoch time is about 1.5 hours, and the beam width is set to 12 in the decoding stage.

4.4 Experimental results and analysis

Table 3. WeChat data set result

Model	Segmentation granularity	Number	Pinyin coding	Typo rate
Baseline	-	1	-	20.41%
Phrase translation model	phrase	2	-	18.82%
Deep CNN model	character	3	None	17.32%
		4	CPinyinCh	16.51%
		5	CPinyinChT	16.02%
		6	CInitials	17.16%
		7	CInitialsT	16.47%
Deep CNN model	word	8	None	16.85%
		9	WPinyinCh	16.47%
		10	WPinyinChT	16.27%
		11	WInitials	16.23%
		12	WInitialsT	16.18%

[2] https://github.com/pytorch/fairseq

Baseline refers to the error rate of speech recognition system, namely the ratio of the edit distance and the length of the corrected text after speech recognition and manual error correction, representing the effect of speech recognition system. "None" refers to no pinyin coding, only word (character) vector was used.

SMT NMT First, we compared the previously proposed error correction model based on phrase translation. Our best experimental results have increased by 14.8% (line 2 and 5). According to our analysis, 1) the modeling ability of phrase translation error correction model based on statistics in translation tasks is not as good as that based on neural network, which also verifies that NMT proposed by Mahata S K [21] is superior to SMT in large data sets. In the industry, Google, baidu, youdao and others have migrated their translation models to NMT. 2) the phrase translation model first converts Chinese characters into pinyin for error correction, and then converts the corrected pinyin into Chinese characters. It is easy to make mistakes in translating pinyin into Chinese characters. We use word (character) vector and pinyin coding feature, which not only solve the problem of errors in speech recognition due to pinyin errors, but also directly "translate" Chinese characters to reduce the error of pinyin into Chinese characters.

Pinyin In order to verify the validity of the proposed pinyin as an additional code, we designed a comparison experiment using only word (character) vectors and additional pinyin features. It is worth noting that the difference in the experimental results is due to the dimension of the input features. Differently caused (we tend to think that the larger the feature vector dimension is, the better the effect), we train the different dimensional word vectors (350-D and 300-D) in the same corpus, we splicing the 50-D Pinyin vector into 300-D. The dimensions of the different word vectors are guaranteed to be the same. The experimental results show that the Pinyin coding has a significant improvement effect under different granularity divisions, which fully proves our idea: in the speech recognition text error correction task, the Pinyin feature is important.

Tone In deep CNN model, among different encoding schemes, the encoding scheme with tone is always significantly better than the encoding scheme without tone (compared with the relative improvement of 2.9% in line 5 and 4). It is not difficult to understand that tone plays a very important role in putonghua. The commonly used Chinese characters are about 3000 words, and there are about 400 syllables without tone. On average, one syllable corresponds to about 7 Chinese characters, which is very easy to be confused.

Word Character The best experimental result of segmentation is that in the case of character segmentation, the word segmentation effect of the word splitter is not good when there are spelling errors in the source sentences, which increases the difficulty of error correction. The errors in the source sentences are mainly errors of local words, and word segmentation will lead to "distorted" results. Since most Chinese characters are misspelled, it seems more reasonable to use Chinese characters for segmentation.

In the Chinese grammar correction data set, the results in the test set are shown in table 4

Table 4. Chinese grammar correction dataset results

Model	Number	P	R	F0.5
CNN	1	21.28	11.36	18.11
CNN+CPinyinChT	2	21.61	11.49	18.37
CNN+WInitialsT	3	21.84	11.69	18.61

In the data set of Chinese grammar correction, our purpose is not to get the SOTA model, but to verify that adding pinyin features is also helpful for grammar correction, that is, pinyin features have certain generalization ability. CNN refers to the deep CNN model with attention mechanism that only use word vectors as features. In the second line, word vectors and pinyin alphabet coding schemes are used as features. In the third line, word vectors and vowel codes are used as features. Comparing lines 1 and 2, lines 1 and 3, we can see that the scheme with the addition of Pinyin coding improves the effect to varying degrees. We see that the finer-grained word segmentation scheme is worse than the word segmentation scheme. We analyze it is believed that Chinese grammar correction is more focused on a higher level of semantic level, while speech text correction is more focused on lower level word levels, words are the smallest morpheme units, and word segmentation reduces the length of sentences. Chinese grammar correction also contains spelling errors but mainly grammatical errors. Therefore, in Chinese grammar correction, models that achieve better results [22, 23] use word vectors instead of character vectors.

4.5 Boosting

Data Augmentation Theoretically, on large data sets, the neural network error correction model will have better results, and expanding the WeChat transcribed data set is expensive. We propose a data enhancement method, since we regard the error correction task as a "translation" task. We can train a reverse model, that is, the model input is the corrected sentence. The tag is a sentence with errors, and the error sentence generated by the reverse model is added to the original data set, and the original data set is expanded to 655821 pairs.

Language model (LM) In order to further improve the effect of our proposed error correction model, we improve the error correction capability by adding a 3-gram-based language model error corrector. The language model is widely used in Chinese spell checking. We use the kenlm[3] language model training tool to train the 3-gram speech model on the Baidu Encyclopedia dataset. First, the jieba tokenizer is used to segment the sentence. The sentence contains spelling errors. As a result, there is a case where a segmentation error occurs, so errors are detected from both the word granularity and the character granularity, and the two granularity-like erroneous results are integrated, and the error position candidate set is obtained, and then all the suspected error positions are traversed, and the sound shape dictionary is replaced. The wrong position local word, the sentence confusion score is calculated by training the good speech model, and the candidate set results are sorted to obtain the optimal word.

