String Editing based Chinese Grammatical Error Diagnosis

Haihua Xie[1], **Xiaoqing Lyu**[2], **Xuefei Chen**[3]
[1]Yanqi Lake Beijing Institute of Mathematical Sciences and Applications / Beijing, China
[2]Wangxuan Institute of Computer Technology, Peking University / Beijing, China
[3]AI Research Center, Cloopen Group Holding Ltd / Beijing, China
haihua.xie@bimsa.cn, lvxiaoqing@pku.edu.cn, cxfff168@163.com

Abstract

Chinese Grammatical Error Diagnosis (CGED) suffers the problems of numerous types of grammatical errors and insufficiency of training data. In this paper, we propose a string editing based CGED model that requires less training data by using a unified workflow to handle various types of grammatical errors. Two measures are proposed in our model to enhance the performance of CGED. First, the detection and correction of grammatical errors are divided into different stages. In the stage of error detection, the model only outputs the types of grammatical errors so that the tag vocabulary size is significantly reduced compared with other string editing based models. Secondly, the correction of some grammatical errors is converted to the task of masked character inference, which has plenty of training data and mature solutions. Experiments on datasets of NLPTEA-CGED demonstrate that our model outperforms other CGED models in many aspects. The project of our approach is available at https://github.com/xiebimsa/se-cged.

1 Introduction

The traditional methods of Grammatical Error Diagnosis (GED) are pipeline-based, which has three main steps in series, including error detection, selection of candidate corrections, and ranking of candidate corrections (Lee et al., 2015). With the wide application of seq2seq models, seq2seq based GED models are designed to combine error detection and error correction (Wan et al., 2020; Kiyono et al., 2019). Considering the low efficiency of decoding in seq2seq and the uncontrollability of the generated results, researchers tend to apply the mode of sequence tagging for GED which outputs the correction operations instead of direct correction results.

The sequence tagging based GED methods are designed based on edit distance algorithms, such as Levenshtein distance (Levenshtein, 1966), which produces edit operations that are used to transform one sentence to the other. Because the intersection of an ungrammatical sentence and the corrected sentence is often large, it is feasible to transform them to each other based on a few edit operations. Several string editing based GED methods for English have been recently proposed, such as LaserTagger (Malmi et al., 2019), PIE (Awasthi et al., 2019) and GECToR (Omelianchuk et al., 2020).

The string editing based GED methods are highly accurate, controllable and interpretable (Rozovskaya and Roth, 2021; Parnow et al., 2021). Fine-grained data processing and edit tags designing are crucial. However, the size of the tag vocabulary is generally very large, which makes the current string editing based GED methods work better on closed sets, but difficult to deal with the complexities and long-tail problems on open datasets (Omelianchuk et al., 2020).

The current string editing based GED methods are mostly designed for processing English texts. Compared with English, CGED has to deal with more types of errors with less training samples (Li and Shi, 2021). Furthermore, it is difficult to detect Chinese grammatical errors based on explicit linguistic features due to the complexity and ambiguity of Chinese grammars (Rao et al., 2018). Chinese grammatical errors can be corrected by text edit operations in theory, but there are currently no papers detailing how to detect and correct Chinese grammatical errors based on string editing.

In this paper, we propose a string editing based CGED model, named SE-CGED, which detects and corrects Chinese grammatical errors based on text edit operations. The brief description of SE-CGED and the contributions of our work are presented below.

1) In SE-CGED, the process of CGED is separated into three main stages, including Seq2Edit, string editing, and masked charac-

ter inference. Detection of grammatical errors, and correction of different types of errors are handled in different stages.

2) Because the detection and correction of grammatical errors are separated, the size of the edit tag vocabulary is significantly smaller than that of other string editing based models. The smaller size of the tag vocabulary reduces the difficulty of sequence tagging and requires less training samples.

3) Based on the difficulty of modification, the correction of grammatical errors is separated into two steps to deal with different types of errors. The correction of some types of grammatical errors is uniformly converted to the task of masked character inference.

4) The existing methods and training data of masked language model are utilized for masked character inference, which helps improve the performance of CGED for certain error types.

This paper is organized as follows: Section 2 briefly reviews the literature on grammatical error diagnosis and CGED. Section 3 presents the basic concepts and the workflow of string editing based Chinese grammatical error diagnosis. Section 4 explains the model training and prediction of our proposed approach. Section 5 demonstrates the experimental setting and the experiment results. Section 6 discusses the limitations of our methodology and potential speculations of our works.

