InstructGEC: Enhancing Unsupervised Grammatical Error Correction with Instruction Tuning

Jiayi Deng[1,2] and Chen Chen[1,2]* and Chunyan Hou[3] and Xiaojie Yuan[1,2]
[1] College of Computer Science, Nankai University, Tianjin, China
[2] MoE Key Lab of DISSec, Nankai University, Tianjin, China
[3] School of CSE, Tianjin University of Technology, Tianjin, China
2120220621@mail.nankai.edu.cn,nkchenchen@nankai.edu.cn,
houchunyan@tjut.edu.cn,yuanxj@nankai.edu.cn

Abstract

Recent works have proposed methods of generating synthetic data automatically for unsupervised Grammatical Error Correction (GEC). Although a large amount of synthetic data is generated at a low cost, it is unrealistic and of poor quality. The copying phenomenon of synthetic data prevents GEC models from learning the semantic knowledge of contextual language. In this paper, we design an instruction format and consistently use the masking strategy in both an erroneous sentence and the corresponding instruction to alleviate the impact of the copy phenomenon. We also propose a novel approach, InstructGEC, which integrates the knowledge of grammatical detection into GEC models with instruction tuning to address the low-quality issue. Experiments are conducted on English and Chinese GEC datasets and results demonstrate that our method outperforms state-of-the-art unsupervised GEC methods.

1 Introduction

Grammatical Error Correction (GEC) aims to detect and correct grammatical errors in an erroneous sentence and output a correct sentence. It has attracted a lot of attention due to its broad applications. There has been a large amount of research on GEC (Bryant et al., 2019; Rothe et al., 2021; Zhang et al., 2022; Zhao et al., 2018). However, a common drawback of these approaches is their dependence on a large amount of manually labeled data, which is time-consuming and expensive to construct.

To address this limitation, synthetic data generation for GEC is proposed to provide training data. Synthetic data can be generated in a supervised or unsupervised way. Supervised methods of synthetic data generation (Awasthi et al., 2019; Kiyono et al., 2019; Lichtarge et al., 2019; Stahlberg and Kumar, 2021) require manually labeled GEC data while unsupervised methods (Sun et al., 2022; Yasunaga et al., 2021; Zhao et al., 2019) do not need it. Because there is no manually labeled data for unsupervised GEC, many works focus on unsupervised synthetic data generation. Zhao et al. (2019) presented a random noising method to corrupt sentences to generate synthetic data. Yasunaga et al. (2021) proposed the Break-It-Fix-It (BIFI) framework to extract parallel data from unlabeled data. Sun et al. (2022) adopted machine translation pairs and pre-trained language models for erroneous sentence generation.

These unsupervised generation methods usually corrupt normal sentences to construct erroneous sentences automatically by heuristic rules. Then, unsupervised GEC models are trained on such synthetic data. Although a large amount of synthetic data is generated at a low cost, it is unrealistic and of poor quality. In addition, an erroneous sentence is similar to the corresponding correct one in synthetic data because grammatical errors are generally sparse. In this paper, it is named *copying phenomenon*. In this case, GEC models usually copy correct tokens directly from an erroneous sentence to the correct one and are prevented from learning the semantic knowledge of the contextual language. The performance of unsupervised GEC methods is much lower than supervised methods (Alikaniotis and Raheja, 2019; Yasunaga et al., 2021). Thus, the *copying phenomenon* and *low-quality* are challenges for unsupervised GEC.

To address these challenges, we propose a novel approach, InstructGEC, which is based on instruction tuning. Previous studies (Chen et al., 2020; Li et al., 2023) have explored dividing GEC into two stages: Grammatical Error Detection (GED) and Grammatical Error Correction (GEC). Grammatical errors are identified in the GED stage while errors are corrected in the GEC stage. InstructGEC attempts to enhance unsupervised GEC models by integrating GED knowledge into the GEC models

* Corresponding author.

Proceedings of the 31st International Conference on Computational Linguistics, pages 110–122
January 19–24, 2025.

with instruction tuning.

Specifically, instructions, that can identify positions and edit operations of errors, are designed to include GED knowledge. Then instruction tuning is used to train GEC models, and the GEC model is guided by instructions to output correct sentences. The low-quality synthetic data often gives rise to inaccurate instructions, and our proposed InstructGEC is enabled to bridge the gap between an inaccurate instruction and the corresponding accurate one. InstructGEC outputs correct sentences and improves the generalization ability of GEC models even though inaccurate instructions are input. Therefore, our method can alleviate the low-quality issue. Due to the copying phenomenon in synthetic data, GEC models can not learn rich semantic knowledge from such large-scale synthetic data. So we use the masking strategy in both an erroneous sentence and the corresponding instruction consistently to mitigate the harmful impact of the copying phenomenon.

We conduct experiments to validate the effectiveness of InstructGEC for the unsupervised GEC on both English and Chinese GEC datasets. The Experimental results demonstrate that our approach outperforms state-of-the-art baselines and effectively mitigates the impact of the low quality and copy phenomenon.

The contributions of our paper are as follows:

1. We design an instruction format and use the masking strategy in both an erroneous sentence and the corresponding instruction consistently to form a prompt. The prompt enables GEC models to learn correction patterns and alleviate the impact of the copy phenomenon simultaneously.

2. We propose a novel approach, InstructGEC, which can integrate GED knowledge into GEC models with instruction tuning for unsupervised GEC. To improve the generalization ability of the GEC model, our proposed approach addresses the low-quality issue by bridging the gap between an inaccurate instruction and the corresponding accurate one.

3. We conduct experiments on English and Chinese GEC datasets. Experimental results demonstrate that our method outperforms the state-of-the-art GEC methods.

2 Related Work

Grammatical Error Correction (GEC) There are two main lines of research in GEC: (i) Sequence-to-Edit (Seq2Edit) models, and (ii) Sequence-to-Sequence (Seq2Seq) models, typically based on Transformer (Vaswani, 2017) model. Seq2Edit approaches (Awasthi et al., 2019; Malmi et al., 2019; Omelianchuk et al., 2020) treat GEC as a sequence labeling task, while the Seq2Seq (Kaneko et al., 2020; Yang et al., 2023; Zhao et al., 2019) consider GEC as a monolingual translation task. Some Seq2Seq GEC methods have achieved impressive effectiveness. Zhao et al. (2019) proposed a copy-augmented architecture and incorporated multi-task learning into the GEC task. Kaneko et al. (2020) incorporated a pre-trained masked language model into GEC. Yang et al. (2023) leveraged the error type information in the generation process. These works require manually annotated data. However, few studies attempt to explore unsupervised GEC methods that do not rely on manually labeled data. Alikaniotis and Raheja (2019) provided grammatical corrections based on confusion sets and validated these corrections by language models. Yasunaga et al. (2021) proposed an unsupervised synthetic data generation method. Chen et al. (2023) explored the constituent syntax of synthetic data to improve the performance of the GEC model in an unsupervised setting. Coyne et al. (2023) conducted the GEC task on ChatGPT using a zero-shot or few-shot approach. In contrast to these works, our method focuses on alleviating issues caused by the low-quality and copying phenomenon of synthetic data for unsupervised GEC.

