Improving Grammatical Error Correction with Data Augmentation by Editing Latent Representation

Zhaohong Wan[1,2] and **Xiaojun Wan**[1,2] and **Wenguang Wang**[3]
[1]Wangxuan Institute of Computer Technology, Peking University
[2]The MOE Key Laboratory of Computational Linguistics, Peking University
[3]DataGrand Tech Inc.
{xmwzh,wanxiaojun}@pku.edu.cn

Abstract

The incorporation of data augmentation method in grammatical error correction task has attracted much attention. However, existing data augmentation methods mainly apply noise to tokens, which leads to the lack of diversity of generated errors. In view of this, we propose a new data augmentation method that can apply noise to the latent representation of a sentence. By editing the latent representations of grammatical sentences, we can generate synthetic samples with various error types. Combining with some pre-defined rules, our method can greatly improve the performance and robustness of existing grammatical error correction models. We evaluate our method on public benchmarks of GEC task and it achieves the state-of-the-art performance on CoNLL-2014 and FCE benchmarks.

1 Introduction

Grammatical Error Correction (GEC) is a task of detecting and correcting grammatical errors in a sentence. Due to the growing number of language learners of English, there has been increasing attention to the English GEC in the past few years.

Considering the outstanding performance of neural network models in machine translation tasks, many studies have tackled GEC as a machine translation task. They regard ungrammatical sentences as the source language and grammatical sentences as the target language. This approach allows cutting-edge neural machine translation models to be applied to GEC. Many encoder-decoder models, such as recurrent neural network (RNN)(Graves et al., 2013), convolutional neural networks (CNN)(Kim, 2014), have been widely applied to GEC.

A challenge in applying neural machine translation models to GEC is the requirement of a large amount of training data, i.e., the source-target pairs. To address this problem, many data augmentation methods have been proposed. Existing methods, however, are often only able to generate sentences with limited error types, and can only improve the performance of GEC model on these few error types, while it is still hard for the model to correct the sentences with other types of errors.

To address the above problem, we propose a new data augmentation method to generate synthetic samples by editing the latent representations of grammatical sentences. Given a target grammatical error type and the corresponding grammatical error type classifier, we can get a perturbation vector in latent space. Then we add the perturbation vector to the latent representation of input sentence, and use a decoder to generate a sentence with target grammatical error type. In this way, diverse errors can be generated by assigning different target error types. To further improve the performance, we adopt some rules to assist the generation of some local grammatical errors, such as spelling errors, wrong punctuation, etc.

We apply this data augmentation method to the existing GEC model Copy-transformer (Zhao et al., 2019) to evaluate the results. Experiments are conducted on the following widely used benchmarks: CoNLL-2014 (Ng et al., 2014), FCE (Yannakoudakis et al., 2011), BEA-2019 (Bryant et al., 2019). Experimental results show the efficacy of our proposed method which outperforms several existing models.

Proceedings of the 28th International Conference on Computational Linguistics, pages 2202–2212
Barcelona, Spain (Online), December 8-13, 2020

Our contributions are summarized as follows:

1. We propose a new data augmentation method to generate synthetic samples by editing latent representations of grammatical sentences, which is able to generate errors with high quality and diversity.

2. Additional synthetic training samples enable training neural GEC model to detect and correct most error types, and improving the performance and robustness of the model.

3. Our method achieves the state-of-the-art performance on CoNLL-2014 and FCE benchmarks. It outperforms not only all previous single models but also all ensemble models. On BEA-2019 benchmark, our method achieves very competitive performance as well.

2 Related Work

Early GEC models are mainly based on manually designed grammar rules(Murata and Nagao, 1994; Bond et al., 1996; Siegel, 1996). Han et al. (2006) pointed out the limitation of rule-based method and proposed a statistical model. Later, some researchers proposed solutions based on statistical machine learning method(Knight and Chander, 1994; Minnen et al., 2000; Izumi et al., 2003).

With the development of deep learning, recent works proposed a variety of neural network models to deal with GEC task. Some treated the GEC task as a translation problem and applied neural machine translation model to detect and correct grammatical errors. Yuan and Briscoe (2016) used a classical bidirectional recurrent neural network (Graves et al., 2013) with attention. Chollampatt and Ng (2018) proposed a convolution neural network (Kim, 2014) to capture the local context. Many recent works (Junczys-Dowmunt et al., 2018) made use of the powerful machine translation architecture Transformer (Vaswani et al., 2017). Zhao et al. (2019) further applied copying mechanism(Gu et al., 2016; Jia and Liang, 2016) to Transformer. Considering the tremendous performance of pre-trained methods, pre-trained language model, such as BERT(Devlin et al., 2019), can be incorporated into the encoder-decoder model (Kaneko et al., 2020).

