TemplateGEC: Improving Grammatical Error Correction with Detection Template

Yinghao Li[1*] **Xuebo Liu**[2] **Shuo Wang**[3] **Peiyuan Gong**[1] **Derek F. Wong**[4]
Yang Gao[1] **Heyan Huang**[1] **Min Zhang**[2]
[1]School of Computer Science and Technology, Beijing Institute of Technology, Beijing, China
`{yhli,pygong,gyang,hhy63}@bit.edu.cn`
[2]Institute of Computing and Intelligence, Harbin Institute of Technology, Shenzhen, China
`{liuxuebo,zhangmin2021}@hit.edu.cn`
[3]Tsinghua University, Beijing, China [4]University of Macau, Macau, China
`wangshuo.thu@gmail.com, derekfw@um.edu.mo`

Abstract

Grammatical error correction (GEC) can be divided into sequence-to-edit (Seq2Edit) and sequence-to-sequence (Seq2Seq) frameworks, both of which have their pros and cons. To utilize the strengths and make up for the shortcomings of these frameworks, this paper proposes a novel method, TemplateGEC, which capitalizes on the capabilities of both Seq2Edit and Seq2Seq frameworks in error detection and correction respectively. TemplateGEC utilizes the detection labels from a Seq2Edit model, to construct the template as the input. A Seq2Seq model is employed to enforce consistency between the predictions of different templates by utilizing consistency learning. Experimental results on the Chinese NLPCC18, English BEA19 and CoNLL14 benchmarks show the effectiveness and robustness of TemplateGEC. Further analysis reveals the potential of our method in performing human-in-the-loop GEC. Source code and scripts are available at `https://github.com/li-aolong/TemplateGEC`.

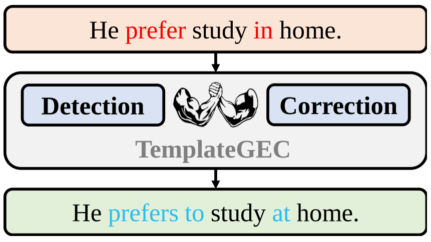

Figure 1: TemplateGEC takes the best of both worlds by utilizing the detection ability of the Seq2Edit framework and the correction ability of the Seq2Seq framework.

1 Introduction

Grammatical error correction (GEC) is a fundamental task in natural language processing that focuses on identifying and correcting grammatical errors in written text (Ng et al., 2013, 2014). The utilization of GEC is wide-ranging, including but not limited to, improving the quality of machine translation (Popović, 2018), increasing the readability of text (Liao et al., 2020), and assisting non-native speakers in language proficiency (Knill et al., 2019). There has been a significant amount of research in the field of GEC (Yuan and Briscoe, 2016; Bryant et al., 2017a; Ren et al., 2018; Zhou et al., 2018; Awasthi et al., 2019; Lai et al., 2022; Gong et al., 2022; Zhang et al., 2022b), which can be broadly classified into two categories: Sequence-to-Edit (Seq2Edit) and Sequence-to-Sequence (Seq2Seq).

Seq2Edit GEC typically involves converting a source sentence into a sequence of editing operations (Stahlberg and Kumar, 2020; Omelianchuk et al., 2020). Despite certain limitations, such as the manual selection of edits and the use of a dictionary (Awasthi et al., 2019; Malmi et al., 2019), Seq2Edit GEC have specific advantages for grammatical error detection due to its high understanding ability (Omelianchuk et al., 2020). Seq2Seq GEC, on the other hand, which approaches GEC as a monolingual translation problem (Ge et al., 2018; Sun et al., 2021), has the advantage of better generation ability of the corrected sentence. However, Seq2Seq GEC still encounters the challenge of over-correction (Park et al., 2020).

In this paper, we propose a novel approach, named TemplateGEC, to merge both frameworks and leverage their respective strengths for grammatical error detection and correction. The proposed approach, as illustrated in Figure 1, utilizes a source template to introduce the detection label from Seq2Edit GEC to Seq2Seq GEC. This enables the Seq2Seq GEC model to make more accurate predictions with the assistance of the detection label. However, the predicted labels from Seq2Edit models may not always be accurate and may contain errors. To enhance the robustness of the model

[*] Work done when Yinghao Li was a remote intern at Harbin Institute of Technology, Shenzhen.

to these inaccurately predicted labels, we propose incorporating gold labels through consistency learning. Experimental results on widely-used English and Chinese GEC benchmarks show the effectiveness and robustness of TemplateGEC. Additional analysis reveals its high potential for human-in-the-loop GEC through the proposed detection template. Our **main contributions** are as follows:

- We explore the integration of Seq2Edit and Seq2Seq GEC, by leveraging their respective strengths of understanding and generation.
- We propose a detection template to introduce detection information from Seq2Edit GEC to Seq2Seq GEC, which allows the model to make more accurate predictions.
- We introduce a gold label-assisted consistency learning method to enhance the robustness of the model to inaccurately predicted labels.
- Our proposed method shows high upper bounds of utilizing gold labels, which has the potential to inspire new research in the area of human-in-the-loop GEC.

2 Related Work

2.1 Sequence-to-Edit GEC

Seq2Edit GEC generally predicts the operation for each token in a sentence, such as insertion, deletion, etc. LaserTagger (Malmi et al., 2019) transforms a source text into a sequence of token-level edit operations, which consist of keeping, deleting, adding and swapping. PIE (Awasthi et al., 2019) reduces the local sequence editing problem to a sequence labeling setup and utilizes BERT to non-autoregressively label input tokens with edits. Stahlberg and Kumar (2020) propose a sequence editing model named Seq2Edits, in which the prediction target is a sequence of edit operations applied to the source. GECToR (Omelianchuk et al., 2020) introduces custom g-transformations in addition to the conventional edit operations, such as capitalization change, merging of two tokens, changing word suffixes and so on. A limitation of Seq2Edit is that it heavily relies on the manual construction of editing operations. This dependence on manual curation renders the model less transferable and results in a lower degree of fluency in the output (Li et al., 2022b). Conversely, its strength is demonstrated in its ability to effectively perform error detection (Yuan et al., 2021), which is facilitated by the accurate prediction of each input category, as opposed to a focus on text fluency.

