Position Offset Label Prediction for Grammatical Error Correction

Xiuyu Wu[1,2] **Jingsong Yu**[1,2] **Xu Sun**[1,3] **Yunfang Wu**[1,3]*
[1]MOE, Key Laboratory of Computational Linguistics, Peking University
[2]School of Software and Microelectronics, Peking University
[3]School of Computer Science, Peking University
{xiuyu_wu,yujingsong,xusun,wuyf}@pku.edu.cn

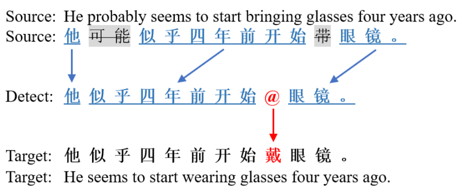
Abstract

We introduce a novel position offset label prediction subtask to the encoder-decoder architecture for grammatical error correction (GEC) task. To keep the meaning of the input sentence unchanged, only a few words should be inserted or deleted during correction, and most of tokens in the erroneous sentence appear in the paired correct sentence with limited position movement. Inspired by this observation, we design an auxiliary task to predict position offset label (**POL**) of tokens, which is naturally capable of integrating different correction editing operations into a unified framework. Based on the predicted POL, we further propose a new copy mechanism (**P-c**opy) to replace the vanilla copy module. Experimental results on Chinese, English and Japanese datasets demonstrate that our proposed **POL-Pc** framework obviously improves the performance of baseline models. Moreover, our model yields consistent performance gain over various data augmentation methods. Especially, after incorporating synthetic data, our model achieves a 38.95 $F_{0.5}$ score on Chinese GEC dataset, which outperforms the previous state-of-the-art by a wide margin of 1.98 points.

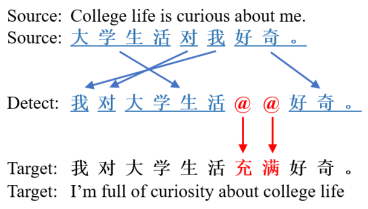
1 Introduction

Grammatical error correction (GEC) is to automatically correct grammatical errors in the input sequence, which is significant for both academic research and practical applications. Comparing with English grammatical error correction, Chinese grammatical error correction (CGEC) is less addressed.

After convolutional neural network and recurrent neural network, the Transformer based sequence-to-sequence models have achieved remarkable performance on GEC task (Junczys-Dowmunt et al., 2018; Kaneko et al., 2020). To alleviate the data sparsity problem, the method of generating


Source: He probably seems to start bringing glasses four years ago.
Source: 他 ~~可 能~~ 似 乎 四 年 前 开 始 带 眼 镜 。
Detect: 他 似 乎 四 年 前 开 始 @ 眼 镜 。
Target: 他 似 乎 四 年 前 开 始 戴 眼 镜 。
Target: He seems to start wearing glasses four years ago.

(a) Example I: deletion and substitution


Source: College life is curious about me.
Source: 大 学 生 活 对 我 好 奇 。
Detect: 我 对 大 学 生 活 @ @ 好 奇 。
Target: 我 对 大 学 生 活 充 满 好 奇 。
Target: I'm full of curiosity about college life

(b) Example II: re-ordering and insertion

Figure 1: Examples of error detection and correction for *CGEC* task. Label @ represents a blank position.

pseudo data is intensively studied (Xie et al., 2018; Lichtarge et al., 2019). Several models simplify the GEC task from sequence generation to token-level edit prediction. For example, LaserTagger (Malmi et al., 2019) predicts three edit tags *Keep*, *Delete* and *Append_#* for each token in the source sentence, and GECToR (Omelianchuk et al., 2020) designs ample English-specific tags to predict.

To keep the meaning of the erroneous sentence unchanged, only a few words could be inserted or deleted while most words remain the same. Taking the CGEC task NLPCC data (Zhao et al., 2018) as an example: (1) about 95.6% tokens in erroneous sentences appear in their corresponding correct sentences (namely "cue tokens" for brevity). (2) the average lengths of erroneous sentences and the longest common sub-sequence of error-corrected sentence pairs are 29.65/27.55 tokens, indicating that cue tokens are almost in the same order. (3) the average lengths of inserted and deleted spans are 1.46 and 1.49 tokens, respectively.

Unfortunately, there is no existing model which takes all of these three characteristics into consid-

*Corresponding author.

eration. The copy mechanism (Zhao et al., 2019) only considers the characteristic (1). The sequence tagging models (Liang et al., 2020; Malmi et al., 2019) take advantage of Characteristic (3) to build a fixed-size vocabulary for *Append_#* label but fail to explicitly model the first and the second points.

In our work, we merge these three characteristics into one key point: most of tokens appear in both erroneous and correct sentences, and their positions in erroneous sentences (denoted as i^{th}) are identical or close to their positions in the corresponding correct sentences (denoted as j^{th}). As shown in Figure 2, this phenomenon could be observed in GEC datasets of different languages. For instance, on NLPCC dataset, 48.2% of cue tokens are exactly in the same position ($i = j$), and about 90% of them can be moved to the right position within a length of 3 tokens ($abs(i - j) \leq 3$).