Previous studies have tried to incorporate detection knowledge into GEC models. Chen et al. (2020) identified erroneous text spans in the GED stage and only output the corrected text for these spans in the GEC stage. Yuan et al. (2021) adopted GED knowledge as an auxiliary input to fine-tune a GEC model and re-rank the GEC output. Li et al. (2023) proposed a detection template to introduce GED knowledge that allows the GEC model to make accurate predictions. Different from the above studies, our method integrates detection knowledge into GEC models with instruction tuning.

Due to the low error rate of erroneous sentences, the copy phenomenon is a common issue for GEC synthetic data. Wu et al. (2022); Zhao et al. (2019) incorporated the copying mechanism in their mod-

els to directly replicate words from the source sentence to the target. Shen et al. (2022) found that the copy phenomenon can harm the effective training of GEC models and proposed a masking strategy to facilitate effective training. Compared with these works, our method designs an instruction format and uses the masking strategy in both instructions and erroneous sentences to alleviate the impact of the copy phenomenon.

Instruction Tuning (IT) IT has proven to be an efficient method for improving the ability of generalization by learning from a collection of tasks (Wei et al.). Some studies attempted to convert the supervised data into the format of instructions to fine-tune LLMs. Zhang et al. (2023) transformed the supervised financial sentiment analysis data into instruction data by randomly selecting an instruction from ten instructions and fine-tuned LLMs. Wang et al. (2023) proposed InstructUIE, which is a unified information extraction framework based on IT to model different information extraction tasks. Our method differs from these works in that an instruction generated in the GED stage depends on the inputted erroneous sentence and we fine-tune GEC models on instruction-tuning data for unsupervised GEC task.

3 InstructGEC

The architecture of InstructGEC is shown in Figure 1. We describe the detection and correction stage of InstructGEC in Section 3.1 and 3.2 respectively, and introduce the training and inference in Appendix A.3.

3.1 Instruction Generation in Detection Stage

We regard Grammar Error Detection (GED) as a token classification task and design an instruction format that includes five sub-instructions. These sub-instructions at the token level are associated with edit operations on erroneous sentences. In Table 1, we define a sub-instruction set S which consists of five sub-instructions. The instruction I for a sentence $X = X_1, X_2, ..., X_n$ is a sequence of sub-instructions, which can be formulated as:

$$I = I_1, I_2, ..., I_n \quad (1)$$

where $I_i \in S$ denotes a sub-instruction, n is the number of tokens in the sentence X.

As shown in the upper half of Figure 1, an erroneous sentence is converted into two instructions I_t and I_p by a *sequence matcher* and a *sequence tagging model* respectively. *Sequence matcher* uses the Levenshtein distance algorithm (Levenshtein, 1966) to match differences between an erroneous sentence and the corresponding correct sentence at the token level. *Sequence matcher* can provide the instruction I_t, that is called ***gold instruction*** afterwards. We consider sub-instructions as the labels of tokens and use the training dataset to train a BERT (Kenton and Toutanova, 2019) model as the *sequence tagging model*. The tagging model can provide the instruction I_p, which is named ***predicted instruction***. Then, the instruction I_t and I_p are fed into the *prompt maker*. The *sequence matcher* and the *sequence tagging model* are designed to reduce the burden of Seq2Seq GEC Models because they are capable of integrating positions and editing operations of errors into an instruction.

Sub-Instruction	Description
<k>	Keep unchanged token
<r>	Replace token
<a>	Add token
<d>	Delete token
<m>	Mask token

Table 1: Sub-Instruction Set

3.2 Instruction Tuning in Correction Stage

Mask Strategy When an erroneous sentence is given, sub-instructions are provided based on the tokens of this erroneous sentence by *sequence tagging model* or *sequence matcher*. Then, *sequence mask marker* randomly masks a proportion of tokens whose sub-instructions are <k> with a mask token [MASK] in an erroneous sentence, and puts the mask sub-instruction <m> in the corresponding position of the instruction. Note that the masking strategy is only used for unchanged tokens whose sub-instructions are <k> in an erroneous sentence.

Specifically, 80% of masked tokens in an erroneous sentence are replaced with the [MASK] token, and the corresponding sub-instructions in the instruction are with <m>. 10% of the tokens are replaced with random tokens from the vocabulary, and the corresponding sub-instructions in the instruction are with <r>. There are three advantages of the proposed masking strategy. First, when GEC models are trained by erroneous sentences with masked tokens, they can learn the language semantic knowledge rather than direct copying. Second, when the masking strategy is only used for unchanged tokens, GEC models are not impeded