2.2 Sequence-to-Sequence GEC

Seq2Seq GEC encodes the erroneous sentence through the encoder and uses the decoder to generate each error-free token, which has been well explored (Liu et al., 2021; Wang et al., 2021; Li et al., 2022a; Fang et al., 2023a). The seq2Seq model is more suitable to generate fluent sentences while the decoding speed of it is slower. Zhao et al. (2019) employ a copy-augmented framework and copy unchanged tokens from the sentence pair to the target sentence. Kaneko et al. (2020) explore how to effectively incorporate pre-trained knowledge into the encoder-decoder framework. Qorib et al. (2022) propose a simple logistic regression method to combine GEC models much more effectively. It is noted that constructing pseudo datasets is most useful on GEC task, as noise can be easily injected into error-free sentences automatically, and receive large pseudo sentence pairs which can be used to pre-train GEC models (Zhao et al., 2019; Zhou et al., 2020; Lichtarge et al., 2019; Kiyono et al., 2020; Yasunaga et al., 2021; Sun et al., 2022; Fang et al., 2023b).

Previous works have preliminary attempted to incorporate detection label knowledge into GEC models in order to improve correction results. Chen et al. (2020) use error spans and source sentences as input and output correct spans. Yuan et al. (2021) take detection labels as auxiliary input and using for re-ranking. In our work, we propose a simple and effective way to exploit detection information, providing a nice alternative for this line of research.

3 TemplateGEC

This section introduces the proposed method as illustrated in Figure 2. TemplateGEC integrates detection information generated by a Seq2Edit model and fuses the information into a Seq2Seq model for model enhancement.

3.1 Definition of Error Detection Label

To incorporate the detection information, we first acquire the error label for a given input sentence. This label is then utilized to identify the specific words or phrases in the sentence that contain grammatical errors. Given the source input sentence $\boldsymbol{x} = x_1, x_2, ..., x_N$, the error detection label of the

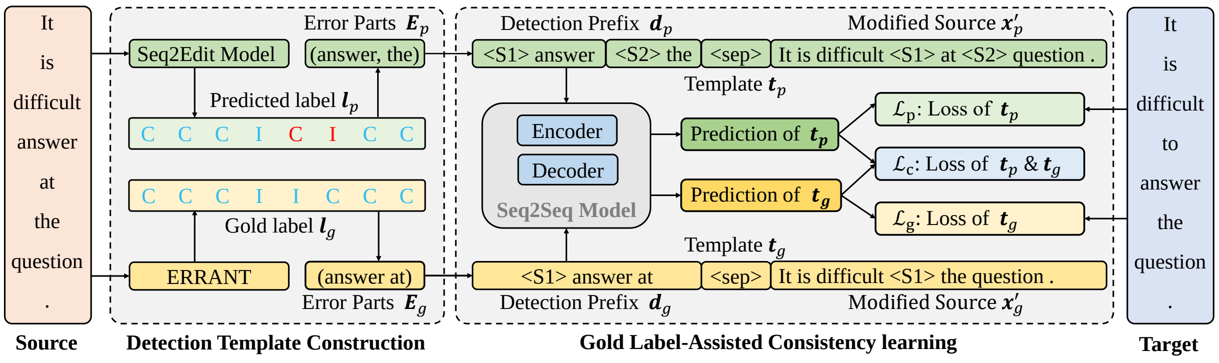

Figure 2: The overall framework of TemplateGEC.

sentence can be formulated as:

$$\boldsymbol{l} = l_1, l_2, ..., l_N, l_n \in \{\texttt{C}, \texttt{I}\}, \tag{1}$$

where C denotes correct and I denotes incorrect. As shown in the left part of Figure 2, the source sentence is transformed into two detection labels: a predicted label $\boldsymbol{l}_p$ and a gold label $\boldsymbol{l}_g$.

Predicted Label The predicted label represents the predicted error positions in a sentence obtained from a detection model. Due to its improved understanding capabilities, the Seq2Edit architecture is used to train a language model with a fully-connected layer as the output layer, which classifies the source tokens as correct or incorrect. We utilize the detection component of the Seq2Edit model to get the predicted label $\boldsymbol{l}_p$, which might contain errors. As shown in Figure 2, there are two detection errors marked as red in the predicted label $\boldsymbol{l}_p$.

Gold Label The gold label, which will be utilized by our model, indicates the true location of errors in a sentence. Based on the parallel source and target pairs, we use ERRANT (Bryant et al., 2017b) to extract the edits, from which we can obtain the gold label $\boldsymbol{l}_g$ of the source sentence.

3.2 Detection Template Construction

We introduce the detection template which incorporates detection information by transforming the input sentence in a specific manner. This template is constructed by concatenating a detection prefix with a modified version of the input sentence, utilizing a specialized token <sep> as a delimiter. The detection template $\boldsymbol{t}$ can be formulated as:

$$\boldsymbol{t} = \boldsymbol{d} \texttt{ <sep> } \boldsymbol{x}', \tag{2}$$

where $\boldsymbol{d}$ and $\boldsymbol{x}'$ represent the detection prefix and modified source, respectively. The detection template $\boldsymbol{t}$ is utilized as input for the Seq2Seq model, instead of the original source, as shown in Figure 2.

Detection Prefix The detection prefix is made up by concatenating the error parts and corresponding ordered special tokens. Error parts represent the continuous tokens that are labeled as I. A source sentence may contain multiple error parts, each comprising a varying number of words. As shown in Figure 2, there are two error parts annotated by the predicted label that are "answer" and "the", while there is only one error part "answer at" annotated by the gold label, due to the continuous label of I. We extract all the error parts $\boldsymbol{E} = E_1, E_2, ..., E_I$ of the source sentence according to the detection labels, then we use $\boldsymbol{d}$ to represent the detection prefix, which can be given by:

$$\boldsymbol{d} = S_1 E_1 S_2 E_2 ... S_I E_I, \tag{3}$$

where S_i is the i-th ordered special token. As shown in Figure 2, the detection prefix $\boldsymbol{d}_p$ is made up of two error parts and their corresponding special tokens <S1> and <S2> and so $\boldsymbol{d}_g$ is.