2 Background and related works

The target of a GED system is to give a correction for a sentence with grammatical errors. The seq2seq generative model is well suited for the GED task. Models of machine translation have been adopted in GED and achieved good results (Hotate et al., 2020). Different from machine translation, the language of the source sentence and the target sentence in GED is the same, and there is a large intersection between the two sequences. Therefore, the copy mechanism was applied for GED to reduce the computation of text generation (Zhao et al., 2019). Lack of training data is a major bottleneck for GED systems. To supplement training data, Ge et al., 2018 tried to use the n-best results of the seq2seq model to produce more pairs of correct and incorrect sentences, and Zhou et al., 2020 constructs training samples based on the uneven results of different translation models. However, the man-made training samples are significantly different from the real grammatical cases.

The mode of sequence tagging is recently prevailing for GED because it is more efficient in decoding and can produce more controllable results, compared with seq2seq models (Sun et al., 2021). Malmi et al., 2019 proposes LaserTagger, a general sequence tagging approach that casts text generation as a text editing task. LaserTagger sets three types of tags, including ‘keep’, ‘delete’ and ‘P’. ‘P’ can be a word or a phrase that should be added before the current position. Awasthi et al., 2019 proposes PIE, a sequence tagging based GED model. In PIE, the types of edit operations include ‘copy’, ‘append’, ‘delete’ and ‘replacement’. In the operations ‘append’ and ‘replacement’, the specific words that used to append or replace should be indicated. An iterative refinement sequence annotation method is proposed to predict the token level edit operations. Omelianchuk et al., 2020 proposes GECToR, a simple and efficient GED sequence tagger. GECToR specifically defines several tags of grammatical edit operations. The tag vocabulary size is 5000, including 4971 basic transformations (token-independent KEEP, DELETE and 1167 token-dependent APPEND, 3802 REPLACE) and 29 token-independent g-transformations.

3 Introduction of the SE-CGED model

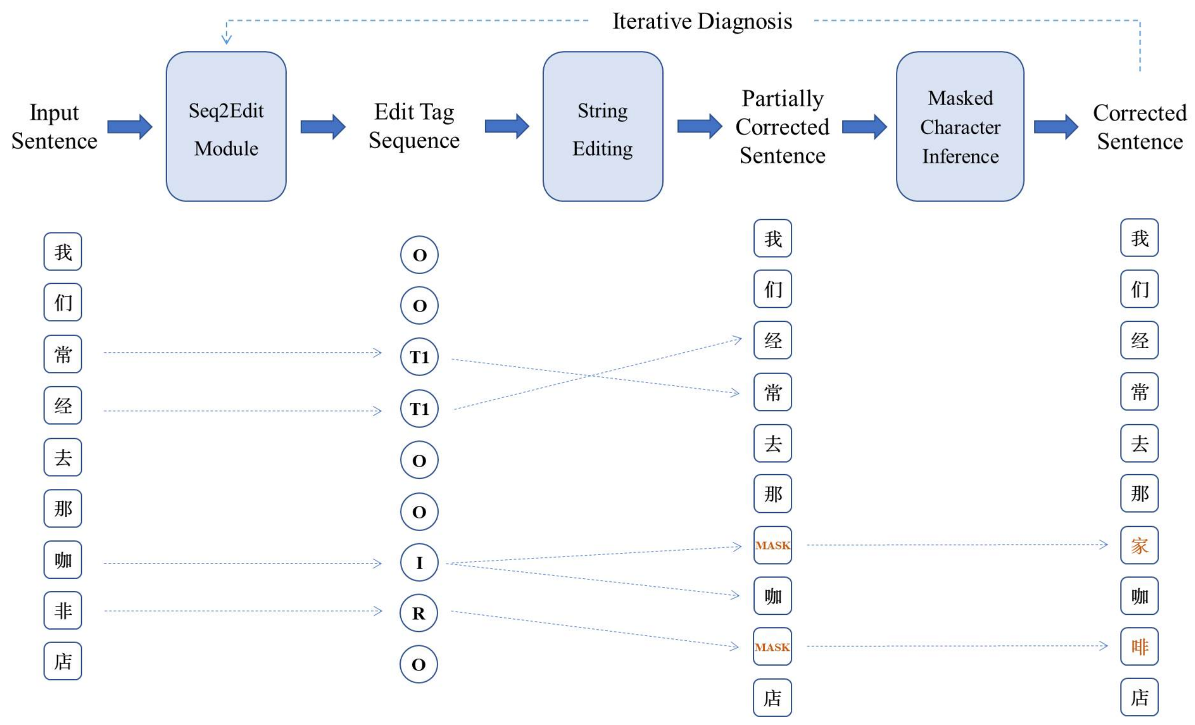
3.1 The workflow of SE-CGED

As shown in Figure 1, the workflow of SE-CGED has the following main stages.