An encoder-decoder GEC model requires a large amount of training data, and the available training corpora usually failed to train a good GEC model. To address this problem, many data augmentation methods have been proposed (Ge et al., 2018). Many works adopted pre-defined rules to generate local grammatical errors. Grundkiewicz et al. (2019) applied a confusion set built by spellchecker to the corpus. Choe et al.(2019) extracted some common text editing operations from human writing habits and got synthetic samples by these extracted operations. Lichtarge et al.(Lichtarge et al., 2018) made use of models trained on large amounts of weakly supervised text. Inspired by back-translation procedure for machine translation (Sennrich et al., 2015) , Xie et al.(2018) proposed a model that can learn to generate erroneous sentences from correct ones. Based on this work, Kiyono et al. (2019) further studied the data augmentation methods and got some empirical conclusions.

3 Our Data Augmentation Method

Our data augmentation method applies noise to the latent space and it can generate sentences with various error types by editing latent representations of grammatical sentences. To further improve the performance, we adopt some rules to assist the generation of some local grammatical errors. Synthetic training samples generated from our method are used to train neural GEC model and enable the model to detect and correct most error types and improve its performance and robustness.

3.1 Editing Latent Representation

Inspired by the adversarial sample generation procedure (Goodfellow et al., 2015), we propose a data augmentation method that applies the noise by editing latent representations. Firstly, we train an encoder, a decoder and an error type classifier. Given the trained models, we can generate synthetic training samples by adding a perturbation vector to the latent representations of sentences. The overall framework of our method is shown in Figure 1.

3.1.1 Training Encoder, Decoder and Classifier

Firstly, we train an encoder and a classifier to deal with the grammatical error classification task. Given a ungrammatical sentence $\boldsymbol{x}$ and its corresponding error type z, we use the encoder ϕ_E to encode $\boldsymbol{x}$ to

its latent representation $\boldsymbol{h}_x$. Then we use the classifier C to get the prediction z'. This process can be formulated as following:

$$\boldsymbol{h}_x = \phi_E(\boldsymbol{x}) \tag{1}$$

$$z' = C(\boldsymbol{h}_x) \tag{2}$$

We denote the classification loss as $L(\boldsymbol{h}_x, z, z')$, where $\boldsymbol{h}_x$ is the latent representation, z' is the prediction label and z is the gold label. In our model, we choose the cross entropy loss as classification loss.

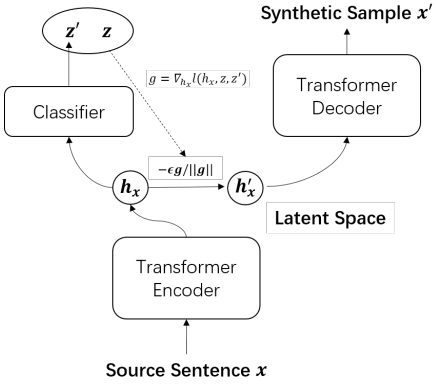

Figure 1: The overall framework of our proposed data augmentation method. It contains an encoder, a classifier, and a decoder. Given a source sentence $\boldsymbol{x}$, we first use the encoder to obtain the latent representation $\boldsymbol{h}_x$ of $\boldsymbol{x}$. We then pass $\boldsymbol{h}_x$ and specified error type z to the classifier to compute the classification loss and the direction in which the loss descends the most. Finally, we project $\boldsymbol{h}_x$ to this direction to get $\boldsymbol{h}'_x$ and decode $\boldsymbol{h}'_x$ into text to get the synthetic sample $\boldsymbol{x}'$.

With the encoder ϕ_E trained in previous process , we train decoder ϕ_D in an auto-encoder way. The goal is to minimize the negative log-likelihood between input $\boldsymbol{x}$ and output $\tilde{\boldsymbol{x}}$:

$$\tilde{\boldsymbol{x}} = \phi_D(\boldsymbol{h}_x) \tag{3}$$

$$J\left(\boldsymbol{h}_x\right) = -\sum_{t=1}^{L} \log P\left(\boldsymbol{x}_t | \tilde{\boldsymbol{x}}_{<t}, \boldsymbol{h}_x\right) \tag{4}$$

We choose the powerful Transformer(Vaswani et al., 2017) as encoder and decoder. Both the encoder and the decoder consist of Transformer blocks with multi-head self-attention layer followed by feed-forward layer.