Detection Template All the error parts $\boldsymbol{E}$ with the number of I can divide the source sentence $\boldsymbol{x}$ into $I + 1$ correct parts, which can be given by:

$$\boldsymbol{x} = X_0 E_1 X_1 ... X_{I-1} E_I X_I, \tag{4}$$

where X_i denotes the i-th correct part of $\boldsymbol{x}$. Taking the predicted error parts $\boldsymbol{E}_p$ illustrated in Figure 2 for example, two error parts ("answer" and "the") divide the source sentence three parts ("It is difficult", "at" and "question"). Then the predicted modified source sentence $\boldsymbol{x}'_p$ is obtained by replacing the error parts, present in the source sentence, with corresponding ordered special tokens ("<S1>" and "<S2>"). The modified source sentence is:

$$\boldsymbol{x}' = X_0 S_1 X_1 ... X_{I-1} S_I X_I. \tag{5}$$

System	NLPCC18-Test (ZH)			BEA-Dev (EN)			CoNLL14-Test 1 (EN)			CoNLL14-Test 2 (EN)		
	P	R	$F_{0.5}$	P	R	$F_{0.5}$	P	R	$F_{0.5}$	P	R	$F_{0.5}$
ELECTRA(Yuan et al., 2021)	-	-	-	72.8	46.9	65.6	55.2	39.8	51.2	76.4	40.1	64.7
GECToR (Omelianchuk et al., 2020)	-	-	-	75.4	52.6	69.4	55.8	38.9	51.3	77.4	38.8	64.6
ELECTRA (Our Reproduced)	70.1	37.5	59.7	73.7	41.4	63.8	57.1	36.4	51.3	75.9	34.8	61.4

Table 1: Comparison of detection results for different systems. CoNLL14-Test 1 and 2 refer to different annotations.

Once $\boldsymbol{x}'_p$ is obtained, the predicted template $\boldsymbol{t}_p$ is constructed by concatenating the detection prefix $\boldsymbol{d}_p$ and modified source sentence $\boldsymbol{x}'_p$. So does $\boldsymbol{t}_g$.

3.3 Gold Label-Assisted Consistency learning

Motivation The proposed template incorporates detection information in the hope that the model can more accurately correct errors at the corresponding positions. However, when the detection information is incorrect, the model may make wrong modifications to correct words, resulting in a decrease of model performance. To overcome this problem and make the model more robust to the predicted error detection information, we propose using gold label-assisted consistency learning to help the model increase consistency in the output of the predicted detection template and gold detection template, thus improving the model performance.

Training Objective We adopt a Seq2Seq model based on the Transformer (Vaswani et al., 2017) architecture as the backbone of our method. As outlined in Section 3.1, we are able to obtain both predicted and gold detection labels for a given source sentence. The templates constructed by these types of labels, represented by $\boldsymbol{t}_p$ and $\boldsymbol{t}_g$ respectively, are then fed into the Seq2Seq model as shown in Figure 2. The two losses can be defined as:

$$\begin{aligned} \mathcal{L}_p &= -\log P(\boldsymbol{y}|\boldsymbol{t}_p;\boldsymbol{\theta}); \\ \mathcal{L}_g &= -\log P(\boldsymbol{y}|\boldsymbol{t}_g;\boldsymbol{\theta}), \end{aligned} \tag{6}$$

where $\boldsymbol{\theta}$ is the set of parameters to optimize, $\boldsymbol{y}$ is the target sequence.

Consistency Learning Following Liang et al. (2021); Wang et al. (2022a), we introduce the consistency loss to our model, which maximizes the similarity of the output distributions with predicted and gold detection information. KL divergence is a measure of the difference between two probability distributions, which is a non-symmetric measure. We set KL divergence as our consistency loss to maximize the consistency between the distributions of the predictions for $\boldsymbol{t}_p$ and $\boldsymbol{t}_g$, thus the consistency loss $\mathcal{L}_c$ is defined as:

$$\begin{aligned} \mathcal{L}_c = &\frac{1}{2}[KL(P(\boldsymbol{y}|\boldsymbol{t}_p;\boldsymbol{\theta})||P(\boldsymbol{y}|\boldsymbol{t}_g;\boldsymbol{\theta})) \\ &+ KL(P(\boldsymbol{y}|\boldsymbol{t}_g;\boldsymbol{\theta})||P(\boldsymbol{y}|\boldsymbol{t}_p;\boldsymbol{\theta}))]. \end{aligned} \tag{7}$$

This final training loss is:

$$\mathcal{L} = \frac{1}{2}(\mathcal{L}_p + \mathcal{L}_g) + \beta\mathcal{L}_c, \tag{8}$$

where β is a hyper-parameter representing the coefficient of consistency loss.

Discussion The two cross-entropy loss items encourage the model to generate the corresponding targets for the templates $\boldsymbol{t}_p$ and $\boldsymbol{t}_g$, which make the model learn how to construct the distributions of predicted and gold detection templates. Based on the two distributions, the consistency loss reduces the distance between the two distributions (Wang et al., 2022b; Li et al., 2022c; Liu et al., 2023). By enforcing consistency between predicted and gold labels, the model can learn more robust and reliable representations of the dataset, which can lead to improved performance for the GEC task. In the inference stage, only predicted detection labels are used to generate the template $\boldsymbol{t}_p$ which is fed into the model, since the gold detection label is not available that is suitable for practical application.

4 Experiments

4.1 Error Label Detection

Setup For the English, to obtain the predicted labels, we train a Seq2Edit model based on ELECTRA (Clark et al., 2020) using the same configurations following Yuan et al. (2021). Additionally, for the comparison of detection performance, we treat GECToR (Omelianchuk et al., 2020) as the detection model. We use the available best-trained RoBERTa model of GECToR[1] to infer the BEA19-Dev set and CoNLL14-Test set for English, and NLPCC18-Test for Chinese. We use ERRANT to extract the predicted labels according to the hypotheses of GECToR. For Chinese, as there are

[1]https://github.com/grammarly/gector

Configuration	English			Chinese	
Architecture	Transformer-large	BART-large	T5-large	Transformer-large	BART-large
Epochs	30	20	5	30	10
Max Tokens	16384	4096	2048	8192	2048
Learning Rate	5×10^{-4}	1×10^{-5}	1×10^{-3}	5×10^{-4}	3×10^{-5}
Optimizer	Adam (Kingma and Ba, 2015) $(\beta_1 = 0.9, \beta_2 = 0.98, \epsilon = 1 \times 10^{-6})$		Adafactor (Shazeer and Stern, 2018)	Adam (Kingma and Ba, 2015) $(\beta_1 = 0.9, \beta_2 = 0.98, \epsilon = 1 \times 10^{-6})$	
Warmup	4000	8000	4000	2000	2000
Loss Function	label smoothed cross entropy (label-smoothing=0.1) (Szegedy et al., 2016)				
Dropout	0.1	0.3	0.3	0.1	0.3
Beam Size	5	5	5	12	12

Table 2: Hyper-parameters of English and Chinese GEC Experiments.