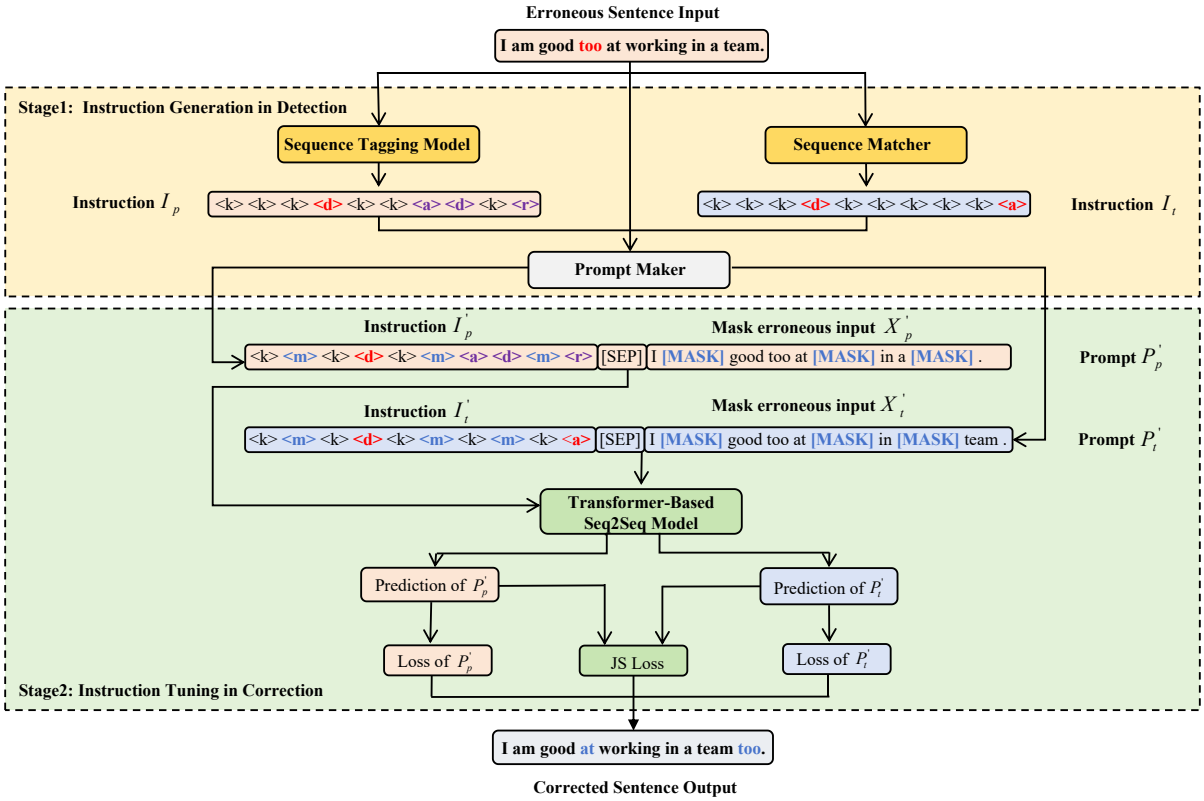

Figure 1: The architecture of InstructGEC, the detection stage in the upper half and the correction stage in the lower half. The red part of the instructions represents the sentence errors, the purple part represents tokens wrongly classified by the sequence labeling model, and the blue part denotes the masked segment.

from learning the correction from an erroneous sentence to a correct one. Third, the instructions provide the correction patterns and the masking strategy enables GEC models to alleviate the impact of copy phenomenon and improve the ability of pattern recognition.

Prompt Construction *Prompt maker* is responsible for generating the prompt. It constructs $(X_p^{'}, I_p^{'})$ and $(X_t^{'}, I_t^{'})$ by applying the masking strategy to (X, I_t) and (X, I_p) with the specified mask probability respectively. The prompt $P_t^{'}$ is the concatenation of $I_t^{'}$ and $X_t^{'}$ with a specialized token [SEP] as a delimiter while the prompt $P_p^{'}$ is of $I_p^{'}$ and $X_p^{'}$, which can be formulated as:

$$P_t^{'} = I_t^{'}[SEP]X_t^{'} \quad (2)$$

$$P_p^{'} = I_p^{'}[SEP]X_p^{'} \quad (3)$$

$P_t^{\prime}$ is constructed using $I_t^{'}$, and we refer to it as the ***gold prompt***. $P_p^{\prime}$ is constructed using $I_p^{'}$, and we refer to it as the ***predicted prompt***.

Loss Function As shown in the lower half of Figure 1, the prompt $P_p^{\prime}$ and $P_t^{\prime}$ are fed into a Transformer-based Seq2Seq model in the correction stage. This model outputs the prediction of $P_p^{\prime}$ and $P_t^{\prime}$ respectively. The loss $\mathcal{L}_s$ is defined as:

$$\mathcal{L}_s = -\frac{1}{2}[\log P(Y|P_p^{\prime};\theta) + \log P(Y|P_t^{\prime};\theta)] \quad (4)$$

where θ is the parameters of Transformer-based Seq2Seq model, Y is the correct sentences.

The prompt $P_p^{\prime}$ includes the instruction $I_p^{\prime}$ that is integrated with GED knowledge for correcting errors in an erroneous sentence exactly. Because the synthetic data is of low quality, the sequence tagging model might make mistakes in predicting the instruction $I_p^{\prime}$. When the instruction $I_p^{\prime}$ is incorrect, GEC models probably make the wrong correction to the erroneous sentence. To address this problem, we propose to use consistency training that can bridge the gap between the predicted instruction and the gold instruction. In detail, we take into account the consistency loss that penalizes the dissimilarity between the distribution of prediction for $P_t^{\prime}$ and $P_p^{\prime}$. The Jensen–Shannon (JS) divergence is a symmetrical metric used to measure the dissimilarity between two probability distributions in statistics. We use JS divergence as the consistency loss to alleviate the impact of the low-quality synthetic data. The consistency loss is

formulated as:

$$Avg = \frac{1}{2}[P(Y|P'_p;\theta) + P(Y|P'_t;\theta)] \quad (5)$$

$$\mathcal{L}_c = \frac{1}{2}[KL(P(Y|P'_p;\theta)||Avg) + KL(P(Y|P'_t;\theta)||Avg)] \quad (6)$$

where $\mathcal{L}_c$ is the consistency loss, KL is the Kullback–Leibler divergence.

The final loss function is defined as:

$$\mathcal{L} = \mathcal{L}_s + \lambda\mathcal{L}_c \quad (7)$$

where λ is the hyper-parameter that controls the weight of $\mathcal{L}_c$.

For the inference, given a new erroneous sentence x, the sequence tagging model is used to get just the predicted instruction of x. The prompt is constructed by concatenating the predicted instruction and x with a token [SEP] as a delimiter, and is fed into the Transformer-based Seq2Seq model which outputs the correct sentence of x. Note that the prompt is not masked in the inference stage.

4 Experiments

We introduce the datasets, evaluation metrics, and baselines in Sections 4.1, 4.2, and 4.3. The results analysis is described in Section 4.4. Implementation details are provided in Appendix A.1 and baseline settings are provided in Appendix A.2.

4.1 Datasets

Both Chinese and English GEC synthetic datasets generated in an unsupervised way are required in our experiments. For the Chinese dataset, we adopt a methodology that is similar to Tang et al. (2023). We treat the sentences as correct sentences and corrupt these sentences to construct erroneous sentences by adding, deleting, replacing, and swapping words or characters on different levels in translation29zh corpus[1]. As to the English dataset, we follow Awasthi et al. (2019). They perform heuristic algorithms on the sentences in the One-billion-word corpus (Chelba et al., 2013) by adding, deleting, and replacing to construct erroneous sentences. The size of Chinese and English synthetic data is 3.4M and 9M pairs respectively. The statistics of datasets are shown in Table 2.

We train the proposed InstructGEC model on the synthetic dataset. If the MuCGEC dev set is used as the validation data, experiments are conducted on the NLPCC2018 test and the MuCGEC test set. When the BEA-2019 test and the CoNLL-2014 test set are adopted for evaluation, we use the BEA-2019 dev set as the validation data.

Language	Corpus	Split	Pair
Chinese	ZH Synthetic data	train	3.40M
English	EN Synthetic data	train	9M
Chinese	MuCGEC test	test	5.93K
Chinese	NLPCC2018 test	test	2.00K
English	BEA-2019 test	test	4.48K
English	CoNLL-2014 test	test	1.31K
Chinese	MuCGEC dev	valid	1.12K
English	BEA-2019 dev	valid	4.38K

Table 2: Statistics of datasets used in experiments

4.2 Evaluation Metrics

In the detection stage, instruction generation is considered as a token classification task, and the quality of predicted instruction by the sequence tagging model is evaluated with the metrics (Nakayama, 2018) named F_1 in Table 6.