As for the classifier, it has several feed-forward layers as the classification layers. The classifier will determine whether the sentence is correct. If not, it will predict the specific type of grammatical errors. We define six types of grammatical errors based on the 25 main types defined in automatic annotation tool ERRANT(Bryant et al., 2017): ADJ/ADV(ADJ, ADJ:FORM and ADV), DET, PREP, NOUN(NOUN, NOUN:INFL, NOUN:NUM and NOUN:POSS), VERB(VERB, VERB:FORM, VERB:INFL, VERB:SVA and VERB:TENSE), OTHER(Errors that do not fall into above categories). These error types are common in human writing and more difficult to be corrected by GEC model.

3.1.2 Generating Synthetic Training Samples

We want to add a perturbation vector r to the latent representation $\boldsymbol{h}_x$ of input sentence $\boldsymbol{x}$, and use decoder ϕ_D to generate additional training samples from $\boldsymbol{h}_x + r$. The algorithm for generating synthetic samples is summarized in Algorithm 1.

Given a correct sentence $\boldsymbol{x}$ and a target grammatical error type z, we can get an optimal perturbation vector $\hat{r}$ by minimizing the classification loss $L(\boldsymbol{h}_x + r, z, z')$. Besides, in order to prevent the outputs

Algorithm 1 Generating synthetic training samples

Input: Latent representation $\boldsymbol{h}_x$, target error type z, similarity discriminator ψ, classifier C,decoder ϕ_D, hyper parameter $\epsilon_0, \epsilon_{\max}, \lambda, t$

Output: Synthetic erroneous sample $\boldsymbol{x'}$ (paired with input $\boldsymbol{x}$)

Function: $Gen(\boldsymbol{h}_x, z, \epsilon_0, \epsilon_{\max}, \lambda, t)$:

Initialization: $\epsilon \longleftarrow \epsilon_0$, $\hat{r} \longleftarrow 0$.
Compute gradient: $\boldsymbol{g} \longleftarrow \nabla_{\boldsymbol{h}_x} L\left(\boldsymbol{h}_x, z, C\left(\boldsymbol{h}_x\right)\right)$
while $\epsilon \leq \epsilon_{max}$ **do**
    if $C(\boldsymbol{h}_x + \hat{r}) = z$ **then**
        break
    $\hat{r} \longleftarrow -\epsilon \boldsymbol{g} / ||\boldsymbol{g}||$
    $\epsilon \longleftarrow \lambda \epsilon$
$\boldsymbol{x'} = \phi_D(\boldsymbol{h}_x + \hat{r})$.
$p \longleftarrow \psi(\boldsymbol{x}, \boldsymbol{x'})$
if $p \geq t$ **then**
    return $\boldsymbol{x'}$
else
    return null

from changing too much, we restrict the L2 norm of perturbation r. This problem can be formulated as the following:

$$\hat{r} = \underset{r, \|r\| \leq \epsilon}{\operatorname{argmin}} \left\{ L(\boldsymbol{h}_x + r, z, z') \right\} \tag{5}$$

However, it is almost impossible to exactly estimate $\hat{r}$ in Eq5 for a deep neural network model. Following the method of Goodfellow et al. (2015), we apply the linearization technique by linearizing loss function $L(\boldsymbol{h}_x + r, z, z')$ around $\boldsymbol{h}_x$, and get the solution as follows:

$$\hat{r} = -\epsilon \boldsymbol{g} / \|\boldsymbol{g}\|_2 \tag{6}$$

where $\boldsymbol{g} = \nabla_{\boldsymbol{h}_x} L\left(\boldsymbol{h}_x, z, z'\right)$.

The hyper-parameter ϵ determines the degree of semantic change in the latent space. A small value can better maintain the semantic, while a large value can make it easier to generate a sample with grammatical errors. We use a heuristic algorithm to select the most appropriate value. We initialize ϵ with a small value, and gradually increase it until the sentence with target grammatical error is produced or the threshold is reached.

Finally, we use decoder ϕ_D to decode from $\boldsymbol{h}_x + \hat{r}$ and generate the corresponding erroneous sentence $\boldsymbol{x'}$:

$$\boldsymbol{x'} = \phi_D(\boldsymbol{h}_x + \hat{r}) \tag{7}$$

In order to filter sentence pairs with low similarity, we use a model proposed by Parikh et al. (2016) as the similarity discriminator. Given an synthetic sample $\boldsymbol{x'}$ with its original sentence $\boldsymbol{x}$, we use similarity discriminator ψ to get a score $p \in [0, 1]$ which reflects the degree of semantic similarity between $\boldsymbol{x}$ and $\boldsymbol{x'}$. We set a threshold t that if p is greater than this threshold, $\boldsymbol{x'}$ can be selected as the augmented sample.