Language	Corpus	Train	Dev	Test
English	cLang-8	2,372,119	-	-
English	WI, LOCNESS	-	4,384	4,477
English	CoNLL-14	-	-	1,312
Chinese	NLPCC18	1,377,172	-	2,000
Chinese	MuCGEC	-	2,467	-

Table 3: Statistics of the used datasets for GEC.

no public detection results for NLPCC18-Test, we train the same Chinese Seq2Edit model as the English experiment to obtain the predicted labels. Besides, ERRANT is used to extract the gold detection labels from all the training data.

Comparasion Results As shown in Table 1, the results of ELECTRA we reproduced are lower than the other two models for English datasets. Based on the superior performance of the GECToR model on the BEA-Dev set and CoNLL14-Test 1 set, as well as its proximity to the best results of another dataset, we select GECToR as the detection model for obtaining the detection labels.

Error Label Preparation Based on the results, we use the open-sourced GECToR model to detect all the English data and our reproduced ELECTRA model to produce predicted labels for Chinese data. Gold labels are extracted by ERRANT.

4.2 Grammatical Error Correction

Dataset For the English, we use cLang-8 (Rothe et al., 2021) as training data, which is a clean version of the original Lang-8 corpus (Mizumoto et al., 2011; Tajiri et al., 2012). Following Yuan et al. (2021), we use BEA-Dev (Bryant et al., 2019) and BEA-Test as the development and test datasets, both of which consist of W&I (Yannakoudakis et al., 2018) and LOCNESS (Granger, 2014). Additionally, we include the CoNLL14-Test set (Ng et al., 2014) in the test dataset. For the Chinese, following Zhang et al. (2022b), we use NLPCC18-Train (Zhao et al., 2018) as the training set, MuCGEC-Dev (Zhang et al., 2022a) as the development set and NLPCC18-Test as the test set. Table 3 shows the statistics of the used datasets.

Model The models we use are based on Transformer (Vaswani et al., 2017) architecture. For English, Transformer-large, BART-large (Lewis et al., 2020) and T5-large (Raffel et al., 2020) are first used as our baseline models, which are fine-tuned with the original format of training data. For Chinese, Transformer-large and Chinese BART-large (Shao et al., 2021) models are used as the baseline in the same way. Due to the absence of a Chinese version of the T5 model, the experiments conducted in Chinese do not incorporate the use of the T5 model. Then we train the models mentioned above with only the predicted template for comparison rather than the original sources. At last, the proposed TemplateGEC is trained with the predicted and gold template described in 3.2, and consistency loss is applied in the training stage. The hyper-parameter β is set to 1 and other main hyper-parameters used in English and Chinese experiments are shown in Table 2. All experiments are run on a GeForce RTX 3090 GPU.

Evaluation Metrics For English experiments, following Yuan et al. (2021), we use ERRANT and M^2 (Dahlmeier and Ng, 2012) to evaluate GEC models on BEA-Test set and CoNLL14-Test set, respectively. For Chinese experiments, following Zhang et al. (2022b), we use M^2 as the metric on the NLPCC18-Test set. Precision, recall, and F$_{0.5}$ values are reported for all the experiments.

Comparison with Previous Works Table 4 shows the main results for English and Chinese GEC tasks, which are compared with previous sin-

System	Proposed Methods		Detection Label		NLPCC18-Test (ZH)			BEA-Test (EN)			CoNLL14-Test (EN)		
	Template	Consistency	Train	Test	P	R	$F_{0.5}$	P	R	$F_{0.5}$	P	R	$F_{0.5}$
GECToR	×	×	-	-	-	-	-	79.2	53.9	72.4	77.5	40.1	65.3
Multi-encoder	×	×	-	-	-	-	-	73.3	61.5	70.6	71.3	44.3	63.5
T5-large	×	×	-	-	-	-	-	-	-	72.1	-	-	66.1
Type-Driven	×	×	-	-	-	-	-	81.3	51.6	72.9	78.2	42.7	67.0
SynGEC	×	×	-	-	50.0	33.0	45.3	75.1	65.5	72.9	74.7	49.0	67.6
Transformer	×	×	-	-	36.1	19.9	31.0	56.2	51.5	55.2	59.3	39.9	54.0
	✓	×	Pred	Pred	37.2	23.9	33.5	60.0	51.7	58.1	61.1	40.0	55.3
	✓	✓	Gold+Pred	Pred	42.0	22.2	35.6	67.8	50.7	63.5	64.7	38.9	57.1
BART	×	×	-	-	48.8	**33.5**	44.7	70.4	60.0	68.0	67.1	47.1	61.9
	✓	×	Pred	Pred	52.2	27.9	44.5	71.7	61.5	69.4	67.6	48.5	62.6
	✓	✓	Gold+Pred	Pred	**54.5**	27.4	**45.5**	74.8	61.0	71.6	69.7	46.7	63.5
T5	×	×	-	-	-	-	-	74.2	**66.5**	72.5	71.8	**50.8**	66.3
	✓	×	Pred	Pred	-	-	-	74.6	64.4	72.3	72.4	50.7	66.7
	✓	✓	Gold+Pred	Pred	-	-	-	**76.8**	64.8	**74.1**	**74.8**	50.0	**68.1**

Table 4: Results on Chinese (ZH) NLPCC18-Test, Enlgish (EN) BEA-Test and CoNLL14-Test sets."Template" denotes only using our template for the GEC task, and "Consistency" denotes the proposed gold label-assisted consistency learning. The items "Pred" and "Gold" denote the predicted and gold label, respectively.

gle models. GECToR (Omelianchuk et al., 2020) treats GEC as a sequence labeling task and assigns the proposed operation labels to each token in the source sentence. Multi-encoder (Yuan et al., 2021) additionally employs an encoder to handle the detection input and uses a re-ranking strategy based on the detection outputs to improve the GEC performance. T5-large (Rothe et al., 2021) directly takes the original source sentence as input and generates the prediction outputs with T5-large. Type-Driven (Lai et al., 2022) proposes a TypeDriven Multi-Turn Corrections approach for GEC, which trains the model to exploit interdependence between different types of errors. SynGEC (Zhang et al., 2022b) adapts the dependency syntax into GEC models to improve performance.