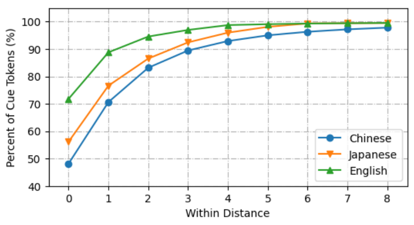


Figure 2: Position movement of cue tokens from erroneous sentences to paired correct sentences in Chinese, Japanese and English GEC datasets, which are described in Section 5.1.

Inspired by the key point summarized above, we propose a novel Position Offset Label (**POL**) prediction task to determine which tokens in the source sentence are error-free, as well as their position movement from the source sequence to the target sentence. As illustrated in Figure 1, our method could simultaneously model different kinds of correction editing operations, including insertion, substitution, deletion and re-ordering. At the decoder, rather than using the conventional attention distribution, a new copy mechanism (**P-copy**) is proposed to take advantage of the result of POL prediction. Overall, we adopt a multi-task learning framework named **POL-Pc**, where the detection network is to perform POL prediction, and the correction network then generates words to fill in the blank positions where no tokens in the source sequence are mapped to.

We conduct extensive experiments on CGEC data NLPCC (Zhao et al., 2018). Experimental results show that without data augmentation, our pure model obtains the best result among all non-pretrained models. After integrating data augmentation methods, our method achieves a new state-of-the-result for CGEC for single models, outperforming the previous best result by 1.98 points. We also conduct experiments on English dataset CoNLL-2014 (Bryant et al., 2019) and Japanese TEC-JL dataset (Koyama et al., 2020), our POL-Pc model consistently improves the performance of the Transformer, validating the generalization ability of our approach. We will make our code publicly available at the GitHub for further research.

To sum, our proposed model for GEC task enjoys the following advantages:

- Comparing with traditional end-to-end methods, our model explicitly separates error detection and error correction, bringing a good interpretability of the neural network.
- Comparing with previous sequence tagging approaches, our model is free to insert any words or phrases without the need to build a specific vocabulary. Moreover, our label strategy is less affected by the imbalance of tag number.
- Our model adopts multi-task learning instead of a pipeline structure to avoid error accumulation.
- Our model outperforms the baseline Transformer by a wide margin for Chinese, Japanese and English GEC tasks. After incorporating data augmentation methods, it obtains a new state-of-the-art result on CGEC task for single models.

2 Related Work

Recently, Transformer-based sequence to sequence models are introduced to GEC task and make great progress (Junczys-Dowmunt et al., 2018; Kaneko et al., 2020). To augment the error-corrected parallel corpus, synthetic data is generated from the Wikipedia revision log (Lichtarge et al., 2019) or generated by applying token-level insertion, substitution, deletion or swapping on error-free corpus (Zhao et al., 2019; Grundkiewicz et al., 2019).

Several models are proposed to simplify GEC task from sequence generation to sequence tagging. LaserTagger (Malmi et al., 2019) predicts *Keep*, *Delete* or *Append_#* tags for each token in the source sentence. PIE (Awasthi et al., 2019) and GECToR (Omelianchuk et al., 2020) manually design detailed English-specific labels, regarding case and tense, which is hard to adapt to other languages like Chinese. TtT (Li and Shi, 2021) adopts a non-autoregressive model to directly predict each token in the correct sentence. ESD-ESC (Chen et al., 2020) detects erroneous spans and then utilizes a seq2seq model to only produce correct text for those annotated spans. Different from these previous works, we build a multi-task learning framework by treating the sequence tagging as an auxiliary subtask, and more importantly, our tag strategy is novel and effective.

For CGEC task, most of previous works (Wang et al., 2018; Zhang et al., 2020) focus on the spelling error correction task on SIGHAN (Tseng et al., 2015) and Hybrid (Wang et al., 2018) datasets. The NLPCC-2018 dataset (Zhao et al., 2018) provides multiple types of grammatical errors and attracts much attention from participated teams, where the top-3 systems are Alibaba (Zhou et al., 2018), YouDao (Fu et al., 2018) and BLCU (Ren et al., 2018). HRG combines language model base spelling checker, NMT-base model and sequence editing model (Hinson et al., 2020). Later, Zhao and Wang proposes MaskGEC by adding random noises to source sentences dynamically, which achieves the state-of-the-art on NLPCC-2018 dataset. This paper also conducts experiments on NLPCC dataset and considers these methods as comparing baselines.

3 Proposed Model

As shown in Figure 1, given an erroneous sentence $X = (x_1, x_2, ...x_m)$ and its corresponding corrected sentence $Y = (y_1, y_2, ...y_n)$, our model firstly decides whether a source token x_i should be deleted and if not predicts its position changing from X to Y. There might exist some blank positions where no source tokens are mapped to (denoted as @ in Figure 1). A decoder with copy mechanism is utilized to copy source tokens and generate new tokens for blank positions, where the copy operation is guided by the result of position offset label prediction. The overview of our framework is illustrated in Figure 3.