In the correction stage, the Transformer-based Seq2Seq model is evaluated with the official M^2 score metrics (Dahlmeier and Ng, 2012) on the NLPCC2018 test and the CoNLL-2014 test set. For the Chinese dataset, pkunlp (Draplater, 2018) is used for Chinese word segmentation. The ChERRANT (Zhang et al., 2022) metrics tool is used on the MuCGEC test set while the ERRANT (Bryant et al., 2017) is on the BEA-2019 dev/test set. Because the MuCGEC test and the BEA-2019 test set are not publicly available, we submit the results of MuCGEC test set on TianChi competition website[2] and the results of BEA-2019 test set on CodaLab Competition website[3].

4.3 Compared Methods

We compare InstructGEC with the following baselines.

Transformer (Vaswani, 2017) is a Transformer-based encoder-decoder Seq2Seq model trained on synthetic data.

GECToR (Omelianchuk et al., 2020) is a Seq2Edit model using a Transformer encoder and designs custom token-level transformations to map input tokens to target corrections.

[1] https://github.com/brightmart/nlp_chinese_corpus

[2] https://tianchi.aliyun.com/dataset/131328

[3] https://codalab.lisn.upsaclay.fr/competitions/4057

Methods	Mask	Instruction Type		NLPCC2018 test			MuCGEC test		
		Train	Test	P	R	$F_{0.5}$	P	R	$F_{0.5}$
Transformer	×	-	-	18.84	9.63	15.82	18.63	8.11	14.79
GECToR	×	-	-	34.45	**22.35**	31.08	35.81	**19.98**	30.91
BART	×	-	-	33.30	20.50	29.61	38.06	18.89	31.64
Ours	×	Gold+Pred	Pred	38.56	18.20	31.51	**43.53**	16.17	32.52
	✓	Gold+Pred	Pred	**39.92**	20.05	**33.31**	43.41	17.16	**33.24**

Table 3: Results on Chinese NLPCC18-test and MuCGEC-test set. Precision (P), Recall (R), $F_{0.5}$ ($F_{0.5}$) are reported (%). In the "Mask" column, the symbol "×" signifies the absence of the masking strategy while the symbol "✓" indicates using the masking strategy. In the "Instruction Type" column,"Gold" and "Pred" refer to the gold and predicted instruction respectively.

Methods	Mask	Instruction Type		CoNLL-2014 test			BEA-2019 test			BEA-2019 dev		
		Train	Test	P	R	$F_{0.5}$	P	R	$F_{0.5}$	P	R	$F_{0.5}$
Transformer	×	-	-	49.90	23.22	40.58	43.51	29.94	39.90	32.95	13.4	25.51
GPT-2 (Alikaniotis and Raheja, 2019)	×	-	-	58.5	24.9	46.1	-	-	-	-	-	-
BART-base	×	-	-	61.33	28.23	49.68	53.68	35.59	48.73	41.07	15.88	31.18
BART-base (Grundkiewicz et al., 2019)	×	-	-	59.7	18.5	41.3	62.4	25.4	48.8	-	-	-
ChatGPT with prompting (Coyne et al., 2023)	×	-	-	50.3	59.7	51.9	42.6	69.3	46.1	-	-	-
BART+BIFI (Yasunaga et al., 2021)	×	-	-	64.4	**35.6**	55.5	-	-	-	51.6	**24.7**	42.4
Ours+BIFI	✓	Gold+Pred	Pred	**68.61**	34.25	**57.14**	-	-	-	**54.44**	23.54	**43.12**
Ours	×	Gold+Pred	Pred	63.18	30.34	51.94	56.17	38.02	51.27	43.44	17.42	33.45
	✓	Gold+Pred	Pred	**64.36**	**31.24**	**53.10**	**57.42**	**38.48**	**52.27**	**44.04**	**17.61**	**33.87**

Table 4: Results on the English CoNLL-2014 test, BEA-2019 test, and BEA-2019 dev set. Scores that are higher than competitive baselines are highlighted in bold.

BART is to initialize the model with the BART and train the model on synthetic data as Awasthi et al. (2019).

BART+BIFI (Yasunaga et al., 2021) is an unsupervised synthetic data generation method using Break-It-Fix-It(BIFI) framework. It adopts a pre-trained language model to generate high-quality synthetic data which is used for training a BART model. It utilizes external data which is referred to as "BIFI" in Table 4. It is the state-of-the-art unsupervised English GEC model.

GPT-2 Alikaniotis and Raheja (2019) substituted the n-gram model with GPT-2 (Radford et al., 2019) and assessed its performance on GEC without any supervised training.

ChatGPT is powerful and achieves high performance across various Natural Language Processing tasks. Coyne et al. (2023) provide a zero-shot prompt for the GEC task on GPT-3.5.

4.4 Results Analysis

The results on the Chinese datasets are listed in Table 3. Our InstructGEC outperforms GECToR with a significant improvement of 2.23 $F_{0.5}$ on the NLPCC2018 test and 2.33 $F_{0.5}$ on the MuCGEC test. Additionally, Compared with BART, InstructGEC raises the $F_{0.5}$ by 3.7 on the NLPCC2018 test and by 1.6 on the MuCGEC test. "Transformer" gets the lowest $F_{0.5}$, and the possible reason is that it is not initialized by parameters of the pre-trained language model. Our method with the proposed masking strategy is better than that without it, and the masking strategy can significantly increase $F_{0.5}$ by 1.8 on the NLPCC2018 test and by 0.72 on the MuCGEC test.

Table 4 shows the experimental results on the English datasets. In contrast to "Transformer" and "BART", InstructGEC achieves the best performance. Specifically, InstructGEC with the masking strategy is better than "BART" with the improvement of the $F_{0.5}$ by 3.42, 3.54, and 2.69 on the CoNLL-2014 test, BEA-2019 test, and BEA-2019 dev set respectively. For a fair comparison, when our method with BIFI (i.e., Ours+BIFI) is trained on the same data as "BART+BIFI", our method outperforms "BART+BIFI" with +1.64, +0.72 $F_{0.5}$