Our method can generate more natural sentences compared to methods that directly apply noise to tokens. Since we edit latent representations of sentences, we can get more diverse samples with different errors which can not be obtained by only applying noise to tokens.

3.2 Pre-defined Rules

Based on previous works(Choe et al., 2019; Lichtarge et al., 2018), the rule-based method can generate local grammatical errors with high quality. We propose five rules to assist in generating synthetic training data.

Delete. Randomly delete a token with a probability of 0.15.

Add. Firstly, randomly select a word from a word list (Google-10000-English[1]), and then add the selected word to random position with a probability of 0.15.

Replace. Randomly replace a token with its possible forms with a probability of 0.5. If the picked token is a word, we use Word forms[2] to generate all possible forms (adverb, adjective, noun and verb). If not, we select replacements from a punctuation set.

Shuffle. Shuffle the tokens by adding a normal distribution bias to the positions of the words with a probability of 0.1. Particularly, let $\boldsymbol{x} = (p_1, ..., p_L)$ represent the positions of words, where L is the length of sentence and p_i is the position of the i-th word. At the beginning, $p_i = i$. Then, add the normal distribution bias,

$$p'_i = p_i + e_i \tag{8}$$

where e_i subjects to normal distribution $e_i \sim N(0, \sigma^2)$. Finally, re-sort the words by the rectified positions p'_i and get the new sequence $\boldsymbol{x}'$.

Spell Error. Randomly apply spell error to a word with a probability of 0.1. We randomly perturb characters using the same operations as above for the word level operations, i.e. substitution, deletion, insertion or transposition of characters.

Using above data augmentation methods, we can get synthetic training samples with various grammatical errors. These synthetic training samples can further improve the performance and robustness of the GEC system.

4 GEC Model

In this study, we choose copy-augmented Transformer (Zhao et al., 2019) as GEC model to test our data augmentation method. Copy-augmented Transformer is a kind of Transformer that incorporates an attention-based copy mechanism in the decoder. It can generate word from a fixed vocabulary and the source input tokens. Considering the similarity between input and output, this copy mechanism leads to great performance of models in GEC task.

5 Experiment Setup

5.1 Datasets

Training data. We use the following GEC datasets as original training corpus: National University of Singapore Corpus of Learner English (NUCLE)(Dahlmeier et al., 2013), Lang-8 Corpus of Learner English (Lang-8)(Tajiri et al., 2012), FCE dataset(Yannakoudakis et al., 2011), and Write & Improve + LOCNESS Corpus(W&I+LOCNESS)(Bryant et al., 2019).

NUCLE is a collection of essays written by students who are non-native English speakers. Professional English instructors were invited to correct the grammatical errors in these essays. There are 28 common grammatical error types.

The Lang-8 corpus is a cleaned English subset of the language learning websites.

FCE and W&I+LOCNESS are public GEC datasets. Bryant et al.(2019) use an automatic annotation tool ERRANT to annotate the types of grammatical errors. There are 25 main grammatical error types.

Evaluation data. We report results on CoNLL-2014 benchmark evaluated by official M2 scorer(Dahlmeier and Ng, 2012), and on BEA-2019 and FCE benchmarks evaluated by ERRANT.

Seed Corpus Following the Kiyono et al.(2019), we choose the large English corpus Gigaword as seed corpus for data augmentation.

The datasets used in the experiments are summarized in Table 1.

5.2 Pre-processing

We first tokenize the data by NLTK (Bird et al., 2009). Then we apply byte-pair encoding (BPE) (Sennrich et al., 2016b) to sentences using subword-nmt(Sennrich et al., 2016b) before feeding the texts into

[1]https://github.com/first20hours/google-10000-english

[2]https://github.com/gutfeeling/word_forms

Dataset	#Sentences	Parallel	Annotated
Gigaword	131.8M	no	-
NUCLE	57.2K	yes	yes
Lang-8	1.04M	yes	no
FCE	33.2K	yes	yes
W&I+LOCNESS	34.3K	yes	yes

Table 1: Summary of datasets. 'Parallel' means if the dataset has parallel GEC pairs. 'Annotated' means if the types of grammatical errors are annotated.

models. It allows us to avoid $< unk >$ tokens in most datasets. Following Choe et al. (2019), we use spellcheck Enchant[3] to assist the correction of spelling errors. Besides, we use ERRANT to annotate the types of grammatical errors in Lang-8 dataset. In this way, we can get a large amount of annotated data for the training of the grammatical error type classifier.