3.1 Position Offset Label Prediction

For each token x_i in the erroneous sentence X , we tag its offset (i.e., the distance between its position in X and Y) as:

$$o_i = \begin{cases} j - i, & x_i = y_j \\ null, & x_i \notin Y \end{cases} \tag{1}$$

where o_i denotes the position offset label. If there exist more than one y_j equaling to x_i , x_i will match the nearest one.

In order to retain the original meaning of the input sentence, the lengths of inserted or deleted spans in correction editing should be as short as possible, and cue tokens' positions in corrected sentences are usually close to their positions in erroneous sentences. Benefiting by this observation, we limit the maximum absolute value of offset o_i by a constant k, to reduce the number of possible categories:

$$o_i = \begin{cases} j - i, & x_i = y_j, abs(i-j) \leq k \\ other, & x_i = y_j, abs(i-j) > k \\ null, & x_i \notin Y \end{cases} \tag{2}$$

According to the statistics in Figure 2, there are over 90% of cue tokens whose offsets are less than or equal to 3 for GEC dataset of any language. We consistently set $k = 3$ for experiments on different languages GEC dataset.

For a clear understanding of the operation of position offset label, Figure 4 presents two examples. We can observe that our designed offset label o_i is able to simultaneously model different edit operations in an elegant way, including insertion, substitution, deletion and re-ordering.

We also carry a statistic comparison of our position offset label with previous sequence tagging methods, as listed in Table 1. The edit label approach predicts *Keep*, *Delete*, *Append_#* or labels representing form transformations of each token (Malmi et al., 2019; Omelianchuk et al., 2020), where one token might have multiple labels (for example, one token might have *Append_#* and tense transformation label simultaneously). The right/wrong label (Zhao et al., 2019; Chen et al., 2020) predicts whether each token in the erroneous sentence should be kept in the correct sentence, which suffers from data imbalance. Compared with them, our strategy avoids all these problems.

We adopt the Transformer encoder to obtain the sequence representation vectors, and then a linear layer and softmax operation are utilized to generate a probability distribution:

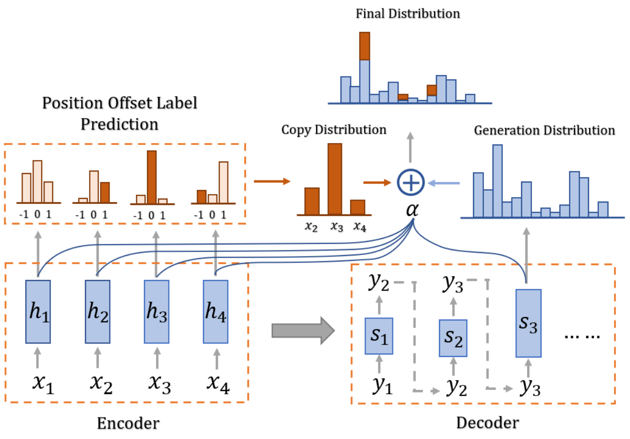
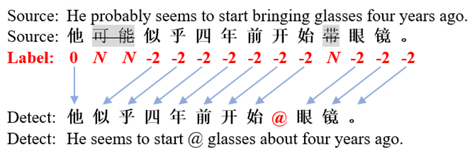
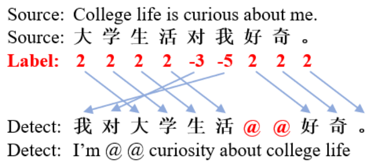


Figure 3: Overview of our POL-Pc framework. For a clear display, we limit the position offset label to $\{-1, 0, 1\}$, which respectively means the source token should be moved to the previous position, be kept in the same position or be moved to the next position.



Source: He probably seems to start bringing glasses four years ago.
Source: 他 ~~可 能~~ 似 乎 四 年 前 开 始 ~~带~~ 眼 镜 。
Label: 0 *N* *N* -2 -2 -2 -2 -2 -2 -2 *N* -2 -2 -2
Detect: 他 似 乎 四 年 前 开 始 @ 眼 镜 。
Detect: He seems to start @ glasses about four years ago.

(a) Example I: Deletion and substitution



Source: College life is curious about me.
Source: 大 学 生 活 对 我 好 奇 。
Label: 2 2 2 2 -3 -5 2 2 2
Detect: 我 对 大 学 生 活 @ @ 好 奇 。
Detect: I'm @ @ curiosity about college life

(b) Example II: Insertion and reorder

Figure 4: Examples of position offset labels for different types of operations. *N* represents *null*. @ represents a blank position.

$$(h_1^L, h_2^L, ..., h_m^L) = Transformer(x_1, x_2, ..., x_m) \tag{3}$$

$$P_o(o_t|X) = softmax(W_o^T h_t^L + b_o) \tag{4}$$

where W_o and b_o are learned parameters. The loss function for label prediction is defined using log-likelihood:

$$Loss_{off} = -\sum_{t=1}^{m} log P_o(o_t|X) \tag{5}$$

3.2 Copying with Position Offset Label Prediction

We employ a decoder with copy mechanism to generate error-free sentences. Different from conventional copy methods, we propose a new copy mechanism (P-copy) based on the prediction of position offset label.