Methods	Mask	Instruction Type		CoNLL-2014 test			BEA-2019 dev			BEA-2019 test		
		Train	Test	P	R	$F_{0.5}$	P	R	$F_{0.5}$	P	R	$F_{0.5}$
Instruction effect												
BART	×	-	-	61.33	28.23	49.68	41.07	15.88	31.18	53.68	35.59	48.73
	×	Pred	Pred	62.57	29.09	50.86	43.28	16.62	32.77	**55.75**	**37.13**	**50.67**
	×	Gold	Gold	**66.01**	**60.11**	**64.74**	**57.62**	**52.0**	**56.4**	-	-	-
JS divergence effect												
BART	×	Gold ∪ Pred	Pred	59.94	30.61	50.30	43.36	17.1	33.17	55.37	38.53	50.92
Ours with KL	×	Gold+Pred	Pred	64.03	28.55	51.28	**44.69**	16.18	33.04	56.51	35.95	50.71
Ours with KL	✓	Gold+Pred	Pred	**64.25**	29.89	52.24	43.62	17.0	33.22	56.28	36.87	50.92
Ours with JS	×	Gold+Pred	Pred	63.18	30.34	51.94	43.44	17.42	33.45	56.17	38.02	51.27
Ours with JS	✓	Gold+Pred	Pred	64.02	**31.69**	**53.18**	43.59	**17.72**	**33.74**	**56.51**	**38.89**	**51.81**
Mask Instruction effect												
Ours	×	Gold+Pred	Pred	63.18	30.34	51.94	43.44	17.42	33.45	56.17	38.02	51.27
	✓	Gold+Pred	Pred	**64.36**	**31.24**	**53.10**	**44.04**	**17.61**	**33.87**	**57.42**	**38.48**	**52.27**

Table 5: Ablation study with different setups. The mask ratio is 0.15 for the mask probability setting.

Language	Dataset	F_1
Chinese	NLPCC2018 test	28.43
	MuCGEC dev	20.04
English	CoNLL-2014 test	18.78
	BEA-2019 dev	28.95

Table 6: Results of instruction quality

improvement on CoNLL-2014 test and BEA-2019 dev set. The results demonstrate that InstructGEC can achieve better effectiveness than state-of-the-art baselines. The masking strategy has a consistent effect on English datasets.

For further analysis, we conduct experiments on Chinese and English test sets to investigate the quality of instructions generated by the sequence tagging model. The experimental results are shown in Table 6. On Chinese datasets, we achieve $F_1 = 28.43$ on the NLPCC2018 test set and $F_1 = 20.04$ on the MuCGEC dev set. On English datasets, we obtain $F_1 = 18.78$ on the CoNLL-2014 test set and $F_1 = 28.95$ on the BEA-2019 dev set. A lower F_1 score indicates poorer instruction quality. We can observe that the sequence tagging model trained on the unsupervised synthetic GEC data provides poor instructions. Thus, consistency loss is necessary to mitigate the effect of poor instructions. Additional error type analysis and comparison results for generated examples are provided in Appendix B.

5 Ablation Study

We perform ablation studies to explore the impact of various factors, including the instruction, JS divergence, masking strategy, and mask ratio, on the performance of models tested on the CoNLL-2014 test, the BEA-2019 dev, and the BEA-2019 test.

Effect of Instruction As shown in Table 5, BART is trained on the synthetic data without instruction tuning, and with the predicted and gold instructions respectively. Results with predicted instructions are slightly better than those without instruction tuning. It reveals that the predicted instructions are of poor quality and do not effectively enhance the performance of GEC models. This outcome is consistent with the results in Table 6. Results with gold instructions demonstrate the performance upper bound of instruction tuning. Specifically, if the gold instructions on the test set are given, we achieve 64.74 $F_{0.5}$ score on the CoNLL-2014 test set and 56.4 $F_{0.5}$ score on the BEA-2019 dev set, which are competitive with the results of supervised GEC methods.

Effect of JS divergence Because the low-quality synthetic data results in poor instructions, consistency loss is proposed to alleviate this issue. Jensen–Shannon (JS) and Kullback–Leibler (KL) divergence can be used to measure the dissimilarity between two probability distributions in statistics. The consistency loss based on KL divergence is defined as:

$$\begin{aligned}\mathcal{L}'_c = \frac{1}{2}[&KL(P(Y|P'_p;\theta)||P(Y|P'_t;\theta)) \\ &+ KL(P(Y|P'_t;\theta)||P(Y|P'_p;\theta))]\end{aligned} \tag{8}$$

As shown in Table 5, when the JS divergence is changed to KL divergence and the masking strategy is not used, $F_{0.5}$ drops by 0.66 on CoNLL-2014 test set, by 0.41 on BEA-2019 dev set, and by 0.56 on on BEA-2019 test set. When JS is changed to KL and the masking strategy is used, $F_{0.5}$ decreases

Mask Ratio $m\%$	CoNLL-2014 test			BEA-2019 dev			BEA-2019 test		
	P	R	$F_{0.5}$	P	R	$F_{0.5}$	P	R	$F_{0.5}$
0%	63.18	30.34	51.94	43.44	17.42	33.45	56.17	38.02	51.27
15%	64.02	31.69	53.18	43.59	17.72	33.74	56.51	38.89	51.81
20%	64.36	31.24	53.10	**44.04**	17.61	33.87	**57.42**	38.48	**52.27**
25%	**64.42**	31.65	**53.37**	43.48	17.91	33.82	56.50	38.59	51.70
30%	62.92	30.55	51.92	43.83	17.93	**34.00**	56.91	38.77	52.04
40%	63.11	**31.87**	52.77	43.19	**18.36**	33.99	55.74	**39.35**	51.45
50%	63.03	31.32	52.42	42.79	18.22	33.7	55.39	38.88	51.06

Table 7: The results of different mask ratios $m\%$.

by 0.94 on CoNLL-2014 test set, by 0.52 on BEA-2019 dev set, and by 0.89 on BEA-2019 test set.

It is unfair to compare our proposed InstructGEC with BART since BART does not use instructions and the consistency loss. We combine (P_p', Y) and (P_t', Y) to build a larger training data (i.e., Gold $\cup$ Pred) which is used for training a BART model with instruction tuning. If the masking strategy is not used, both "Ours with JS" and "Ours with KL" outperform BART trained on Gold $\cup$ Pred. Therefore, JS divergence is better for enhancing the unsupervised GEC models.

Effect of Masking Strategy We conduct experiments with and without the masked prompt to validate the effectiveness of the mask strategy. As shown in Table 5, the performance of InstructGEC without the masked prompt declines by 1.16 $F_{0.5}$ on CoNLL-2014 test set, 0.45 $F_{0.5}$ on BEA-2019 dev set, and 1.0 $F_{0.5}$ on BEA-2019 test set. The mask strategy is able to alleviate the impact of the copying phenomenon in the task of unsupervised GEC. Our method learns not only to correct the grammatical errors but also to predict the masked tokens based on contextual language using the masking strategy. This allows our model to enhance the ability of semantic understanding based on the context and to correct grammatical errors accurately.

Effect of Mask Ratio We test different values of mask ratio $m\%$ for InstructGEC. After the gold and predicted instructions are masked with a specified mask ratio, they are used to construct prompts for training InstructGEC. Predicted instructions are not masked in the inference stage.