Main Results As shown in Table 4, utilizing pre-trained models results in a marked improvement in performance across all datasets, in comparison to models that are not pre-trained. Compared to the baseline, when the detection template is introduced, the performance of the majority of the models improves, particularly in models that are not pre-trained, while the improvement in pre-trained models is less significant. The results with weak or declining performance may be attributed to the possibility that the model is not effectively addressing errors present in the template. The proposed method TemplateGEC, utilizing both the template and consistency learning, achieves the best $F_{0.5}$ values on all the datasets when compared to other methods. This indicates that the incorporation of consistency learning allows the model to make more accurate corrections with the help of error labels provided in the template. The improvement of the proposed methods is primarily driven by an increase in precision, with some recall values experiencing a decline. This phenomenon is encouraged in GEC tasks since ignoring an error is not as bad as proposing a wrong correction (Ng et al., 2014).

5 Analysis

5.1 Potential of Human-in-the-loop GEC

Upper Bound Results In order to determine the performance upper bound of TemplateGEC, we initially evaluate its performance using gold detection labels during the testing stage. Subsequently, we conduct additional experiments where the gold labels are utilized both in the training and testing stages. As shown in Tabel 5, in contrast to the benchmark models and the label-based TemplateGEC utilizing predicted labels, utilizing gold labels in the TemplateGEC results in a marked improvement in performance, especially when pre-trained knowledge is not fully introduced (i.e., the results of Transformer). This serves as evidence that the proposed template plays a significant role in impacting the performance of the GEC system. The upper bound results of both BART and T5 models exhibit a significant improvement and are relatively comparable. This suggests that by training and testing TemplateGEC with the correct error distribution, it is possible to achieve superior performance compared to the predicted error distribution.

System	Proposed Methods		Detection Label		NLPCC18-Test (ZH)			BEA-Dev (EN)			CoNLL14-Test (EN)		
	Template	Consistency	Train	Test	P	R	$F_{0.5}$	P	R	$F_{0.5}$	P	R	$F_{0.5}$
Transformer	×	×	-	-	36.1	19.9	31.0	45.5	31.7	41.8	59.3	39.9	54.0
	✓	✓	Gold+Pred	Pred	42.0	22.2	35.7	52.8	29.5	45.6	64.7	38.9	57.1
	✓	✓	Gold+Pred	Gold	47.5	27.3	41.4	55.3	35.2	49.6	63.9	42.2	57.9
	✓	×	Gold	Gold	48.3	48.5	48.4	51.0	51.8	51.1	59.4	56.7	58.8
BART	×	×	-	-	52.2	27.9	44.5	57.3	38.5	52.2	67.1	47.1	61.8
	✓	✓	Gold+Pred	Pred	54.5	27.4	45.5	60.7	39.0	54.6	69.7	46.7	63.5
	✓	✓	Gold+Pred	Gold	56.7	30.2	48.2	64.0	46.0	59.4	70.5	50.9	65.5
	✓	×	Gold	Gold	**59.7**	**55.4**	**58.8**	**68.9**	62.4	67.5	69.4	**63.6**	68.2
T5	×	×	-	-	-	-	-	58.9	43.1	54.8	71.8	50.8	66.3
	✓	✓	Gold+Pred	Pred	-	-	-	61.0	41.0	55.6	74.8	50.0	68.0
	✓	✓	Gold+Pred	Gold	-	-	-	63.7	45.9	59.1	**76.3**	51.9	69.7
	✓	×	Gold	Gold	-	-	-	68.8	**64.6**	**68.0**	72.8	62.7	**70.5**

Table 5: Results on NLPCC18-Test, BEA-Dev and CoNLL14-Test sets inferred by the templates with gold detection labels for different training methods. We use BEA-Dev here since we cannot access the gold labels of BEA-Test.

Setup	BEA-Dev			BEA-Test		
	P	R	$F_{0.5}$	P	R	$F_{0.5}$
TemplateGEC	61.0	41.0	**55.6**	76.8	64.8	**74.1**
GED Model (Default: GECToR)						
ELECTRA	60.6	40.4	55.1	76.3	64.1	73.5
Detection Class (Default: 2-class)						
4-class	60.1	41.0	55.0	74.4	64.0	72.1
Detection Template (Default: t)						
t^s	60.7	41.2	55.5	76.3	64.9	73.7
Loss Type (Default: KL)						
MSE	58.9	42.3	54.6	72.8	66.5	71.4
Coefficient of Consistency Loss (Default: $\beta = 1$)						
$\beta = 2$	62.2	38.9	55.5	77.1	62.1	73.5
$\beta = 3$	62.6	36.6	54.8	78.4	60.1	73.9

Table 6: Comparison results of different setups.

Potential Direction In addition to the proposed TemplateGEC, we contend that our method has the potential for application in the development of a human-in-the-loop GEC system. We envision a scenario where a user inputs a sentence in need of correction or refinement, and our GEC model supports the identification of specific spans within the sentence that the user has identified as being in error or uncertain. Given this scenario, the TemplateGEC system can convert the identified error spans to the template format outlined in Section 3.2, resulting in the GEC model placing increased emphasis on these specific areas during the correction. The incorporation of user interaction in the TemplateGEC system allows for the utilization of user-annotated spans as the gold standard for error detection labels, resulting in improved error correction capabilities and increased efficiency as the need for a separate detection model is eliminated.

5.2 Ablation Study

In order to evaluate the effectiveness of the various components in TemplateGEC, we conduct multiple experimental evaluations using a variety of model configurations, testing them on both the BEA-Dev and BEA-Test datasets. In each experimental setting, we conduct evaluations, varying only one component while keeping the remaining constant.

Effect of Detection Model Given that the proposed template method incorporates the use of detection label knowledge, the performance of the TemplateGEC may be impacted by the performance of various detection models. In order to investigate the relationship between TemplateGEC and the detection model, we replace the GECToR model with the ELECTRA-based detection model described in Section 4.1. The results show that both detection models can produce reliable detection labels, indicating that the proposed method can accommodate various detection models.