Tag	# Class	Single label	Maj. (%)
Edit	$\geq$ 500		85.6
Right/Wrong	2	✓	90.2
Position Offset	9	✓	43.5

Table 1: Comparison of different tagging strategies. *Single label* refers to whether each token in the erroneous sentence has only one label. *Maj.* refers to the percentage of the major label that takes up the highest proportion.

At timestep t, the generation distribution is computed as:

$$P_t^{gen} = softmax(W_s^T s_t + b_s) \tag{6}$$

where s_t is the hidden state of decoder, W_s and b_s are parameters to learn.

The copy distribution of each token at timestep t is computed by:

$$P_t^{copy}(y_t) = \begin{cases} \sum\limits_{i:x_i=y_t} P_o(o_i = t - i), & y_t \in X \\ 0, & otherwise \end{cases} \tag{7}$$

It means if the source token x_i appears at the t^{th} position in the sequence Y with a probability of $P_o(o_i = t - i)$, the model will copy the token x_i at timestep t with the same probability. We then utilize the softmax function to normalize copy distribution P_t^{copy}. The copy rate α_t is computed using attention mechanism and sigmoid function:

$$Q_t = W_q s_t, K_i = W_k h_i^L, V_i = W_v h_i^L \tag{8}$$

$$a_i = \frac{exp(Q_t K_i^T)}{\sum_j exp(Q_t K_j^T)} \quad (9)$$

$$h_c = \sum_{i=1}^{m} a_i V_i \quad (10)$$

$$\alpha_t = sigmoid(W_c^T h_c + b_c) \quad (11)$$

where W_q, W_k, W_v, W_c and b_c are parameters to learn.

We fuse the generation probability and copy probability to get the final distribution:

$$P_v = (1 - \alpha_t)P_t^{gen} + \alpha_t P_t^{copy} \quad (12)$$

Moreover, the loss function for generation is:

$$Loss_{gen} = -\sum_{t=1}^{n} logP_v(y_t|y_{t-1}, y_{t-2}, ...y_1, X) \quad (13)$$

Combing the generation loss with the loss of POL prediction, the final optimization object of our model is:

$$Loss = Loss_{gen} + \gamma Loss_{off} \quad (14)$$

where γ is a hyper-parameter to balance two loss functions.

4 Data Augmentation

In order to provide more training samples for our model, we do data augmentation by generating synthetic data and introducing dynamic noises. We introduce the following types of noises to produce erroneous sentences.

Deletion: To discard each word with a probability of P_{del}.

Insertion: To insert a [UNK] symbol before each word with a probability of P_{ins}.

Substitution: To replace each word using [UNK] symbol / random token / its homophone with a probability of P_{sub}.

We set $P_{ins} = P_{del} = P_{sub} = 0.25$ in all experiments.

4.1 Synthetic Data

We select THUCnews[1] as the external data which contains 740,000 news documents. We delete the format information such as headlines and bylines in the news corpus and cut the reserved text into 11.4 million single sentences, from which we randomly select 5 million sentences. In our experiment, we introduce only one type of noises into each sentence. Since mixing up synthetic data with training data might dilute human-made corpus and degrade model performance, we only use synthetic data to initialize the parameters of our model by pre-training 10 epoches.

4.2 Dynamic Noise

Inspired by MaskGEC (Zhao and Wang, 2020), we add dynamic noises to input sentences when training on NLPCC dataset to provide more instances. For each human-made erroneous sentence, we randomly choose to keep it unchanged or introduce one type of noises described above.

5 Experimental Setup

5.1 Dataset

We conduct experiments for Chinese, English and Japanese GEC tasks. The details of datasets of different languages is listed in Table 2.

For CGEC task, we choose the dataset of NLPCC 2018 Task 2 (Zhao et al., 2018) , where the training data is collected from the language learning platform Lang-8[2] while the test data is created by teachers. Following the prior work (Zhao and Wang, 2020), we randomly select 5,000 instances from 1.09 million training samples as the development set, and evaluate our model on the official test set containing 2000 sentences. We use BasicTokenizer from BERT project[3] to tokenize the Chinese texts and keep the non-Chinese words unchanged. During evaluation, We tokenize texts with PKUNLP[4] toolkit, which has been used officially at the campaign.

To verify the generalization of our method, we also conduct experiments on English and Japanese GEC tasks. For English GEC task, following Bryant et al. (2019), we use FCE (Yannakoudakis et al., 2011), Lang-8 Corpus of Learner English (Mizumoto et al., 2011), NUCLE (Dahlmeier et al., 2013) and W&I+LOCNESS (Bryant et al., 2019) as training data, CoNLL-2013 test set as dev set and evaluate on CoNLL-2014 (Ng et al., 2014) test set. For Japanese GEC task, we select 1.19 million/5000 sentence pairs as training/dev set from

[1]http://thuctc.thunlp.org/

[2]https://lang-8.com/

[3]https://github.com/google-research/bert

[4]http://59.108.48.12/lcwm/pkunlp/downloads/libgrass-ui.tar.gz

corpora collected from Lang-8 website[5], and evaluate our model on TEC-JL dataset (Koyama et al., 2020).