The experimental results are shown in the Table 7. We observe that our method is robust and insensitive to the mask ratio $m\%$. The possible reason is that a large amount of synthetic data alleviates the impact of the mask ratio. The superior performance is achieved when the $m\%$ is between 15% and 25%. If the $m\%$ is more than 25% or less than 15%, the performance drops. The lower value of $m\%$ reduces the number of masked tokens, which is harmful to GEC models for learning the semantic knowledge of contextual language. The higher value might make the masked sentence lack semantic coherence and hinder effective training.

6 Conclusion

In this paper, we design an instruction format and use the masking strategy in both an erroneous sentence and the corresponding instructions, enabling our method to alleviate the impact of the copying phenomenon and learn rich semantic knowledge on large-scale synthetic data. We propose a novel approach, InstructGEC, which incorporates GED knowledge into GEC models with instruction tuning for unsupervised GEC. InstructGEC can mitigate the low-quality issue to improve the generalization ability of GEC models. Experiments are conducted to validate our proposed method on English and Chinese GEC datasets. Experimental results demonstrate that InstructGEC can achieve state-of-the-art performance. In addition, the results of ablation studies show that the masking strategy and the consistency loss based on JS divergence are effective in achieving superior performance. Our method is not sensitive to the mask ratio.

In the future, because the generation of high-quality instructions in the GED stage is challenging, we will explore more effective methods for generating high-quality instructions or use multi-task learning to enhance GEC models' capability.

7 Limitations

Our model exhibits a limitation during the instruction generation phase, leading to suboptimal instruction quality. To address the discrepancy between actual and predicted instructions, we use a Jensen–Shannon divergence in the loss function.

Accurate token classification remains a challenge task. Additionally, while the masking strategy effectively reduces duplication and enhances semantic understanding for grammar correction, our model still lacks interpretability.

8 Ethical Statements

In this paper, we introduce InstructGEC which combines GED knowledge with instructions and uses a masked strategy in GEC. All data are from publicly available sources, with no sensitive information involved. GEC is widely studied and applied, but unsupervised GEC has been underexplored. Our work aims to advance unsupervised GEC methods and broaden their potential applications.

Acknowledgments

This work was partially supported by the National Natural Science Foundation of China (No. 62372252, 62302245, 62172237, 62077031, 62176028), Ministry of Education of the People's Republic of China Humanities and Social Sciences Youth Foundation (No. 23YJCZH240), the NSFC-General Technology Joint Fund for Basic Research (No. U1936105,U1936206). We thank the AC, SPC, PC and reviewers for their insightful comments on this paper.

References

Dimitris Alikaniotis and Vipul Raheja. 2019. The unreasonable effectiveness of transformer language models in grammatical error correction. In *Proc. of the Workshop on Innovative Use of NLP for Building Educational Applications*, pages 127–133.

Abhijeet Awasthi, Sunita Sarawagi, Rasna Goyal, Sabyasachi Ghosh, and Vihari Piratla. 2019. Parallel iterative edit models for local sequence transduction. In *Proc. of EMNLP-IJCNLP*, pages 4260–4270.

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

Christopher Bryant, Mariano Felice, and Ted Briscoe. 2017. Automatic annotation and evaluation of error types for grammatical error correction. In *Proc. of ACL*, pages 793–805.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*.

Chen Chen, Bo He, Jing Yuan, Chunyan Hou, and Xiaojie Yuan. 2023. Incorporating constituent syntax into grammatical error correction with multi-task learning. In *Proc. of CIKM*, page 286–295.

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 *Proc. of EMNLP*, pages 7162–7169.

Steven Coyne, Keisuke Sakaguchi, Diana Galvan-Sosa, Michael Zock, and Kentaro Inui. 2023. Analyzing the performance of gpt-3.5 and gpt-4 in grammatical error correction. *arXiv preprint arXiv:2303.14342*.

Daniel Dahlmeier and Hwee Tou Ng. 2012. Better evaluation for grammatical error correction. In *Proc. of NAACL*, pages 568–572.

Biug Draplater, Buptcjj. 2018. pkunlp: A python tool for data segmentations. Software available from https://github.com/biug/pkunlp.

Roman Grundkiewicz, Marcin Junczys-Dowmunt, and Kenneth Heafield. 2019. Neural grammatical error correction systems with unsupervised pre-training on synthetic data. In *Proc. of the Workshop on Innovative Use of NLP for Building Educational Applications*, pages 252–263.

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 *Proc. of ACL*, pages 4248–4254.

Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proc. of NAACL-HLT*, volume 1, page 2.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Shun Kiyono, Jun Suzuki, Masato Mita, Tomoya Mizumoto, and Kentaro Inui. 2019. An empirical study of incorporating pseudo data into grammatical error correction. In *Proc. of EMNLP-IJCNLP*, pages 1236–1242.

VI Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. *Proc. of the Soviet Physics Doklady*.

M Lewis. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Yinghao Li, Xuebo Liu, Shuo Wang, Peiyuan Gong, Derek F Wong, Yang Gao, He-Yan Huang, and Min Zhang. 2023. Templategec: Improving grammatical error correction with detection template. In *Proc. of ACL*, pages 6878–6892.

Jared Lichtarge, Chris Alberti, Shankar Kumar, Noam Shazeer, Niki Parmar, and Simon Tong. 2019. Corpora generation for grammatical error correction. In *Proc. of NAACL*, pages 3291–3301.

Ilya Loshchilov and Frank Hutter. 2018. Fixing weight decay regularization in adam. *arXiv preprint arXiv:1711.05101*.

Eric Malmi, Sebastian Krause, Sascha Rothe, Daniil Mirylenka, and Aliaksei Severyn. 2019. Encode, tag, realize: High-precision text editing. In *Proc. of EMNLP-IJCNLP*, pages 5054–5065.

Hiroki Nakayama. 2018. seqeval: A python framework for sequence labeling evaluation. Software available from https://github.com/chakki-works/seqeval.

Kostiantyn Omelianchuk, Vitaliy Atrasevych, Artem Chernodub, and Oleksandr Skurzhanskyi. 2020. Gector–grammatical error correction: Tag, not rewrite. In *Proc. of the Workshop on Innovative Use of NLP for Building Educational Applications*, pages 163–170.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9.

Sascha Rothe, Jonathan Mallinson, Eric Malmi, et al. 2021. A simple recipe for multilingual grammatical error correction. In *Proc. of ACL and IJCNLP*, pages 702–707.

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 arXiv:2109.05729*.

Kai Shen, Yichong Leng, Xu Tan, Siliang Tang, Yuan Zhang, Wenjie Liu, and Edward Lin. 2022. Mask the correct tokens: An embarrassingly simple approach for error correction. In *Proc. of EMNLP*, pages 10367–10380.