Effect of Detection Class As stated in Yuan et al. (2021), the performance of GEC models incorporating detection labels is influenced by the choice of detection class. The results indicate that the $F_{0.5}$ score of the 4-class detection is slightly lower than that of the 2-class detection on average. As the performance of the 4-class detection model is suboptimal, further research is required to explore methods to enhance the TemplateGEC system with more fine-grained class detection labels.

Effect of Detection Template To investigate the significance of designing appropriate detection templates, we create a simple template that concatenates the detection prefix with the original source

Template	Consistency	ERR = 0 (42.4%) P	R	$F_{0.5}$	ERR = 1 (30.5%) P	R	$F_{0.5}$	ERR = 2 (16.5%) P	R	$F_{0.5}$	ERR = 3 (7.5%) P	R	$F_{0.5}$	ERR > 3 (3.1%) P	R	$F_{0.5}$
×	×	65.4	35.1	**55.8**	72.8	51.2	67.1	73.5	55.9	69.1	75.2	58.1	71.0	68.3	56.1	**65.5**
✓	×	64.3	31.5	53.2	75.0	51.5	68.7	74.7	56.5	**70.2**	75.7	60.3	72.0	64.4	54.6	62.2
✓	✓	68.2	31.9	55.6	78.6	51.6	**71.2**	74.6	54.6	69.5	77.4	60.1	**73.2**	69.3	52.7	65.2

Table 7: Results of error numbers in the source of CoNLL14-Test set. ERR denotes the number of errors.

	Example 1	Example 2
Source	She decided to divorce with her husband .	Therefore there is nothing to be shy for or be afraid of .
Target	She decided to divorce her husband .	Therefore there is nothing to be shy about or be afraid of .
Predicted Label	C C C C I C C C	C I C C C C C I C C C C C
Gold Label	C C C C I C C C	C C C C C C C I C C C C C
Vanilla	She decided to divorce with her husband .	Therefore , there is nothing to be shy of or be afraid of .
w/ Template	She decided to divorce from her husband .	Therefore , there is nothing to be shy about or be afraid of .
w/ Template&Consistency	She decided to divorce her husband .	Therefore there is nothing to be shy about or be afraid of .

Table 8: Examples from CoNLL14-Test set.

Type	Baseline P	R	$F_{0.5}$	TemplateGEC P	R	$F_{0.5}$
M	75.1	72.2	74.5	76.6	69.0	**74.9**
R	73.5	63.2	71.2	76.3	62.4	**73.0**
U	75.4	72.2	74.7	80.7	68.7	**78.0**

Table 9: Results of error types in BEA-Test.

sentence without any reformatting of the source. The simple template $\boldsymbol{t}^s$ is:

$$\boldsymbol{t}^s = \boldsymbol{d} \texttt{<sep>} \boldsymbol{x}. \tag{9}$$

The results show that the use of the simple template results in a decrease in performance, highlighting the effectiveness and appropriateness of our method in incorporating detection labels.

Effect of Consistency Loss Type Various loss functions can be used to measure how different two probability distributions are from each other, to find the divergence by employing different loss functions, we change the KL divergence loss to the Mean Squared Error (MSE) loss between two output representations. The results show that when MSE loss is adopted as the consistency function, a certain degree of performance degradation will be observed, which indicates that KL divergence loss is more appropriate for enhancing the model performance in our method.

Effect of Coefficient of Consistency Loss To learn the influence of the coefficient of consistency loss, we test several different values of β. Default value of $\beta = 1$, and we test for $\beta = 2$ and $\beta = 3$. The results show that our default setting $\beta = 1$ can get the best $F_{0.5}$ score. The result reveals that the consistency between the predicted and gold predictions is not always the higher the better.

5.3 Model Robustness

Error Detection Robustness As shown in Table 7, we explore the performance of different models under various sentence error numbers and the results. The baseline model achieves the best results when the error number is zero or more than three, while the template-only method shows a weak performance. It may be due to the unbalanced data distribution and the performance of the detection. Based on the results, TemplateGEC is still competitive in the two situations and outperforms the baseline in other situations, which is attributed to consistency learning. By introducing gold labels, the model is guided in the right direction even though the predicted labels may be wrong. It indicates that TemplateGEC is robust for different error numbers and performs better when there are few errors.

Error Type Robustness To explore if TemplateGEC can correct every error type well, the results of three error types, which are categorized as M (Missing), R (Replacement), and U (Unnecessary), are computed and shown in Table 9. Results show that compared with the baseline model, TemplateGEC gets the better $F_{0.5}$ score for all error types, especially the replacement and unnecessary types, which demonstrates the robustness of TemplateGEC on the error type level.

Case Study Table 8 illustrates how TemplateGEC works better than the baseline model in terms of model robustness. For the first example, the baseline model fails to correct the error. In contrast, despite the correction being incorrect, the template-only model attempt to correct the error indicated by the predicted label, which also confirms the effectiveness of the template. Based on the results of the template-only model, TemplateGEC successfully corrects the unnecessary type of error, corresponding to the ability of TemplateGEC reflected in Table 9. For the second example, the template-only model still modifies the corresponding positions indicated by the template, but one of them is wrongly corrected, which is misguided by the predicted label. However, we surprisingly observe that TemplateGEC ignores this misdirection and corrects the whole source sentence successfully. This result strongly suggests that our model can make correct corrections even when the prior information is wrong, which fully demonstrates the reliability and robustness of our method.

6 Conclusion

This paper presents a new method for GEC, called TemplateGEC, which integrates the Seq2Edit and Seq2Seq frameworks, leveraging their strengths in error detection and correction. TemplateGEC converts the original erroneous sentence into a novel template format that incorporates predicted and gold error detection labels, which are generated by a Seq2Edit model. Besides, TemplateGEC incorporates gold label-assisted consistency learning to enhance performance by maximizing the consistency between the predictions of the predicted and gold templates through the use of a Seq2Seq model. Experimental results on widely-used English and Chinese benchmarks show that TemplateGEC exhibits competitive performance in comparison to previous GEC methods. Additional analysis suggests that the proposed method is a promising approach for human-in-the-loop GEC and confirms that TemplateGEC is effective and robust. We will investigate the feasibility of adapting TemplateGEC to other languages and assess its effectiveness through additional experimentation in our future work.