1) Seq2Edit. Seq2Edit is a sequence tagging model that outputs a sequence of edit tags for a sentence. Each edit tag indicates the edit operation that should be performed on the corresponding token in the input sentence.

2) String editing. This step is to perform string editing based on the tag sequence output in Seq2Edit. According to the types of tags (i.e., types of errors), different operations are performed on the tokens. The tokens can be deleted or transposed, and the tag ‘MASK’ may be added at certain positions.

3) Masked character inference. This step is to infer what the masked characters should be.



Iterative Diagnosis
Input Sentence
Seq2Edit Module
Edit Tag Sequence
String Editing
Partially Corrected Sentence
Masked Character Inference
Corrected Sentence
我 们 常 经 去 那 咖 非 店
O O T1 T1 O O I R O
我 们 经 常 去 那 MASK 咖 MASK 店
我 们 经 常 去 那 家 咖 啡 店

Figure 1: Workflow of SE-CGED

Grammar Error	Edit Operation	Edit Tags	Other Tags
Character redundancy	Deletion	D	/
Misuse of character	Replacement	R	R2
Character missing	Insertion	I	I2
Character disorder	Transposition	T1, T2, I-T2	/

Table 1: Grammatical error types and the corresponding edit operations in SE-CGED.

4) Iterative correction. In some cases, more than one iteration of the above steps may be performed to get the final corrections.

3.2 Error types and edit operations

The edit operations are critical for string editing based methods. The types of edit operations in SE-CGED are shown in Table 1. Because Chinese character is the basic unit of Chinese sentences, we use 'character' to represent Chinese character in the following chapters unless otherwise specified.

Error types: Chinese grammatical errors are categorized into four types.

1) Character redundancy.

In SE-CGED, redundant characters are labeled with 'D'. In the step of string editing, the characters labeled with 'D' will be directly deleted.

2) Misuse of character.

A misused character should be replaced by another character. In SE-CGED, misused characters are labeled with 'R'. In the step of string editing, a character labeled with 'R' is converted to a 'MASK' tag. The correction of character misusing is performed in the step of masked character inference. In some cases, a character should be replaced by a word consisting of more than one characters. However, the tag 'R' doesn' t indicate the number of characters in the substitute. This problem is handled in the stage of iterative correction.

3) Character missing.

The missing word should be inserted into the sentence. The token before which a word should be inserted is labeled with 'I'. Similar to character misusing, the correction of character missing is performed in the step of string editing, masked character inference and iterative correction.

4) Character disorder.

A character in a wrong position should be moved to a right position. There are two types of character disorder.

a) Disorder of characters within a word or a phrase, such as “特普朗” (disorder of “特朗普” (‘Trump’)). A character with such type of disorder is labeled with ‘T1’ (transposition type 1).

The corresponding correction is to swap the positions of the first half and the second half of the sequence labeled with ‘T1’, and it is done in the step of string editing. Therefore, the number of consecutive tokens labeled with ‘T1’ is even.

b) Improper collocation between words.

Each character in a word with such type of disorder is labeled with ‘T2’ (transposition type 2). To give a direct correction suggestion for such error, a tag ‘I-T2’ is given to the token before which the words labeled with ‘T2’ should be moved to. For instance, the tagging sequence of “我去了公园在中午。” (“I went to the park in the noon.”) is ‘O / I-T2 / O / O / O / T2 / T2 / T2 / O’.

In the step of string editing, the subsequence labeled with ‘T2’ will be moved to the position before the token labeled with ‘I-T2’.

In the training samples, the tag ‘T2’ and ‘I-T2’ always appear together or neither in a tag sequence. To ensure that ‘T2’ and ‘I-T2’ also appear together or neither in the tagging result of a trained Seq2Edit model, the decoding module of Seq2Edit is designed to output the top scoring tag sequence which contains both of ‘T2’ and ‘I-T2’ or neither.

Separation of detection and correction: The correction of character misusing and character missing is not performed in the step of string editing. Instead, the character labeled with ‘R’ (or ‘I’) is replaced (or added) with a ‘MASK’ tag. And in the step of masked character inference, the replacement of the misused character (or the character to be inserted) will be given.