5.3 Model Training Details

In this paper, we use the Transformer implementation in the public Fairseq Toolkit (Ott et al., 2019). For the Transformer model, the hidden size of embedding is 512. The encoder and decoder have 6 layers and 8 attention heads. For the inner layer in the feed-forward network, the size of is 4096. The number of feed-forward layers in classifier is 3.

The classifier model is trained using the Adam optimization method (Kingma and Ba, 2015). The learning rate is initially set as 0.001 ,the decay factor is set as 0.99 for every epoch. To avoid overfitting, we adopt dropout mechanism (Srivastava et al., 2014). The dropout rate is 0.1.

During decoding, models are optimized with Nesterovs Accelerated Gradient (Nesterov, 1983). We set the dropout to 0.2, the learning rate with 0.002, the weight decay 0.5, the patience 0, the momentum 0.99, minimum learning rate 10-4, and beam-size 5.

From the seed corpus, we generate 16 million training samples in all. Half of training samples are generated by editing latent representations and the other half are generated by pre-defined rules. The probability of five error types (ADJ/ADV, DET, PREP, NOUN, VERB) is equal.

For the GEC model, we follow the default configuration of the copy-augmented Transformer from Zhao et al. (2019). Following Omelianchuk et al.(2020), we train the GEC model in three stages. Firstly, we pre-train the model on synthetic sentences that generated by data augmentation method. Then, we extract sentence pairs containing grammatical errors from four training datasets (NUCLE, Lang-8, FCE and W&I+LOCNESS) and fine-tune the model on these sentence pairs. Finally, we fine-tune the model on respective entire training dataset corresponding to each test set.

5.4 Post-processing

To further improve the performance, we incorporate the following techniques that are widely used in GEC task:

Features Re-scoring (FR). Following Chollampatt and Ng(2018), we use edit operation (EO) features and language model (LM) features to re-score the final beam candidates. EO features denote three features about token-level edit operation. LM features include the score of a language model which is trained on the web-scale Common Crawl corpus (Chollampatt and Ng, 2017; Junczys-Dowmunt and Grundkiewicz, 2016), and the length of the output sequences.

Right-to-left Re-ranking (R2L). Following Sennrich et al. (2016a; 2017), we use the right-to-left re-ranking method to build the ensemble of independently trained models. We pass n-best candidates generated from four left-to-right models to four right-to-left models, and re-rank the n-best candidates based on their corresponding scores.

[3]https://github.com/pyenchant/pyenchant

6 Results and Analysis

6.1 Compared with Existing Methods

We evaluate the performance of our method on public benchmarks and compare the scores with the current top models which adopt data augmentation methods. Table 2 shows the results. Our method achieves the best F-scores on CoNLL-2014 and FCE benchmarks. It outperforms not only all previous single models but also all ensemble models. On BEA-2019 benchmark, our method achieves very competitive performance as well.

Method	Augmented Data Size	CoNLL-2014			BEA-2019			FCE-test		
		P	R	$F_{0.5}$	P	R	$F_{0.5}$	P	R	$F_{0.5}$
Single Model										
Lichtarge et al.(2018)	170M	65.5	37.1	56.8	-	-	-	-	-	-
Kiyono et al.(2019)	70M	67.9	44.1	61.3	65.5	59.4	64.2	-	-	-
Kaneko et al.(2020)	70M	69.2	45.6	62.6	**67.1**	60.1	**65.6**	59.8	46.9	56.7
Our method without R2L	16M	**69.5**	**47.3**	**63.5**	66.9	**60.6**	65.5	**63.0**	**51.4**	**60.3**
Ensemble Model										
Zhao et al.(2019)	30M	71.6	38.7	61.2	-	-	-	-	-	-
Grundkiewicz et al.(2019)	100M	-	-	64.2	72.3	60.1	69.5	-	-	-
Kiyono et al.(2019)	70M	72.4	46.1	65.0	**74.7**	56.7	**70.2**	-	-	-
Kaneko et al.(2020)	70M	**72.6**	46.4	65.2	72.3	**61.4**	69.8	62.8	48.8	59.4
Our method	16M	72.3	**48.8**	**65.9**	72.6	61.3	70.0	**65.4**	**53.6**	**62.6**

Table 2: Comparison results of GEC methods. The top group shows the results of the single models. The second group shows the results of the ensemble models. Augmented data sizes show the amounts of additional training sentences used in each method. **Bold** indicates the highest score in each column.