Felix Stahlberg and Shankar Kumar. 2021. Synthetic data generation for grammatical error correction with tagged corruption models. In *Proc.of the Workshop on Innovative Use of NLP for Building Educational Applications*, pages 37–47.

Xin Sun, Tao Ge, Shuming Ma, Jingjing Li, Furu Wei, and Houfeng Wang. 2022. A unified strategy for multilingual grammatical error correction with pre-trained cross-lingual language model. In *Proc. of IJCAI*, pages 4367–4374.

Chenming Tang, Xiuyu Wu, and Yunfang Wu. 2023. Are pre-trained language models useful for model ensemble in chinese grammatical error correction? In *Proc. of ACL*, pages 893–901.

Ashish Vaswani. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762*.

Xiao Wang, Weikang Zhou, Can Zu, Han Xia, Tianze Chen, Yuansen Zhang, Rui Zheng, Junjie Ye, Qi Zhang, Tao Gui, Jihua Kang, Jingsheng Yang, Siyuan Li, and Chunsai Du. 2023. Instructuie: Multi-task instruction tuning for unified information extraction. *arXiv preprint arXiv:2304.08085*.

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. Finetuned language models are zero-shot learners. In *Proc. of ICLR*.

Xiuyu Wu, Jingsong Yu, Xu Sun, and Yunfang Wu. 2022. Position offset label prediction for grammatical error correction. In *Proc. of COLING*, pages 5409–5418.

Lingyu Yang, Hongjia Li, Lei Li, Chengyin Xu, Shutao Xia, and Chun Yuan. 2023. Let: Leveraging error type information for grammatical error correction. In *Findings of ACL*, pages 5986–5998.

Michihiro Yasunaga, Jure Leskovec, and Percy Liang. 2021. Lm-critic: Language models for unsupervised grammatical error correction. In *Proc. of EMNLP*, pages 7752–7763.

Zheng Yuan, Shiva Taslimipoor, Christopher Davis, and Christopher Bryant. 2021. Multi-class grammatical error detection for correction: A tale of two systems. In *Proc. of EMNLP*, pages 8722–8736.

Boyu Zhang, Hongyang Yang, and Xiao-Yang Liu. 2023. Instruct-fingpt: Financial sentiment analysis by instruction tuning of general-purpose large language models. *arXiv preprint arXiv:2306.12659*.

Yue Zhang, Zhenghua Li, Zuyi Bao, Jiacheng Li, Bo Zhang, Chen Li, Fei Huang, and Min Zhang. 2022. Mucgec: A multi-reference multi-source evaluation dataset for chinese grammatical error correction. In *Proc. of NAACL*, pages 3118–3130.

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 *Proc. of NAACL*, pages 156–165.

Yuanyuan Zhao, Nan Jiang, Weiwei Sun, and Xiaojun Wan. 2018. Overview of the nlpcc 2018 shared task: Grammatical error correction. In *Proc. of NLPCC*, pages 439–445.

A Additional details of InstructGEC

In this section, we discuss the details of training and inference for InstructGEC.

A.1 InstructGEC Implementation Details

In the detection stage, the sequence tagging model is implemented with the BERT model base version on HuggingFace. We use the dropout strategy on all encoders with a probability of 0.1, and the AdamW (Loshchilov and Hutter, 2018) optimizer with a learning rate of 2e-5, a beta value of (0.9,0.998), and a weight decay value of 0.01. The linear learning rate schedule is adopted with 2000-step warmup steps. We use a cross-entropy loss function and an early-stop technique to select the best model on the validation set for the sequence tagging model.

In the correction stage, the Transformer-based Seq2Seq model is based on the encoder-decoder architecture. Chinese BART-large[4] and Bart-base[5] are used to initialize parameters of Chinese and English Seq2Seq model respectively. The hyper-parameter λ is set to 1 while the learning rate is to 3e-5. It is not fair to compare our InstructGEC with "BART+BIFI" since "BART+BIFI" uses more synthetic data than InstructGEC. We also build a model (i.e., "Ours+BIFI") that uses the same amount of synthetic data as "BART+BIFI". Note that "Ours+BIFI" adopts the masking strategy and "BART+BIFI" does not. Our experiments were trained and evaluated on a single 24GB Nvidia RTX 3090 GPU.

A.2 Baseline Settings

For the Chinese baseline model, we follow the setup of Zhang et al. (2022) for the GECToR and BART model. We used the original Transformer architecture as the baseline, which includes 6 encoders and 6 decoders. The specific settings are shown in the Table 8.

For the English baseline model, we follow the work of Yasunaga et al. (2021), using bart-base as the baseline under the unsupervised setting. The Transformer baseline consists of 6 encoders and 6 decoders. The specific settings are shown in the Table 9.

[4] https://huggingface.co/fnlp/bart-large-chinese/tree/v1.0

[5] https://huggingface.co/facebook/bart-base

Algorithm 1 Training Procedure of InstructGEC

Input: The synthetic dataset $SD = (X, Y)$, the mask probability $m\%$, the number of epochs $epoch$

Output: The sequence tagging model, the Transformer-based Seq2Seq model

1: Generate the gold instruction I_t for X using the sequence matcher
2: Initialize Sequence Tagging Model (STM) with BERT Base and train STM on (X, I_t)
3: Predict the instruction I_p for X by STM
4: Initialize the Transformer-based Seq2Seq model with BART Model
5: **for** $i = 1$ to $epoch$ **do**
6: $X' \leftarrow \varnothing$, $I'_p \leftarrow \varnothing$, $I'_t \leftarrow \varnothing$, $P'_p \leftarrow \varnothing$, $P'_t \leftarrow \varnothing$
7: Convert (X, I_p) and (X, I_t) into (X'_p, I'_p) and $(X'_t, , I'_t)$ using the proposed masking strategy respectively with the mask probability $m\%$
8: Construct the prompt P'_t and P'_p respectively by Equation 2 and 3
9: Train the Seq2Seq model on (P'_p, P'_t, Y).
10: **end for**

A.3 Training and Inference

Although our method is associated with two stages (i.e., GED and GEC), we can train our proposed InstructGEC model by optimizing Equation 7. The overview of the training procedure is summarized in Algorithm 1. The GED stage is described in lines 1 to 3. The gold instruction I_t for erroneous sentences X is generated by the sequence matcher. The parameters of the Sequence Tagging Model (STM) are initialized by the BERT-based (Kenton and Toutanova, 2019) model and trained on (X, I_t). STM is used to output the predicted instruction I_p for X. The GEC stage is introduced in the line 4 to 9. The parameters of the Transformer-based Seq2Seq model are initialized with the BART model. Specifically, the Transformer-based Chinese Seq2Seq model is initialized with the Chinese BART large version (Shao et al., 2021), while the English Seq2Seq model is initialized with the BART Base version (Lewis, 2019). Finally, the sequence tagging model and the Transformer-based Seq2Seq model are returned after the training procedure.