In the existing string editing based GED models, the detection and correction of grammatical errors are both performed in the step of generating edit tags. If a misused word is detected, the edit tag indicates the word that is used to replace it. There are two main drawbacks of such mode.

- The size of the tag vocabulary is huge, thus a large amount of training samples are required.
- The correction options are limited.

The above problems can be overcome in SE-CGED. Because Seq2Edit only gives the types of grammatical errors, the size of the tag vocabulary is only 7. The masked character inference based correction provides more options for correcting the errors of character missing and misusing.

Iterative correction: Iterative correction is performed for three reasons. 1) some errors may not be detected in one iteration. 2) new errors may occur after an iteration of correction. 3) character replacement and character insertion may need many iterations to obtain the result. A maximum number of iterations can be set, such as 3.

The first two reasons are easy to understand. For the third one, we take character misusing as an example. In practical scenarios, a misused character sometimes should be replaced by two or more characters. For example,

- Incorrect: <u>跟</u>两个方法的比较 (“<u>With</u> the comparison of the two methods”).
- Correct: <u>通过</u>两个方法的比较 (“<u>Based on</u> the comparison of the two methods”).

“跟” (‘with’) is replaced by “通过” (‘based on’). However, the tag ‘R’ doesn’t contain the information of how many characters should be used to replace the misused character. To deal with such problem, a mechanism of iterative correction is applied.

i. The token labeled with ‘R’ at the position *p* is converted to ‘MASK’ in string editing.
ii. Masked character inference is performed, and a corrected sentence is obtained.
iii. Seq2Edit is performed again on the corrected sentence.
iv. If an error still occurs at the position *p*.
    a. The character is labeled with ‘R2’.
    b. The token labeled with ‘R2’ is converted to ‘MASK MASK’ in string editing.
    c. Masked character inference is performed, and a new corrected sentence is obtained.

Based on the procedure of iterative correction, if the first iteration gives a wrong correction (for instance, ‘跟’ is replaced by ‘和’), a second iteration is run to replace the character with two characters. Replacing a character with one or two characters can cover most of the cases of character misusing. Thus, we do not try to replace a character with more than two characters.

Tag vocabulary: The tag vocabulary of Seq2Edit in SE-CGED contains 7 main tags, including 'D', 'R', 'I', 'T1', 'T2', 'I-T2' and 'O'. The other two tags, 'R2' and 'I2', are used in the stage of iterative correction, and they are not considered in Seq2Edit. Compared with the previous Seq2Edit models, such as GECToR (Omelianchuk et al., 2020), the size of tag vocabulary of SE-CGED is significantly reduced.

3.3 Converting the tags of Damerau Levenshtein to the tags of Seq2Edit

```
Seq2Edit Tag Generation based on DL distance
Require: x = ($x_1$, …, $x_n$), y = ($y_1$, …, $y_n$)
diffs ← DamerauLevenshtein-Dist (x, y)
for each insert(a, b) tag in diffs
    s = all tokens inserted before a
    if token a is replaced by a'
        s = s + a'
        token a is labeled with 'D'
    if s is deleted or replaced
        token a is labeled with 'I-T2'
        tokens in s are labeled with 'T2'
    otherwise, token a is labeled with 'I'
for each replace(a, b) tag in diffs
    if token a has been labeled, skip
    otherwise
        s = a + following tokens labeled with 'replace'
        if the front & rear of s are mutually replaced
            tokens in s are labeled with 'T1'
        otherwise, token a is labeled with 'R'
for each delete(a, b) tag in diffs
    if token a has been labeled, skip
    otherwise, token a is labeled with 'D'
for each swap(a, b) tag in diffs
    if token a has been labeled, skip
    otherwise, token a and b are labeled with 'T1'
for each token t in x
    if t is not labeled
        t is labeled with 'O'
```

Figure 2: Seq2Edit tag generation based on Damerau Levenshtein distance.

Most of the training samples of CGED only contain parallel pairs of incorrect and correct sentences. Some training data (Rao et al., 2020) gives the steps of corrections, but they are incompatible with the tagging system of our approach.

To prepare the training samples of the Seq2Edit module, we first apply the Damerau Levenshtein algorithm (Brill and Moore, 2000) to analyze the difference between ungrammatical sentences and the corresponding correct sentences, and generates DL tags which indicates the steps of modifying the ungrammatical sentence to obtain the correct one. Then we convert the DL tags to the tags of Seq2Edit.

There are four types of Damerau Levenshtein tags, which are 'delete', 'replace', 'insert' and 'swap'. Some examples of these four tags are given below (suppose the following operations are used to transform sentence A to sentence B).