6.2 Analysis of Data Augmentation Method

We evaluate the performance of our different data augmentation methods on benchmarks. Results are shown in Table 3. 'None' means no use of data augmentation method and 'Both' means using both representation editing based method and rule-based method.

Method	CoNLL-2014			BEA-test			FCE-test		
	P	R	$F_{0.5}$	P	R	$F_{0.5}$	P	R	$F_{0.5}$
None	65.2	33.1	54.6	61.5	49.0	58.5	57.9	31.1	49.4
Editing Latent Representation	69.3	45.6	62.8	69.6	57.0	66.7	62.4	48.1	58.9
Both	**72.3**	**48.8**	**65.9**	**72.6**	**61.3**	**70.0**	65.4	**53.6**	**62.6**

Table 3: Results of our different data augmentation methods.

As we can see, our data augmentation methods can improve the performance of the GEC model, especially the recall. A large amount of synthetic training samples enable the model to detect and correct more errors.

CoNLL-2014 is a typical benchmark, widely used for evaluating GEC models. Besides, this dataset has been hand-corrected by professional English instructors. In view of these, we use CoNLL-2014 dataset as example to do the following experiments.

We investigate the influence of the amount of synthetic training samples on the performance. We pre-train the ensemble model with different amounts of synthetic samples. Considering the limitation of computing resources, we set amounts of synthetic samples as {1M, 2M, 4M, 8M, 16M}. As mentioned above, we use pre-defined rules and editing latent representation method to generate half of the synthetic samples respectively in this experiment. The results in Figure 2 show that the increase of synthetic samples can improve the performance of GEC model, but the growth rate is on the decline.

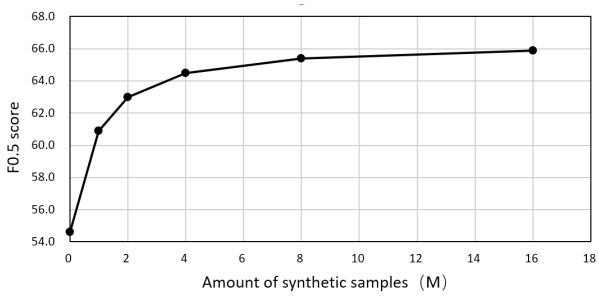

Figure 2: Performance on CoNLL-2014 for different amounts of synthetic samples.

Augmentation Method		**None**	**Pre-defined Rules**	**Editing Latent Representation**	**Both**
Error Type	**%**	**Recall**	**Recall**	**Recall**	**Recall**
Article Or Determiner	14.31	47.52	**55.90**	54.51	55.21
Wrong Collocation/Idiom	12.75	9.41	14.11	15.69	**17.10**
Spelling, Punctuation, etc.	12.47	48.11	**64.15**	52.13	60.14
Preposition	10.38	48.17	67.43	65.51	**68.40**
Noun number	9.38	53.30	74.63	72.49	**75.69**
Verb Tense	5.41	22.18	42.51	**51.71**	50.83
Subject-Verb Agreement	4.93	30.43	40.57	42.60	**53.30**
Verb form	4.69	31.98	52.24	**59.70**	57.57
Redundancy	4.65	18.92	23.01	17.20	**23.66**
Others	20.99	19.53	23.82	34.30	**38.11**
All	100.00	33.18	44.62	45.60	**48.83**

Table 4: Recall of our augmentation methods on different error types. Evaluation dataset is CoNLL-2014. **Bold** indicates the highest recall in each type.

We further analyze the results on different error types. Note that due to the definition of precision on CoNLL-2014 dataset, we cannot calculate the precision for each error type. So we use recall to evaluate the performance of our different data augmentation methods. In CoNLL-2014 dataset, 28 error types are defined, and we list the recall on the top 9 error types. The other 19 types are summarized in the 'others' type. Results are shown in Table 4.

As can be seen from the table, the correction abilities of model on different error types are quite different. For example, model without synthetic data corrects 53.3% errors on the 'Noun number' type, but only corrects 9.41% errors on the 'Wrong Collocation/Idiom' type. Data augmentation methods can address this problem in some degree. The recall of local errors, such as 'Spelling, Punctuation, etc' and 'Noun number', is improved by rule-based method. Representation editing based method improves the performance of model on other error types, such as 'Verb tense' and 'verb form'. With the assistance of pre-defined rules, our method achieves the highest recall on most error types. Our proposed method not only performs well in overall evaluation score, but also greatly improves the performance on most error types. The two data augmentation methods can complement each other and the use of both of them can cover most error types and generate samples with high quality and diversity. It enables the model to detect and correct various errors, which meets the needs in practical application.