We adopt M^2 scorer (Dahlmeier and Ng, 2012) as the evaluation tool, where the value of precision, recall and $F_{0.5}$ score is computed.

Language	Train	Dev	Test	Dict	Vocab
Chinese	1.09 M	5,000	2,000	char	13.8 K
Japanese	1.19 M	5,000	1,874	char	6.5 K
English	1.01 M	1,381	1,312	bpe	32 K

Table 2: Data statistics for Chinese, Japanese and English GEC tasks.

5.2 Comparing Methods

We compare our model with the top-3 participated teams of the campaign.

AliGM combines rule-based, SMT-based and NMT-based approaches (Zhou et al., 2018).

YouDao employs five different hybrid models and the final result is selected via a language model (Fu et al., 2018).

BLCU builds a mutli-layer convolutional model with pre-trained embeddings (Ren et al., 2018).

Furthermore, we compare with some of mainstream neural network models for text generation.

Transformer adopts a standard encoder-decoder framework (Vaswani et al., 2017).

Levenshtein-**Transformer** takes insertion and deletion as atomic operations (Gu et al., 2019).

LaserTagger predicts edit operations *Keep*, *Delete* or *Append_#* for each token (Malmi et al., 2019).

ESD-ESC adopts a pipeline structure to firstly detect erroneous spans and then output the correct text for annotated spans (Chen et al., 2020).

Pointer Generator decides to generate a token or copy a token from the source sentence (See et al., 2017), where Transformer is used as the encoder and decoder.

Copy-Augmented introduces copy mechanism into Transformer-based seq2seq framework and employs multi-task learning by predicting whether tokens in input sentence appear in target sentence (Zhao et al., 2019).

HRG proposes a heterogeneous approach composed of language model base spelling checker, NMT-base model and sequence editing model (Hinson et al., 2020).

BERT-fuse incorporates pre-trained BERT model to enhance Transformer (Kaneko et al., 2020).

MaskGEC (Zhao and Wang, 2020) adds random noises to source sentences dynamically.

5.3 Training Details

Our model is implemented using Fairseq [6]. The detail of hyper-parameters of our model is described in Table 3.

Hyper-parameter	Value
encoder layers	6
decoder layers	6
encoder embedding dim	512
decoder embedding dim	512
encoder ffn dim	2048
decoder ffn dim	2048
dropout	0.2
attention dropout	0.1
learning rate	5e-4
Adam β_1	0.9
Adam β_2	0.998
Adam ϵ	1e-8
lr scheduler	inverse_sqrt
warmup updates	8000
max tokens	4096
update-freq	2
loss function	label smoothed cross entropy label-smoothing=0.1

Table 3: Hyper-parameter values of our model.

We average parameters of the last 5 checkpoints. In inference, the beam size is set as 12. In the final loss of Equation 14, we set $\gamma = 0.5$.

6 Results and Discussion

6.1 Main Results for CGEC task

For Chinese GEC task, the overall results of different models are reported in Table 4.

Our pure model POL-Pc yields a 30.64 $F_{0.5}$ score, which beats all the pure models without data augmentation or other extra resources. The Copy-Augmented model reaches 29.85 $F_{0.5}$ by employing the copy mechanism and predicting whether each token in source sentence is kept in the target sentence. Compared with it, our sub-task of POL prediction suffers less from the data imbalance, leading to an improvement of 0.79 $F_{0.5}$ score.

After applying dynamic noises, our model gets a 37.97 $F_{0.5}$ score which outperforms the previous best model MaskGEC. After pretrained on the synthetic data, our model achieves a new state-of-the-art result of 38.95 $F_{0.5}$.

[5]https://sites.google.com/site/naistlang8corpora

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

Models	Precision	Recall	$F_{0.5}$
LaserTagger*	25.60	10.50	19.90
Lev-Transformer*	24.90	15.00	22.00
Transformer	34.02	15.44	27.42
Pointer Generator	33.51	16.45	27.75
ESD-ESC*	37.30	14.50	28.40
Copy-Augmented	35.03	18.75	29.85
AliGM*▲♢	41.00	13.75	29.36
YouDao*▲♢	35.24	18.64	29.91
BLCU*▲♢	**47.63**	12.56	30.57
BERT-fuse♢	35.16	23.32	31.92
HRG*▲♢	36.79	**27.82**	34.56
MaskGEC*♡	44.36	22.18	36.97
POL-Pc	36.75	18.39	30.64
POL-Pc + DN♡	44.64	23.77	37.97
POL-Pc + DN + SD♡♢	46.45	23.68	**38.95**

Table 4: Overall performance of different models on NLPCC-2018 test dataset. The model performance with * is from the original published paper and the result of LaserTagger is from (Chen et al., 2020). Other models are re-implemented on our data using the released codes or Fairseq. **SD** and **DN** refer to synthetic data and dynamic noise described in Section 4. The symbols ▲ / ♡ / ♢ denote ensemble / data augmentation / pre-training approaches respectively.