- ('delete', 5, 5), to delete the fifth token in A.
- ('replace', 5, 6), to replace the fifth token in A with the sixth token in B.
- ('insert', 5, 6), to insert the sixth token in B before the fifth token in A.
- ('swap', 5, 6), to swap the positions of the fifth token and the sixth token in A.

The pseudo code of converting the DL tags to the Seq2Edit tags is shown in Figure 2. In the conversion process, a token may be given more than one tags. However, in a training sample of Seq2Edit, a token can only have one tag. To deal with this problem, an intermediate transformation is performed to split one parallel pair of incorrect and correct sentences into two or more training samples of Seq2Edit, to ensure there is only one tag for each token. The following steps are performed to generate the training samples of Seq2Edit based on a parallel pair of incorrect and correct sentences.

For ($\boldsymbol{x}$, $\boldsymbol{y}$), an ungrammatical sentence $\boldsymbol{x}$ and the corresponding correction $\boldsymbol{y}$.

(1) Run the algorithm of Seq2Edit Tag Generation based on $\boldsymbol{x}$ and $\boldsymbol{y}$.

(2) If there is a token in $\boldsymbol{x}$ has more than one tags,

   a. Generate a training sample ($\boldsymbol{x}$, $\boldsymbol{t_1}$), $\boldsymbol{t_1}$ is the set of first tags of each token,
   b. Modify $\boldsymbol{x}$ based on tags in $\boldsymbol{t_1}$,
   c. Rerun the process from step (1).

(3) Otherwise, let ($\boldsymbol{x}$, $\boldsymbol{t}$) be a training sample, $\boldsymbol{t}$ is the set of tags of each token.

4 Training and application of SE-CGED

4.1 Generation of the training data

Training data of Seq2Edit: According to the description in the previous section, a part of training data of Seq2Edit can be generated based on the Damerau Levenshtein distance of the existing training samples of CGED. The other part of training data of Seq2Edit is generated through data augmentation. Different types of synthetic ungrammatical sentences are created based on our summarized patterns of grammatical errors as shown in Table 2.

1) Character missing and character redundancy. Several patterns of character missing and character redundancy are summarized based on the real training samples. Table 2 gives three example patterns. For instance, for the first example pattern of character missing, a synthetic ungrammatical sentence is created by removing the character ‘了’ after a verb. Before creating a synthetic sentence, word segmentation is performed on the original sentence. We do not remove a character within a word for creating an error of character missing.

2) For the type of character misusing, a part of synthetic samples are created by replacing a word with its synonyms, selected from a synonyms toolkit which is based on word2vec (Mikolov et al., 2013) similarity calculations, and the others are created by replacing a character with another character with the similar pronunciation or shape, selected from a prepared confusion set.

3) For the type of character disorder, most of the synthetic samples are created by changing the position of one word, and the others are created by swapping the positions of two adjacent characters within a word.

Training data of masked character inference: The training of masked character inference is performed in the step of masked LM pretraining of BERT (Devlin et al., 2019). The training samples of word replacement and word insertion are used here as part of training samples.

Other training samples are produced by randomly masking words in sentences. One-character words and two-character words are randomly selected and masked. Besides, two consecutive words with a high PMI value can also be masked together. Given two tokens w_1 and w_2, the PMI of the bigram ‘$w_1 w_2$’ is:

$$PMI(w_1 w_2) = \log \frac{p(w_1 w_2)}{p(w_1)p(w_2)} \qquad (1)$$

$p(w)$ is the probability of the occurrences of token w in the corpus.

4.2 Model training and model inference

Model of masked character inference: The task of masked character inference is performed based on the masked language model (MLM) of BERT. During model pre-training of BERT, the task of masked character inference is performed, but the task of next sentence prediction is skipped.

MLM consists of a multi-layer transformer encoder and a feed-forward neural network. At the stage of inference, the partially masked sequence is fed into MLM to produce a $1{*}V$ vector $\{C_1, C_2, ..., C_{V-1}, C_V\}$ at each position with token ‘[MASK]’. C_i is the confidence value of fitting the ith character in the vocabulary to the targeted position calculated in equation (2).

$$Conf(e_i = c|\vec{x}) = softmax(W_c^T h_i) \qquad (2)$$

e_i is a masked character in $\vec{x}$ and c is a character in the vocabulary. h_i is the multi-layer transformer embedding of e_i, and W_c is a matrix of parameters regarding c in the feed-forward neural network.