Settings	Transformer	Bart	Gector
Pretrained model	-	Chinese-BART-Large	Chinese-Struct-Bert-Large
Number of max epochs	30	10	10
Device	1 Nvidia RTX 3090 GPU		
Optimizer	Adam (Kingma and Ba, 2014) ($\beta_1 = 0.9, \beta_2 = 0.98, \epsilon = 1 \times 10^{-6}$)		
Cold learning rate	-	-	1×10^{-3}
Learning rate	5×10^{-4}	1×10^{-5}	1×10^{-5}
Max Tokens	4096	2048	-
Warmup	2000	2000	-
Loss Function	label smoothed cross entropy		cross entropy
Dropout	0.1	0.1	-
Stopping criteria Label	prediction accuracy on the dev set		
Patience	3	3	3

Table 8: Chinese Baseline Settings.

Settings	Transformer	Bart
Pretrained model	-	BART-base
Number of max epochs	30	5
Device	1 Nvidia RTX 3090 GPU	
Optimizer	Adam (Kingma and Ba, 2014) ($\beta_1 = 0.9, \beta_2 = 0.98, \epsilon = 1 \times 10^{-6}$)	
Learning rate	5×10^{-4}	5×10^{-5}
Max Tokens	8192	8192
Warmup	4000	4000
Loss Function	label smoothed cross entropy ($label - smoothing = 0.1$)	
Dropout	0.1	0.1
Stopping criteria Label	prediction accuracy on the dev set	
Patience	3	3

Table 9: English Baseline Settings.

B Case Study

B.1 Error type analysis

To demonstrate that our model has a better understanding of sentence context and semantics for grammatical error correction, we compare the performance of our model with baseline models across different error types. The error type analysis is conducted using the Errant[6] tool for English and the ChErrant (Zhang et al., 2022) tool for Chinese. ChErrant and Errant can automatically assign error types based on source erroneous sentences and target corrected sentences. The error type analysis for English is shown in Table 12, and the error type analysis for Chinese is listed in Table 11. $F_{0.5}$ is used as the evaluation metric for each error type.

B.2 Generation Comparison

Some examples are listed in Table 10. Our model can generate the correct target sentence, whereas the baseline Bart model cannot. In the first example, even though the predicted instruction type is incorrect, the grammatical error correction model can still correct the mistake based on the hint provided. The possible reason is that the dynamic masking strategy can enhance the model's semantic understanding. As a result, "secret" is correctly modified to "secrets," whereas the Bart baseline model fails to make this correction. In the second example, although the predicted instruction does not identify all the grammatical errors in the sentence, it correctly indicates that "vertically" is needed to be replaced with "immediately." In the third example, despite the error in the instruction, the model still correctly modify the sentence by changing "said" to "say" and does not change other tokens. This is due to the introduction of the dynamic masking strategy, which enhanced the model's semantic understanding.

[6] https://github.com/chrisjbryant/errant

Type	Examples
Source	Above all, life is more important than secret.
InstructGEC Correction Stage Input	<k> <k> <k> <k> <k> <k> <k> <k> <a> <k> [SEP] ĠAbove Ġall , Ġlife Ġis Ġmore Ġimportant Ġthan Ġsecret Ġ.
InstructGEC Output	Above all, life is more important than secrets.
Baseline Output	Above all, life is more important than secret.
Source	The notion of authority also extended 'vertically' .
InstructGEC Correction Stage Input	<k> <k> <k> <k> <k> <k> <k> <k> <r> <r> <k> [SEP] ĠThe Ġnotion Ġof Ġauthority Ġalso Ġextended 've rt ically ' Ġ.
InstructGEC Output	The notion of authority also extended immediately.
Baseline Output	The notion of authority also extends vertically.
Source	Some said that the genetic risk that is found in a person should be kept secret because it is considered as a personal information which should be kept confidential .
InstructGEC Correction Stage Input	<k> <r> <k> <k> <k> <k> <k> <k> <k> <k> <k> <k> <k> <k> <k> <k> <k> <k> <k> <k> <k> <d> <k> <k> <k> <k> <k> <k> <r> <d> [SEP] ĠSome Ġsaid Ġthat Ġthe Ġgenetic Ġrisk Ġthat Ġis Ġfound Ġin Ġa Ġperson Ġshould Ġbe Ġkept Ġsecret Ġbecause Ġit Ġis Ġconsidered Ġas Ġa Ġpersonal Ġinformation Ġwhich Ġshould Ġbe Ġkept Ġconfidential Ġ.
InstructGEC Output	Some say that the genetic risk that is found in a person should be kept secret because it is considered as a personal information which should be kept confidential.
Baseline Output	Some said that the genetic risk that is found in a person should be kept secret because it is considered as a personal information which should be kept confidential.

Table 10: Examples Comparison. "Baseline" refers to the Bart model.

Error Type	InstructGEC	Bart
M	**31.14**	24.92
R	**24.32**	20.63
S	**27.44**	25.71
W	**34.63**	32.93
ADJ	**27.19**	16.9
ADV	**21.95**	17.01
AUX	**37.07**	37.01
CONJ	**27.21**	18.42
NOUN	**24.05**	15.76
NUM	**31.58**	25.18
OTHER	**9.23**	8.33
PREP	**35.65**	30.76
PRON	**19.46**	11.8
PUNCT	11.48	**14.15**
QUAN	**13.89**	6.94
SPELL	**70.62**	69.02
VERB	**28.28**	24.38

Table 11: The performance of our model and the baseline for specific error types in Chinese.

Error Type	InstructGEC	Bart
M	**49.32**	46.86
R	**54.30**	51.58
U	**49.70**	41.68
ADJ	**30.53**	23.81
ADJ:FORM	62.50	**75.00**
ADV	**35.03**	32.70
CONJ	28.07	**31.03**
CONTR	45.45	**62.50**
DET	**65.32**	59.92
MORPH	**60.44**	56.72
NOUN	28.30	**28.35**
NOUN:INFL	63.83	63.83
NOUN:NUM	**72.60**	68.24
NOUN:POSS	59.26	**59.44**
ORTH	49.41	**53.48**
OTHER	**25.26**	24.86
PART	63.49	**70.65**
PREP	**66.22**	61.63
PRON	**64.97**	62.83
PUNCT	**32.55**	30.91
SPELL	**59.09**	51.84
VERB	**33.52**	31.47
VERB:FORM	**73.81**	73.11
VERB:INFL	**97.22**	86.21
VERB:SVA	85.85	**86.26**
VERB:TENSE	**58.23**	48.66
WO	18.35	**20.62**

Table 12: The performance of our model and the baseline for specific error types in English.