6.3 Case Study

In this section, we use specific cases to analyze influence of different data augmentation methods.

In Table 5, we present an example of the synthetic samples generated by our different data augmentation methods. In this case, rule-based method generates an ungrammatical sentence with verb tense error by replacing the verb 'crashed' with its present tense. The editing latent representation based method

Grammatical Sentence	An Israeli military helicopter crashed near the northern town of Afula, army radio said
Augmentation Method	**Example**
Pre-defined Rules	An Israeli military helicopter **crashes** near the northern town of Afula, army radio said.
Editing Latent Representation	An Israeli military helicopter **has been crashed** near the northern town of Afula, army radio said.

Table 5: Example of the synthetic samples. **Bold** indicates the grammatical errors generated by augmentation methods.

also generates a sample with verb tense error. However, this error cannot be generated by using simple pre-defined rules. What's more, the error is more complicated than a simple verb tense error. In fact, it is a grammatical error related to reported speech. Therefore, editing latent representation based method can generate errors which could not be generated by rules. With the help of editing latent representation, we can generate grammatical errors with high quality and diversity.

In Table 6, we present an example of corrections generated by GEC model with different augmentation methods. The performance of GEC model without data augmentation is poor. It cannot detect the error of 'would'. The model with rule-based augmentation method can detect the error, but it fails to correct it by only changing the form of the word. The model with representation editing based method successfully corrects the error. However, it is hard for our proposed method to correct sentences with multiple errors. This problem needs to be solved in future works.

Standard Correction	Although the problem [would → may] not be serious, people [would → might] still be afraid.
Augmentation Method	**Example**
None	Although the [problem → problems] would not be serious, people would still be afraid.
Pre-defined Rules	Although the problem [would → will] not be serious, people [would → will] still be afraid.
Editing Latent Representation	Although the problem [would → may] not be serious, people would still be afraid.
Both	Although the problem [would → may] not be serious, people would still be afraid.

Table 6: Example of corrections. Brackets mark the spans of errors. The text on the right of arrow is the correction of the error on the left.

7 Conclusion

In this paper, we propose a data augmentation method to apply noise to latent space. By editing latent representations of grammatical sentences, we can generate synthetic samples with diverse error types. These synthetic training samples can further improve the performance and robustness of the GEC model and it enables the model to detect and correct most errors. We evaluate our method on public benchmarks of GEC task and it achieves the state-of-the-art performances on CoNLL-2014 and FCE datasets.

Acknowledgments

This work was supported by National Natural Science Foundation of China (61772036), Beijing Academy of Artificial Intelligence (BAAI) and Key Laboratory of Science, Technology and Standard in Press Industry (Key Laboratory of Intelligent Press Media Technology). We appreciate the anonymous reviewers for their helpful comments. Xiaojun Wan is the corresponding author.

References

Steven Bird, Ewan Klein, and Edward Loper. 2009. Natural language processing with python.

Francis Bond, Kentaro Ogura, and Tsukasa Kawaoka. 1996. Noun phrase reference in japanese-to-english machine translation. *ArXiv*, cmp-lg/9601008.

Christopher Bryant, Mariano Felice, and Ted Briscoe. 2017. Automatic annotation and evaluation of error types for grammatical error correction. In *ACL*.

Christopher Bryant, Mariano Felice, Øistein E. Andersen, and Ted Briscoe. 2019. The bea-2019 shared task on grammatical error correction. In *BEA@ACL*.

Yo Joong Choe, Jiyeon Ham, Kyubyong Park, and Yeoil Yoon. 2019. A neural grammatical error correction system built on better pre-training and sequential transfer learning. In *BEA@ACL*.

Shamil Chollampatt and Hwee Tou Ng. 2017. Connecting the dots: Towards human-level grammatical error correction. In *BEA@EMNLP*.

Shamil Chollampatt and Hwee Tou Ng. 2018. A multilayer convolutional encoder-decoder neural network for grammatical error correction. *ArXiv*, abs/1801.08831.

Daniel Dahlmeier and Hwee Tou Ng. 2012. Better evaluation for grammatical error correction. In *HLT-NAACL*.

Daniel Dahlmeier, Hwee Tou Ng, and Siew Mei Wu. 2013. Building a large annotated corpus of learner english: The nus corpus of learner english. In *BEA@NAACL-HLT*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. *ArXiv*, abs/1810.04805.

Tao Ge, Furu Wei, and Ming Zhou. 2018. Reaching human-level performance in automatic grammatical error correction: An empirical study. *ArXiv*, abs/1807.01270.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. *CoRR*, abs/1412.6572.