Model of Seq2Edit: The Seq2Edit model is built on BERT for sentence embedding and a CRF layer for sequence tagging. The BERT model used here is the same as that used in masked character inference, and the CRF layer is trained based on the training data of Seq2Edit to learn how to detect and identify grammatical errors in a sentence.

Seq2Edit is to output a tag sequence L for the input sentence S. Let $S = \{s_1, s_2, ..., s_K\}$ and K is the length of S. $U = \{u_1, u_2, ..., u_K\}$ is the embedding result of S from BERT, and u_i is the embedding of s_i. U is input into the CRF layer and $L = \{l_1, l_2, ..., l_K\}$ is the tagging result. l_i is the tag for s_i. The confidence of labeling U with L is shown in the following equation.

$$P(L, U) = \sum_{k=1}^{K} (H_{(l_{k-1}, l_k)} + \varphi(l_k, u_k)) \qquad (3)$$

H is the probability transition matrix of tags. H is 7*7 matrix because there are seven tags in the tag set. $H_{(l_{k-1}, l_k)}$ is the transition probability from tag l_{k-1} to tag l_k. Additionally, $\varphi(l_k, u_k)$ is the score of labeling u_k as l_k. φ is a $7{*}V$ matrix where V is the size of the vocabulary and 7 is the size of the tag set. Both H and φ are randomly initialized and updated during the training process.

Error Type	Ratio in real training data	Example patterns of grammatical errors
Character missing	0.275	1) Missing ‘了’ after a verb or at the end of a sentence 2) Missing pronoun at the beginning of a sentence 3) Missing a conjunction word
Character redundancy	0.234	1) Reduplication of an adjective, verb, pronoun or preposition 2) Redundant ‘是’ before an adjective or adverb 3) Redundant ‘了’ after a verb
Misuse of character	0.422	1) Misuse of synonyms 2) Misuse of characters with similar pronunciation or shape
Character disorder	0.069	1) Wrong position of an adverb, conjunction or preposition 2) Wrong order of adjacent verbs and adverbs 3) Disorder of adjacent characters within a verb, adjective or noun

Table 2: Example patterns of synthetic grammatical errors.

Model	FPR	Detection Level			Identification Level			Position Level			Correction Level (Top1 & Top3)		
		Pre.	Rec.	F1	Pre.	Rec.	F1	Pre.	Rec.	F1	Pre.	Rec.	F1
Flying	0.3257	0.9101	0.8800	0.8948	0.7320	0.6011	0.6601	0.4715	0.3536	0.4041	0.2290 0.2290	0.1575 0.1575	0.1867 0.1867
YD-NLP	**0.2182**	**0.9357**	0.8478	0.8896	**0.7711**	0.5577	0.6473	0.5011	0.2995	0.3749	**0.3386** **0.3217**	0.1259 0.1333	0.1836 0.1885
Orange-Plus	0.2606	0.9252	0.8600	0.8914	0.7230	0.6287	0.6726	0.4428	0.3610	0.3977	0.1780 0.0934	0.1536 0.2283	0.1649 0.1325
SE-CGED	0.2769	0.9111	**0.8962**	**0.9036**	0.7059	**0.6557**	**0.6799**	**0.5060**	**0.4645**	**0.4844**	0.2331 0.2417	**0.2077** **0.2186**	**0.2197** **0.2296**

Table 3: Performance of different models for CGED.

5 Experiments and Analysis

5.1 Datasets in the experiment

The dataset of NLPTEA-CGED evaluations is used in our experiment. The training sets and test sets of NLPTEA-CGED evaluations in the year of 2014 to 2021 are collected. The NLPTEA-CGED data provides parallel pairs of ungrammatical and correct sentences, and the ungrammatical samples are collected from essays written by learners of Chinese as a foreign language. A sentence in the dataset may not have grammatical errors, or it may have one or more errors.

In order to make a fair comparison with other methods, our model was trained and tested based on the training dataset and test dataset of the NLPTEA-CGED 2020 evaluation (Rao et al., 2020), and the performance was compared with the submitted results of the NLPTEA-CGED 2020 evaluation. Because the evaluation allows the participants to use external data, we pretrained our model based on the training data and test data in the NLPTEA-CGED evaluation of other years. There are totally 182,486 grammatical errors collected from NLPTEA-CGED evaluations.