Limitations

The primary limitation of the proposed model is computational efficiency. Specifically, during the training phase, the input size of the model is more than double that of traditional models, which is due to the inclusion of both predicted and gold templates. Besides, the source sentences are transformed into longer sequences, resulting in an increased memory footprint and longer training time. Additionally, both during the training and testing phase, an additional step of preparing detection labels for the data further contributes to the increased processing time. In future research, we aim to investigate methods for achieving comparable or superior performance while reducing the input size and addressing these limitations, building upon the foundation of our current work. Additionally, TemplateGEC does not support the joint training of the Seq2Edit model. We will further explore how to jointly train the Seq2Edit model in future work, particularly focusing on the continuous modeling of detection labels based on an end-to-end model.

Ethics Statement

Our work aims to develop and evaluate algorithms that automatically detect and correct grammatical errors in written English and Chinese text. We use publicly available datasets for training and evaluation purposes. These datasets consist of anonymized and de-identified text samples, ensuring the privacy and confidentiality of the original authors. We are committed to conducting our research in an ethical and responsible manner.

Acknowledgments

Heyan Huang is the corresponding author. This work was supported in part by the National Natural Science Foundation of China (Grant Nos. U21B2009, 62206076), the Science and Technology Development Fund, Macau SAR (Grant Nos. FDCT/060/2022/AFJ, FDCT/0070/2022/AMJ), Shenzhen College Stability Support Plan (Grant Nos. GXWD20220811173340003, GXWD20220817123150002), Shenzhen Science and Technology Program (Grant No. RCBS20221008093121053), CCF Fund for Young Scholars and the Multi-year Research Grant from the University of Macau (Grant No. MYRG2020-00054-FST).

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.

Christopher Bryant, Mariano Felice, Øistein E. Andersen, and Ted Briscoe. 2019. The BEA-2019 shared task on grammatical error correction. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 52–75, Florence, Italy. Association for Computational Linguistics.

Christopher Bryant, Mariano Felice, and Ted Briscoe. 2017a. Automatic annotation and evaluation of error types for grammatical error correction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 793–805, Vancouver, Canada. Association for Computational Linguistics.

Christopher Bryant, Mariano Felice, and Ted Briscoe. 2017b. Automatic annotation and evaluation of error types for grammatical error correction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 793–805, Vancouver, Canada. Association for Computational Linguistics.

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 *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7162–7169, Online. Association for Computational Linguistics.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: pre-training text encoders as discriminators rather than generators. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Daniel Dahlmeier and Hwee Tou Ng. 2012. Better evaluation for grammatical error correction. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 568–572, Montréal, Canada. Association for Computational Linguistics.

Tao Fang, Jinpeng Hu, Derek F. Wong, Xiang Wan, Lidia S. Chao, and Tsung-Hui Chang. 2023a. Improving grammatical error correction with multimodal feature integration. In *Findings of the Association for Computational Linguistics: ACL 2023*, Toronto, Canada. Association for Computational Linguistics.

Tao Fang, Xuebo Liu, Derek F. Wong, Runzhe Zhan, Liang Ding, Lidia S. Chao, Dacheng Tao, and Min Zhang. 2023b. Transgec: Improving grammatical error correction with translationese. In *Findings of the Association for Computational Linguistics: ACL 2023*, Toronto, Canada. 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.

Peiyuan Gong, Xuebo Liu, Heyan Huang, and Min Zhang. 2022. Revisiting grammatical error correction evaluation and beyond. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*.

Sylviane Granger. 2014. The computer learner corpus: a versatile new source of data for sla research. In *Learner English on computer*, pages 3–18. Routledge.

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 *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4248–4254, Online. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Shun Kiyono, Jun Suzuki, Tomoya Mizumoto, and Kentaro Inui. 2020. Massive exploration of pseudo data for grammatical error correction. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28:2134–2145.

Kate M. Knill, Mark J. F. Gales, P. P. Manakul, and Andrew Caines. 2019. Automatic grammatical error detection of non-native spoken learner english. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2019, Brighton, United Kingdom, May 12-17, 2019*, pages 8127–8131. IEEE.

Shaopeng Lai, Qingyu Zhou, Jiali Zeng, Zhongli Li, Chao Li, Yunbo Cao, and Jinsong Su. 2022. Type-driven multi-turn corrections for grammatical error correction. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 3225–3236, Dublin, Ireland. Association for Computational Linguistics.

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

Bei Li, Quan Du, Tao Zhou, Yi Jing, Shuhan Zhou, Xin Zeng, Tong Xiao, JingBo Zhu, Xuebo Liu, and Min Zhang. 2022a. ODE transformer: An ordinary differential equation-inspired model for sequence generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8335–8351, Dublin, Ireland. Association for Computational Linguistics.

Jiquan Li, Junliang Guo, Yongxin Zhu, Xin Sheng, Deqiang Jiang, Bo Ren, and Linli Xu. 2022b. Sequence-to-action: Grammatical error correction with action guided sequence generation. *arXiv e-prints*, pages arXiv–2205.

Zhaocong Li, Xuebo Liu, Derek F. Wong, Lidia S. Chao, and Min Zhang. 2022c. ConsistTL: Modeling consistency in transfer learning for low-resource neural machine translation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 8383–8394, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Xiaobo Liang, Lijun Wu, Juntao Li, Yue Wang, Qi Meng, Tao Qin, Wei Chen, Min Zhang, and Tie-Yan Liu. 2021. R-drop: Regularized dropout for neural networks. In *Advances in Neural Information Processing Systems*, volume 34, pages 10890–10905. Curran Associates, Inc.

Junwei Liao, Sefik Emre Eskimez, Liyang Lu, Yu Shi, Ming Gong, Linjun Shou, Hong Qu, and Michael Zeng. 2020. Improving readability for automatic speech recognition transcription. *ArXiv preprint*, abs/2004.04438.

Jared Lichtarge, Chris Alberti, Shankar Kumar, Noam Shazeer, Niki Parmar, and Simon Tong. 2019. Corpora generation for grammatical error correction. 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 3291–3301, Minneapolis, Minnesota. Association for Computational Linguistics.

Shudong Liu, Xuebo Liu, Derek F. Wong, Zhaocong Li, Wenxiang Jiao, S. Chao Lidia, and Min Zhang. 2023. knn-tl: k-nearest-neighbor transfer learning for low-resource neural machine translation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Toronto, Canada. Association for Computational Linguistics.