Alex Graves, Abdel Rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks.

Roman Grundkiewicz, Marcin Junczys-Dowmunt, and Kenneth Heafield. 2019. Neural grammatical error correction systems with unsupervised pre-training on synthetic data. In *BEA@ACL*.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O. K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. *ArXiv*, abs/1603.06393.

Na-Rae Han, Martin Chodorow, and Claudia Leacock. 2006. Detecting errors in english article usage by non-native speakers. *Nat. Lang. Eng.*, 12:115–129.

Emi Izumi, Kiyotaka Uchimoto, Toyomi Saiga, Thepchai Supnithi, and Hitoshi Isahara. 2003. Automatic error detection in the japanese learners' english spoken data. In *ACL*.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. *ArXiv*, abs/1606.03622.

Marcin Junczys-Dowmunt and Roman Grundkiewicz. 2016. Phrase-based machine translation is state-of-the-art for automatic grammatical error correction. *ArXiv*, abs/1605.06353.

Marcin Junczys-Dowmunt, Roman Grundkiewicz, Shubha Guha, and Kenneth Heafield. 2018. Approaching neural grammatical error correction as a low-resource machine translation task. In *NAACL-HLT*.

Masahiro Kaneko, Masato Mita, Shun Kiyono, Jun Suzuki, and Kentaro Inui. 2020. Encoder-decoder models can benefit from pre-trained masked language models in grammatical error correction. *ArXiv*, abs/2005.00987.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *Eprint Arxiv*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Shun Kiyono, Jun Suzuki, Masato Mita, Tomoya Mizumoto, and Kentaro Inui. 2019. An empirical study of incorporating pseudo data into grammatical error correction. In *EMNLP/IJCNLP*.

Kevin Knight and Ishwar Chander. 1994. Automated postediting of documents. *ArXiv*, abs/cmp-lg/9407028.

Jared Lichtarge, Christopher Alberti, Shankar Kumar, Noam Shazeer, and Niki Parmar. 2018. Weakly supervised grammatical error correction using iterative decoding. *ArXiv*, abs/1811.01710.

Guido Minnen, Francis Bond, and Ann A. Copestake. 2000. Memory-based learning for article generation. In *CoNLL/LLL*.

Masaki Murata and Makoto Nagao. 1994. Determination of referential property and number of nouns in japanese sentences for machine translation into english. *ArXiv*, abs/cmp-lg/9405019.

Yurii Nesterov. 1983. A method for solving the convex programming problem with convergence rate o($1/k^2$).

Hwee Tou Ng, Siew Mei Wu, Ted Briscoe, Christian Hadiwinoto, and Christopher Bryant. 2014. The conll-2014 shared task on grammatical error correction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*.

Kostiantyn Omelianchuk, Vitaliy Atrasevych, Artem N. Chernodub, and Oleksandr Skurzhanskyi. 2020. Gector - grammatical error correction: Tag, not rewrite. In *BEA@ACL*.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*.

Ankur P. Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. *ArXiv*, abs/1606.01933.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Improving neural machine translation models with monolingual data. *Computer Science*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Edinburgh neural machine translation systems for wmt 16. In *WMT*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Neural machine translation of rare words with subword units. *ArXiv*, abs/1508.07909.

Rico Sennrich, Alexandra Birch, Anna Currey, Ulrich Germann, Barry Haddow, Kenneth Heafield, Antonio Valerio Miceli Barone, and Philip Williams. 2017. The university of edinburgh's neural mt systems for wmt17. *ArXiv*, abs/1708.00726.

Melanie Siegel. 1996. Preferences and defaults for definiteness and number in japanese to german machine translation. In *Proceedings of the 11th Pacific Asia Conference on Language, Information and Computation*, pages 43–52.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15:1929–1958.

Toshikazu Tajiri, Mamoru Komachi, and Yuji Matsumoto. 2012. Tense and aspect error correction for esl learners using global context. In *ACL*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *ArXiv*, abs/1706.03762.

Ziang Xie, Guillaume Genthial, Stanley Xie, Andrew Y. Ng, and Dan Jurafsky. 2018. Noising and denoising natural language: Diverse backtranslation for grammar correction. In *NAACL-HLT*.

Helen Yannakoudakis, Ted Briscoe, and Ben Medlock. 2011. A new dataset and method for automatically grading esol texts. In *ACL*.

Zheng Yuan and Ted Briscoe. 2016. Grammatical error correction using neural machine translation. In *HLT-NAACL*.

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 *NAACL-HLT*.