Meanwhile, 247,185 synthetic training samples are created for Seq2Edit based on the data argumentation method introduced in section 4.1. The source sentences used for generating synthetic training samples are collected from *The People's Daily*[1]. The ratios of different types of synthetic grammatical errors are approximate to the ratios in real data shown in Table 2. Besides, we took several measures to make the distribution of synthetic data close to that of real data. For instance, the errors of character misusing in NLPTEA-CGED are significantly different from those made by Chinese native

[1]http://paper.people.com.cn

		FPR	Detection Level			Identification Level			Position Level			Correction Level (Top1 & Top3)		
			Pre.	Rec.	F1	Pre.	Rec.	F1	Pre.	Rec.	F1	Pre.	Rec.	F1
Itera-tions	2	0.2704	0.8556	0.8415	0.8485	0.6471	0.6011	0.6233	0.4464	0.4098	0.2730	0.1840	0.1639	0.1734
												0.1964	0.1776	0.1865
	4	0.2834	0.9091	0.9016	0.9053	0.7051	0.6598	0.6817	0.5052	0.4672	0.4855	0.2340	0.2090	0.2208
												0.2420	0.2202	0.2306
	5	0.2899	0.9089	0.9021	0.9055	0.7021	0.6611	0.6810	0.5031	0.4689	0.4854	0.2324	0.2115	0.2215
												0.2413	0.2217	0.2311
W/O synthetic data		0.2117	0.8958	0.8689	0.8821	0.7143	0.6284	0.6686	0.5161	0.4372	0.4734	0.2406	0.1940	0.2148
												0.2443	0.2036	0.2221

Table 4: Performance of the ablation study.

speakers, which usually occur between characters with the similar pronunciation or shape. Foreigners learning Chinese often misuse synonyms or relevant words. Thus, for the type of character misusing, 60% of the synthetic samples are created by replacing a word with its synonyms, and 40% are created by replacing a character with another character with the similar pronunciation or shape, selected from a prepared confusion set.

5.2 Evaluation metrics and comparison methods

The performance of grammatical error diagnosis systems is evaluated from the following aspects.

- Detection-level, to detect whether the statement contains grammatical errors.
- Identification-level, to determine the types of the grammatical errors.
- Position-level, to determine the position of the grammatical errors.
- Correction-level, to give the correction of the grammatical errors.

The measurement method of the above metrics can be referred to (Rao et al., 2020). We select three representative models with good performance in the NLPTEA-CGED 2020 evaluation as the comparison models, including Flying, YD-NLP and Orange-Plus. FLYing has the highest F1 score, YD-NLP has the highest precision score, and Orange-Plus are more balanced between the precision and recall rate. The maximum numbers of iterations in SE-CGED was set to 3. The performance of different models is shown in Table 3.

Our proposed model, SE-CGED, achieves a better performance in the identification level, and a much better performance in the position level and correction level. The performance of the position level does not decay much compared with that of the identification level because the Seq2Edit module gives the type and position of the grammatical errors at the same time. Meanwhile, our method is more balanced in terms of accuracy and recall, compared with the comparison methods, in which the recall rates are significantly lower.

5.3 Ablation analysis

The structure of our method is pipeline based, and all the stages are indispensable. Therefore, the ablation analysis in our experiment was performed by making minor adjustments to the model. Firstly, the maximum numbers of iterations was set to 2, 4 and 5 respectively in the stage of iterative correction. Secondly, synthetic data was not used to train the module of Seq2Edit. The results of ablation analysis are shown in Table 4.

When the number of iterations is set to 2, the performance is significantly reduced compared with that when the number of iterations is 3. The reason is that lots of errors have not been detected or corrected after two iterations. On the other hand, iterations more than 3 lead to a slight improvement in performance, but the improvement is very limited. Thus, it is reasonable to set the number of iterations to 3. Training the model without the synthetic data leads to lower recall rates. Out synthetic training data is helpful to detect errors that do not appear in the real training corpus.

6 Conclusions

In this paper, we proposed a phased string editing based approach for Chinese grammatical errors diagnosis. The detection and correction of grammatical errors are performed based on edit tags and masked character inference. In this way, different types of grammatical errors can be handled in the same way, and the training data is significantly reduced. Experiments on the dataset of NLPTEA-CGED evaluations show our proposed approach performs better than the traditional methods in performance and speed.

Because of the complexity of Chinese grammars, there are still many deficiencies in our current work, and the detection of error type and location is not accurate. In future research, we will further improve the rationality of data argumentation and make the synthetic error samples more consistent with practical grammatical errors. In addition, we will try to import linguistic features to further improve the detection and correction of Chinese grammatical errors.