Xuebo Liu, Longyue Wang, Derek F. Wong, Liang Ding, Lidia S. Chao, and Zhaopeng Tu. 2021. Understanding and improving encoder layer fusion in sequence-to-sequence learning. In *International Conference on Learning Representations*.

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.

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 *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 147–155, Chiang Mai, Thailand. Asian Federation of Natural Language Processing.

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. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–14, Baltimore, Maryland. Association for Computational Linguistics.

Hwee Tou Ng, Siew Mei Wu, Yuanbin Wu, Christian Hadiwinoto, and Joel Tetreault. 2013. The CoNLL-2013 shared task on grammatical error correction. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–12, Sofia, Bulgaria. 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.

Chanjun Park, Yeongwook Yang, Chanhee Lee, and Heuiseok Lim. 2020. Comparison of the evaluation metrics for neural grammatical error correction with overcorrection. *IEEE Access*, 8:106264–106272.

Maja Popović. 2018. Error classification and analysis for machine translation quality assessment. In *Translation quality assessment*, pages 129–158. Springer.

Muhammad Qorib, Seung-Hoon Na, and Hwee Tou Ng. 2022. Frustratingly easy system combination for grammatical error correction. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1964–1974, Seattle, United States. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Hongkai Ren, Liner Yang, and Endong Xun. 2018. A sequence to sequence learning for chinese grammatical error correction. In *Natural Language Processing and Chinese Computing: 7th CCF International Conference, NLPCC 2018, Hohhot, China, August 26–30, 2018, Proceedings, Part II*, pages 401–410. Springer.

Sascha Rothe, Jonathan Mallinson, Eric Malmi, Sebastian Krause, and Aliaksei Severyn. 2021. A simple recipe for multilingual 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 2: Short Papers)*, pages 702–707, Online. Association for Computational Linguistics.

Yunfan Shao, Zhichao Geng, Yitao Liu, Junqi Dai, Fei Yang, Li Zhe, Hujun Bao, and Xipeng Qiu. 2021. Cpt: A pre-trained unbalanced transformer for both chinese language understanding and generation. *ArXiv preprint*, abs/2109.05729.

Noam Shazeer and Mitchell Stern. 2018. Adafactor: Adaptive learning rates with sublinear memory cost. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 4603–4611. PMLR.

Felix Stahlberg and Shankar Kumar. 2020. Seq2Edits: Sequence transduction using span-level edit operations. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5147–5159, Online. Association for Computational Linguistics.

Xin Sun, Tao Ge, Shuming Ma, Jingjing Li, Furu Wei, and Houfeng Wang. 2022. A unified strategy for multilingual grammatical error correction with pretrained cross-lingual language model. *ArXiv preprint*, abs/2201.10707.

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.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 2818–2826. IEEE Computer Society.

Toshikazu Tajiri, Mamoru Komachi, and Yuji Matsumoto. 2012. Tense and aspect error correction for ESL learners using global context. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 198–202, Jeju Island, Korea. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Shuo Wang, Peng Li, Zhixing Tan, Zhaopeng Tu, Maosong Sun, and Yang Liu. 2022a. A template-based method for constrained neural machine translation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 3665–3679, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Yu Wang, Yuelin Wang, Kai Dang, Jie Liu, and Zhuo Liu. 2021. A comprehensive survey of grammatical error correction. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 12(5):1–51.

Zhijun Wang, Xuebo Liu, and Min Zhang. 2022b. Breaking the representation bottleneck of Chinese characters: Neural machine translation with stroke sequence modeling. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 6473–6484, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Helen Yannakoudakis, Øistein E Andersen, Ardeshir Geranpayeh, Ted Briscoe, and Diane Nicholls. 2018. Developing an automated writing placement system for esl learners. *Applied Measurement in Education*, 31(3):251–267.

Michihiro Yasunaga, Jure Leskovec, and Percy Liang. 2021. LM-critic: Language models for unsupervised grammatical error correction. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7752–7763, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Zheng Yuan and Ted Briscoe. 2016. Grammatical error correction using neural machine translation. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 380–386, San Diego, California. Association for Computational Linguistics.

Zheng Yuan, Shiva Taslimipoor, Christopher Davis, and Christopher Bryant. 2021. Multi-class grammatical error detection for correction: A tale of two systems. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8722–8736, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Yue Zhang, Zhenghua Li, Zuyi Bao, Jiacheng Li, Bo Zhang, Chen Li, Fei Huang, and Min Zhang.

2022a. MuCGEC: a multi-reference multi-source evaluation dataset for Chinese grammatical error correction. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3118–3130, Seattle, United States. Association for Computational Linguistics.

Yue Zhang, Bo Zhang, Zhenghua Li, Zuyi Bao, Chen Li, and Min Zhang. 2022b. Syngec: Syntax-enhanced grammatical error correction with a tailored gec-oriented parser. In *Proceedings of EMNLP*.

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.

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

Junpei Zhou, Chen Li, Hengyou Liu, Zuyi Bao, Guangwei Xu, and Linlin Li. 2018. Chinese grammatical error correction using statistical and neural models. In *Natural Language Processing and Chinese Computing: 7th CCF International Conference, NLPCC 2018, Hohhot, China, August 26–30, 2018, Proceedings, Part II 7*, pages 117–128. Springer.

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.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
Section 7

☒ A2. Did you discuss any potential risks of your work?
The data we used are publicly available and do not contain this issue.

☑ A3. Do the abstract and introduction summarize the paper's main claims?
Section 1, 6

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

Section 3

☑ B1. Did you cite the creators of artifacts you used?
Section 4

☒ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
The datas we used are public available and do not contain this issue.

☐ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
Not applicable. The data we used are publicly available and do not contain this issue.

☐ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
Not applicable. The data we used are publicly available and do not contain this issue.

☐ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Not applicable. The data we used are publicly available and do not contain this issue.

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
Section 4

C ☑ Did you run computational experiments?

Section 4

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Section 4

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Section 4

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
Section 4

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Section 4

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?

The data we used are publicly available and do not contain this issue.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
Not applicable. Left blank.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
Not applicable. Left blank.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
Not applicable. Left blank.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
Not applicable. Left blank.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
Not applicable. Left blank.