6.2 Results for Various Language GEC tasks

In addition to Chinese, Table 5 reports the experimental results on English and Japanese datasets. Compared with the baseline Transformer, our pure model brings a relative performance gain of 11.7%, 7.4% and 4.0% on Chinese, Japanese and English datasets respectively, validating the effectiveness of our approach for different language GEC tasks.

It also demonstrates that our model is more suitable for Chinese and Japanese GEC task. Both Chinese and Japanese language utilize function words instead of affixes to represent forms and tenses, which leads to more insertion and deletion operation when correcting grammatical errors. In contrast, the substitution operation such as shifting of tenses is the main operation for English GEC task. As demonstrated in Figure 2, tokens in English GEC dataset are more likely kept in the exactly same position. Besides, English word are divided into subwords by BPE, which makes POL prediction module hard to train. As a result, the prediction of token position movement is more beneficial for Chinese and Japanese GEC tasks.

6.3 Results with Different Data Augmentation Methods

Since data augmentation is intensively studied for GEC, we implement our model trained on different synthetic data, and the experimental results are shown in Table 6. Our model consistently outperforms Transformer utilizing different data augmentation approaches. Compared with MaskGEC which benefits both Transformer and our POL-Pc model, our dynamic noise, as described in Section 4, brings more improvement to POL-Pc. It demonstrates that providing more training samples with different position movement generated by deletion and insertion operations could further improve the performance of our model.

Models	NLPCC	TEC-JL	CoNLL-14
Lev-Transformer	22.00	16.21	42.48
Pointer Generator	27.75	26.62	49.99
Transformer	27.42	26.11	48.63
POL-Pc	30.64	28.03	50.56
vs. Transformer	+11.7%	+7.4%	+4.0%

Table 5: Performance on Chinese, Japanese and English GEC datasets.

Data Augmentation	Transformer	POL-Pc	Imp.
None	27.42	30.64	+3.22
MaskGEC	36.97	37.26	+0.29
Dynamic Noise	37.02	37.97	+0.95
Synthetic Data (SD)	32.81	33.37	+0.56
SD + MaskGEC	37.83	38.21	+0.38
SD + Dynamic Noise	37.71	38.95	+1.24

Table 6: Performance of our model after incorporating different data augmentation methods on NLPCC-2018 dataset. *Imp* refers to improvement.

6.4 Results of POL Prediction on Different Model Architectures

To further evaluate the effectiveness of position offset label prediction for GEC task, we apply this module to other model architectures, including the standard Transformer, Pointer-Generator and Copy-augmented models. We evaluate their performance on NLPCC test dataset. As shown in Table 7, adding POL prediction as one of jointly-training tasks consistently improves the performance of different baseline models.

It is noteworthy that combining the POL prediction sub-task with Pointer Generator reaches 29.57 $F_{0.5}$ score, which is lower than our pure model ($F_{0.5} = 30.64$). Because our model adopts the probability of POL prediction as the copy score, which is more effective than making POL prediction and copy mechanism work separately.

Model	Precision	Recall	$F_{0.5}$	Imp.
Transformer	34.02	15.44	27.42	-
+ *POL*	32.59	18.14	28.11	+0.69
Pointer Generator	33.51	16.45	27.75	-
+ *POL*	36.22	17.05	29.57	+1.82
Copy-Augmented	35.03	18.75	29.85	-
+ *POL*	37.89	16.89	30.34	+0.49

Table 7: Experimental results of POL prediction based on different model architectures NLPCC-2018 dataset.

Model	Precision	Recall	$F_{0.5}$	Imp.
POL-Pc	36.75	18.39	30.64	-
- *P-copy*	32.59	18.14	28.11	-2.53
- *(POL + P-copy)*	34.02	15.44	27.42	-3.22
POL-Pc+DN+SD	46.45	23.68	38.95	-
- *P-copy*	45.87	23.23	38.39	-0.56
- *(POL + P-copy)*	45.05	22.84	37.71	-1.24
- *Synthetic Data*	44.64	23.77	37.97	-0.98
- *Dynamic Noise*	43.43	17.32	33.37	-4.58

Table 8: Ablation study on NLPCC dataset. *POL* refers to position offset label prediction, and *P-copy* refers to our POL-based copy mechanism. **DN** refers to dynamic noises and **SD** refers to synthetic data.

6.5 Ablation Study

We do ablation study on NLPCC dataset to evaluate the effect of each module, and list the results in Table 8. On the pure model, removing the POL-based copy mechanism results in a 2.53 decrease, and removing both POL prediction and copy modules leads to a sharp decrease of 3.22 points, which proves that our proposed copy mechanism is a suitable way to take advantage of the results of POL prediction.