Acknowledgements

This work is supported by the National Key Research and Development Program (No. 2019YFB1406302), the projects of National Natural Science Foundation of China (No. 61472014, No. 61573028 and No. 61432020), the Natural Science Foundation of Beijing (No. 4142023) and the Beijing Nova Program (XX2015B010). We also thank all the anonymous reviewers for their valuable comments.

References

Abhijeet Awasthi, Sunita Sarawagi, Rasna Goyal, Sabyasachi Ghosh, and Vihari Piratla. 2019. Parallel iterative edit models for local sequence transduction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4260–4270, Hong Kong, China. Association for Computational Linguistics.

Eric Brill and Robert C. Moore. 2000. An improved error model for noisy channel spelling correction. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pages 286–293, Hong Kong. Association for Computational Linguistics.

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, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Tao Ge, Furu Wei, and Ming Zhou. 2018. Fluency boost learning and inference for neural grammatical error correction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1055–1065, Melbourne, Australia. Association for Computational Linguistics.

Kengo Hotate, Masahiro Kaneko, and Mamoru Komachi. 2020. Generating diverse corrections with local beam search for grammatical error correction. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2132–2137, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Shun Kiyono, Jun Suzuki, Masato Mita, Tomoya Mizumoto, and Kentaro Inui. 2019. An empirical study of incorporating pseudo data into grammatical error correction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1236–1242, Hong Kong, China. Association for Computational Linguistics.

Lung-Hao Lee, Liang-Chih Yu, and Li-Ping Chang. 2015. Overview of the NLP-TEA 2015 shared task for Chinese grammatical error diagnosis. In *Proceedings of the 2nd Workshop on Natural Language Processing Techniques for Educational Applications*, pages 1–6, Beijing, China. Association for Computational Linguistics.

Vladimir I Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet physics doklady*, 10:707–710.

Piji Li and Shuming Shi. 2021. Tail-to-tail non-autoregressive sequence prediction for Chinese grammatical error correction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4973–4984, Online. Association for Computational Linguistics.

Eric Malmi, Sebastian Krause, Sascha Rothe, Daniil Mirylenka, and Aliaksei Severyn. 2019. Encode, tag, realize: High-precision text editing. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5054–5065, Hong Kong, China. Association for Computational Linguistics.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751, Atlanta, Georgia. Association for Computational Linguistics.

Kostiantyn Omelianchuk, Vitaliy Atrasevych, Artem Chernodub, and Oleksandr Skurzhanskyi. 2020. GECToR – grammatical error correction: Tag, not rewrite. In *Proceedings of the Fifteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 163–170, Seattle, WA, USA → Online. Association for Computational Linguistics.

Kevin Parnow, Zuchao Li, and Hai Zhao. 2021. Grammatical error correction as GAN-like sequence labeling. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3284–3290, Online. Association for Computational Linguistics.

Gaoqi Rao, Qi Gong, Baolin Zhang, and Endong Xun. 2018. Overview of NLPTEA-2018 share task Chinese grammatical error diagnosis. In *Proceedings of the 5th Workshop on Natural Language Processing Techniques for Educational Applications*, pages 42–51, Melbourne, Australia. Association for Computational Linguistics.

Gaoqi Rao, Erhong Yang, and Baolin Zhang. 2020. Overview of NLPTEA-2020 shared task for Chinese grammatical error diagnosis. In *Proceedings of the 6th Workshop on Natural Language Processing Techniques for Educational Applications*, pages 25–35, Suzhou, China. Association for Computational Linguistics.

Alla Rozovskaya and Dan Roth. 2021. How good (really) are grammatical error correction systems? In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 2686–2698, Online. Association for Computational Linguistics.

Xin Sun, Tao Ge, Furu Wei, and Houfeng Wang. 2021. Instantaneous grammatical error correction with shallow aggressive decoding. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5937–5947, Online. Association for Computational Linguistics.

Zhaohong Wan, Xiaojun Wan, and Wenguang Wang. 2020. Improving grammatical error correction with data augmentation by editing latent representation. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2202–2212, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Wei Zhao, Liang Wang, Kewei Shen, Ruoyu Jia, and Jingming Liu. 2019. Improving grammatical error correction via pre-training a copy-augmented architecture with unlabeled data. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 156–165, Minneapolis, Minnesota. Association for Computational Linguistics.

Wangchunshu Zhou, Tao Ge, Chang Mu, Ke Xu, Furu Wei, and Ming Zhou. 2020. Improving grammatical error correction with machine translation pairs. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 318–328, Online. Association for Computational Linguistics.