The experimental effects on the WeChat speech transcription data set by adding various schemes are shown in table 5.

Table 5. boosting result

Model	Typo rate
CNN CPinyinChT	16.02%
CNN CPinyinChT + Data Augmentation	15.21%
CNN CPinyinChT + Data Augmentation + Language model	15.07%

Experimental analysis showed that after adding data augmentation and 3-gram language model, the model was significantly improved. Finally, our experimental result showed that the error rate was 15.07%, which was 26.2% lower than Baseline.

5 Summary

In this paper, a text error correction model based on pinyin is proposed. Compared with the Chinese speech recognition error correction algorithm based on phrase translation model proposed previously, our model is improved by 20%. On WeChat transcribed dataset, our model makes the typo rate reduced 26%, we verify the effectiveness of the pinyin as coding, at the same time, we confirmed the tones in the importance of the speech recognition, confirmed in speech text correction, fine-grained character segmentation is better than word segmentation. We propose a reverse-transformed data augmentation method that incorporates a language model and eventually achieves a good level. Future work integrates word（character) vector and pinyin coding to further enhance the model's effects.

[3]https://github.com/kpu/kenlm

References

1. Ng, Hwee Tou, et al. The CoNLL-2013 shared task on grammatical error correction. Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task. 2013: 15-23.
2. Brockett C, Dolan W B, Gamon M. Correcting ESL errors using phrasal SMT techniques[C]//Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics. Association for Computational Linguistics, 2006: 249-256.
3. Susanto R H, Phandi P, Ng H T. System combination for grammatical error correction[C]//Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). 2014: 951-962.
4. Chollampatt S, Ng H T. Connecting the dots: Towards human-level grammatical error correction[C]//Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications. 2017: 327-333.
5. Yuan, Zheng, and Ted Briscoe. Grammatical error correction using neural machine translation. Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. 2016.
6. Ji J, Wang Q, Toutanova K, et al. A nested attention neural hybrid model for grammatical error correction[J]. arXiv preprint arXiv:1707.02026, 2017.
7. Chollampatt S, Ng H T. A multilayer convolutional encoder-decoder neural network for grammatical error correction[J]. arXiv preprint arXiv:1801.08831, 2018.
8. Junczys-Dowmunt M, Grundkiewicz R, Guha S, et al. Approaching Neural Grammatical Error Correction as a Low-Resource Machine Translation Task[J]. arXiv preprint arXiv:1804.05940, 2018.
9. Grundkiewicz R, Junczys-Dowmunt M. Near Human-Level Performance in Grammatical Error Correction with Hybrid Machine Translation[J]. arXiv preprint arXiv:1804.05945, 2018.
10. Yu L C, Lee L H, Chang L P. Overview of grammatical error diagnosis for learning Chinese as a foreign language[C]//Proceedings of the 1st Workshop on Natural Language Processing Techniques for Educational Applications (NLP-TEA 2014). 2014: 42-47.
11. Zheng B, Che W, Guo J, et al. Chinese Grammatical Error Diagnosis with Long Short-Term Memory Networks[C]//Proceedings of the 3rd Workshop on Natural Language Processing Techniques for Educational Applications (NLPTEA2016). 2016: 49-56.
12. Xie P. Alibaba at IJCNLP-2017 Task 1: Embedding Grammatical Features into LSTMs for Chinese Grammatical Error Diagnosis Task[J]. Proceedings of the IJCNLP 2017, Shared Tasks, 2017: 41-46.
13. Tan yongmei,Yang yixiao,yanglin,et.al. Grammatical Error Correction Using LSTM and N-gram[J]. Journal Of Chinese Information Processing,2018,32(6):19-27.
14. Hu yi, Liu Yunfeng, Yang Haisong, et al. An Online System for Chinese Query Correction in Search Engine[J]. Journal Of Chinese Information Processing, 2016, 30(1): 71-79.

15. Wu long. Chinese Speech Recognition Results Correction Based on Phrase-based Statistical Machine Translation Model [C]//The 14th national conference on human-computer voice communication.2017:6.
16. Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need[C]//Advances in neural information processing systems. 2017: 5998-6008.
17. Srivastava N, Hinton G, Krizhevsky A, et al. Dropout: a simple way to prevent neural networks from overfitting[J]. The Journal of Machine Learning Research, 2014, 15(1): 1929-1958.
18. Du J, Way A. Pinyin as Subword Unit for Chinese-Sourced Neural Machine Translation[C]//AICS. 2017: 89-101.
19. Zhao Y, Jiang N, Sun W, et al. Overview of the nlpcc 2018 shared task: Grammatical error correction[C]//CCF International Conference on Natural Language Processing and Chinese Computing. Springer, Cham, 2018: 439-445.
20. Dahlmeier D, Ng H T. Better evaluation for grammatical error correction[C]//Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. Association for Computational Linguistics, 2012: 568-572.
21. Mahata S K, Mandal S, Das D, et al. SMT vs NMT: A Comparison over Hindi & Bengali Simple Sentences[J]. arXiv preprint arXiv:1812.04898, 2018.
22. Ren H, Yang L, Xun E. A Sequence to Sequence Learning for Chinese Grammatical Error Correction[C]//CCF International Conference on Natural Language Processing and Chinese Computing. Springer, Cham, 2018: 401-410.
23. Zhou J, Li C, Liu H, et al. Chinese grammatical error correction using statistical and neural models[C]//CCF International Conference on Natural Language Processing and Chinese Computing. Springer, Cham, 2018: 117-128.