Our full model greatly benefits from data augmentation with a 8.31 increase in $F_{0.5}$. For the full model, removing the copy module causes a decrease of 0.56 $F_{0.5}$ score. Removing both POL prediction and copy modules results in a drop of 1.24 $F_{0.5}$ score, which shows that our proposed module can improve model performance even after a huge amount of pseudo data being incorporated. Discarding the synthetic data, the performance decreases from 38.95 to 37.97, suggesting that using synthetic data for pre-training provides the model with better initial parameters.

6.6 Hyper-parameter Setting

In this section, we explore the effect of hyper-parameter γ in Equation 14. As illustrated in Figure 5, $F_{0.5}$ score shows a single-peaked pattern as γ increases. In most cases, our pure model surpasses the basic Transformer model and achieves the best performance when γ is 0.5. When γ decreases, the model fails to explicitly predict the position offset label guided by ground-truth labels, making the model gradually degrade to the vanilla copy mechanism, and when γ increases, the model pays less attention to the generation ability of decoder which also harms the performance.

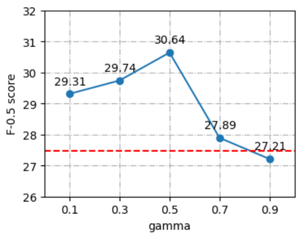


Figure 5: The effect of hyper-parameter γ on multi-task learning based on the standard Transformer model. The red line refers to the performance of basic model on NLPCC dataset.

6.7 Case Study

To investigate how POL prediction and the correction network work together, we select two examples from the NLPCC test data to visualize the intermediate result and final output. As illustrated in Figure 6, in both examples, most of tokens are predicted to be moved to the right position. For tokens which should be deleted or substituted, our model predicts their probability of moving to any position to be low, which means the probability of deletion $P_o(o_i = null)$ is high. Moreover, in the first example, the copy rate of tokens not appearing in the target sentence is low, which shows our copy mechanism works effectively with respect to different offset labels. These two modules in our framework can accomplish their respective tasks and cooperate smoothly with each other.

7 Conclusion

We introduce a detection network to predict POL of tokens between source erroneous sentences and target correct sentences. Based on the output of POL, we design a new copy mechanism P-copy. Our POL-Pc model exceeds both end-to-end models and sequence tagging approaches, achieving a new state-of-the-art result on CGEC task for single models. For English and Japanese GEC tasks, our approach also obtains significant performance gain over the baseline Transformer.

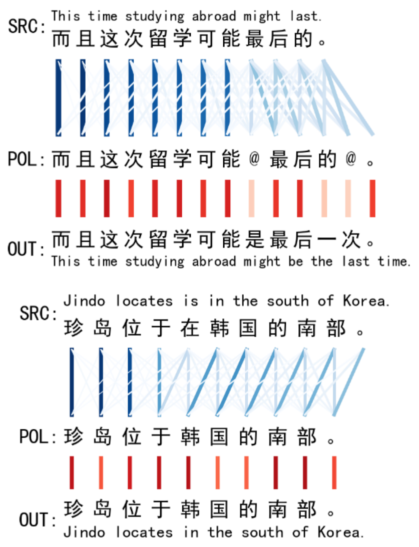

SRC: This time studying abroad might last.
而且这次留学可能最后的。
POL: 而且这次留学可能@最后的@。
OUT: 而且这次留学可能是最后一次。
This time studying abroad might be the last time.
SRC: Jindo locates is in the south of Korea.
珍岛位于在韩国的南部。
POL: 珍岛位于韩国的南部。
OUT: 珍岛位于韩国的南部。
Jindo locates in the south of Korea.

Figure 6: Visualization of POL prediction and copy mechanism. The color blue represents the probability of POL, and the color red represents the copy rate α_t defined in Equation 11. The color becomes darker as the value gets bigger. Label @ represents a blank position.

Acknowledgement

This work is supported by National Key Research and Development Program of China (2020AAA0109703), the National Natural Science Foundation of China (62076008) and the Key Project of Natural Science Foundation of China (61936012).

References

Abhijeet Awasthi, Sunita Sarawagi, Rasna Goyal, Sabyasachi Ghosh, and Vihari Piratla. 2019. Parallel iterative edit models for local sequence transduction. In *EMNLP/IJCNLP (1)*, pages 4259–4269. Association for Computational Linguistics.

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

Mengyun Chen, Tao Ge, Xingxing Zhang, Furu Wei, and Ming Zhou. 2020. Improving the efficiency of grammatical error correction with erroneous span detection and correction. In *EMNLP (1)*, pages 7162–7169. Association for Computational Linguistics.

Daniel Dahlmeier and Hwee Tou Ng. 2012. Better evaluation for grammatical error correction. In *HLT-NAACL*, pages 568–572. The Association for Computational Linguistics.

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*.

Kai Fu, Jin Huang, and Yitao Duan. 2018. Youdao's winning solution to the nlpcc-2018 task 2 challenge: A neural machine translation approach to chinese grammatical error correction. In *NLPCC (1)*, volume 11108 of *Lecture Notes in Computer Science*, pages 341–350. Springer.

Roman Grundkiewicz, Marcin Junczys-Dowmunt, and Kenneth Heafield. 2019. Neural grammatical error correction systems with unsupervised pre-training on synthetic data. In *BEA@ACL*, pages 252–263. Association for Computational Linguistics.

Jiatao Gu, Changhan Wang, and Jake Zhao. 2019. Levenshtein transformer. In *NeurIPS*.

Charles Hinson, Hen-Hsen Huang, and Hsin-Hsi Chen. 2020. Heterogeneous recycle generation for chinese grammatical error correction. In *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*, pages 2191–2201. International Committee on Computational Linguistics.

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*, pages 595–606. Association for Computational Linguistics.

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. In *ACL*, pages 4248–4254. Association for Computational Linguistics.

Aomi Koyama, Tomoshige Kiyuna, Kenji Kobayashi, Mio Arai, and Mamoru Komachi. 2020. Construction of an evaluation corpus for grammatical error correction for learners of japanese as a second language. In *LREC*.

Piji Li and Shuming Shi. 2021. Tail-to-tail non-autoregressive sequence prediction for chinese grammatical error correction. In *ACL/IJCNLP*.

Deng Liang, Chen Zheng, Lei Guo, Xin Cui, Xiuzhang Xiong, Hengqiao Rong, and Jinpeng Dong. 2020. BERT enhanced neural machine translation and sequence tagging model for Chinese grammatical error diagnosis. In *Proceedings of the 6th Workshop on Natural Language Processing Techniques for Educational Applications*, pages 57–66, Suzhou, China. Association for Computational Linguistics.

Jared Lichtarge, Chris Alberti, Shankar Kumar, Noam Shazeer, Niki Parmar, and Simon Tong. 2019. Corpora generation for grammatical error correction. In *NAACL-HLT (1)*, pages 3291–3301. Association for Computational Linguistics.

Eric Malmi, Sebastian Krause, Sascha Rothe, Daniil Mirylenka, and Aliaksei Severyn. 2019. Encode, tag, realize: High-precision text editing. In

EMNLP/IJCNLP (1), pages 5053–5064. Association for Computational Linguistics.

Tomoya Mizumoto, Mamoru Komachi, Masaaki Nagata, and Yuji Matsumoto. 2011. Mining revision log of language learning sns for automated japanese error correction of second language learners. In *IJCNLP*, pages 147–155. The Association for Computer Linguistics.

Hwee Tou Ng, Siew Mei Wu, Ted Briscoe, Christian Hadiwinoto, Raymond Hendy Susanto, and Christopher Bryant. 2014. The conll-2014 shared task on grammatical error correction.

Kostiantyn Omelianchuk, Vitaliy Atrasevych, Artem N. Chernodub, and Oleksandr Skurzhanskyi. 2020. Gector - grammatical error correction: Tag, not rewrite. In *BEA@ACL*, pages 163–170. Association for Computational Linguistics.

Hongkai Ren, Liner Yang, and Endong Xun. 2018. A sequence to sequence learning for chinese grammatical error correction. In *NLPCC (2)*, volume 11109 of *Lecture Notes in Computer Science*, pages 401–410. Springer.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *ACL (1)*, pages 1073–1083. Association for Computational Linguistics.

Yuen-Hsien Tseng, Lung-Hao Lee, Li-Ping Chang, and Hsin-Hsi Chen. 2015. Introduction to sighan 2015 bake-off for chinese spelling check. In *SIGHAN@IJCNLP*, pages 32–37. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30*, page 5998–6008. Curran Associates, Inc.

Dingmin Wang, Yan Song, Jing Li, Jialong Han, and Haisong Zhang. 2018. A hybrid approach to automatic corpus generation for chinese spelling check. In *EMNLP*, pages 2517–2527. Association for Computational Linguistics.

Ziang Xie, Guillaume Genthial, Stanley Xie, Andrew Y. Ng, and Dan Jurafsky. 2018. Noising and denoising natural language: Diverse backtranslation for grammar correction. pages 619–628. Association for Computational Linguistics.

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

Shaohua Zhang, Haoran Huang, Jicong Liu, and Hang Li. 2020. Spelling error correction with soft-masked bert. In *ACL*, pages 882–890. Association for 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 *NAACL-HLT (1)*, pages 156–165. Association for Computational Linguistics.

Yuanyuan Zhao, Nan Jiang, Weiwei Sun, and Xiaojun Wan. 2018. Overview of the nlpcc 2018 shared task: Grammatical error correction. In *CCF International Conference on Natural Language Processing and Chinese Computing*, pages 439–445. Springer.

Zewei Zhao and Houfeng Wang. 2020. Maskgec: Improving neural grammatical error correction via dynamic masking. In *AAAI*, pages 1226–1233. AAAI Press.

Junpei Zhou, Chen Li, Hengyou Liu, Zuyi Bao, Guangwei Xu, and Linlin Li. 2018. Chinese grammatical error correction using statistical and neural models. In *NLPCC (2)*, volume 11109 of *Lecture Notes in Computer Science*, pages 117–128. Springer.