Improving Grammatical Error Correction by Correction Acceptability Discrimination

Bin Cao[1] , Kai Jiang[1] , Fayu Pan[2] , Chenlei Bao[1], Jing Fan[1]*
[1]Zhejiang University of Technology, [2]Xihu Xinchen Hangzhou Technology Co., Ltd
Hangzhou, China
{bincao, jiangkai, baochenlei, fanjing}@zjut.edu.cn, AnticPan@outlook.com

Abstract

Existing Grammatical Error Correction (GEC) methods often overlook the assessment of sentence-level syntax and semantics in the corrected sentence. This oversight results in final corrections that may not be acceptable in the context of the original sentence. In this paper, to improve the performance of Grammatical Error Correction methods, we propose the post-processing task of Correction Acceptability Discrimination (CAD) which aims to remove invalid corrections by comparing the source sentence and its corrected version from the perspective of "sentence-level correctness". To solve the CAD task, we propose a pipeline method where the acceptability of each possible correction combination based on the predicted corrections for a source sentence will be judged by a discriminator. Within the discriminator, we design a symmetrical comparison operator to overcome the conflicting results that might be caused by the sentence concatenation order. Experiments show that our method can averagely improve $F_{0.5}$ score by 1.01% over 13 GEC systems in the BEA-2019 test set.

Keywords: Grammatical Error Correction, Correction Acceptability Discrimination, Invalid Corrections

1. Introduction

Grammatical Error Correction (GEC) is the task to correct different types of errors in written text, such as misspelled words, word choice errors, and grammatical errors. In recent years, many works (Bryant et al., 2019; Grundkiewicz et al., 2019; Omelianchuk et al., 2020; Tarnavskyi et al., 2022) show that using larger pre-trained language models with bigger training datasets can help get better performance for GEC task. For example, the T5 XXL model which has 11B parameters is used in the current state-of-the-art (SOTA) seq2seq-based GEC system (Rothe et al., 2021).

To further improve the performance, the ensembling strategy has also been explored. For example, some works combine multiple GEC systems' corrections and use the optimization method like nonlinear integer programming to get an optimal corrections combination (Kantor et al., 2019; Lin and Ng, 2021; Qorib et al., 2022; Tarnavskyi et al., 2022). Particularly, Tarnavskyi et al. (2022) achieves the SOTA result (76.05 of $F_{0.5}$ score) for ensembles on the BEA-2019 benchmark by simply using the majority votes over the predicted taggers for output edit spans. However, the above ensemble based methods work with more than one base GEC model and require even more time and resources for training, which makes them hard to be deployed in some low-resource environments. Moreover, existing GEC methods, whether ensembled or not, neglect to judge the correctness of sentence-level syntax and semantics for the corrected sentence, causing the final obtained corrections unacceptable to the source sentence.

In this paper, to reject the invalid corrections outputted by the GEC system, we propose a new post-processing task named Correction Acceptability Discrimination (CAD) where each correction combination from the output sentence of the existing GEC system will be applied to the source sentence and then compared with the source sentence from the perspective of "sentence-level correctness". The goal of the CAD task is to find the correction combination that has the best correctness. Though the grammatical acceptability is also included in the task of Linguistic Quality Evaluation (LQE) (Conroy and Dang, 2008; Zhao et al., 2019a; Zhu and Bhat, 2020; Daudaravicius et al., 2016), there is no comparison made between two sentences as required in our CAD task. That is, LQE methods merely consider the grammatical correctness for a single sentence, and when a GEC system mistakenly corrects the semantic information of a source sentence, the wrong corrections will not be found by LQE methods.

To solve the CAD task, a straightforward idea is to view CAD as a binary classification problem. Specifically, the source sentence S_{src} and its corrected version S_{corr} are firstly concatenated in an ordered pair $[S_{src}, S_{corr}]$, then use a classifier to predict 1 if $score(S_{src}) \leq score(S_{corr})$ otherwise 0. However, the concatenation order of two sentences may cause conflict in predicting results. For example, the prediction result based on $[S_{src}, S_{corr}]$ indicates S_{corr} is better than S_{src}, but when different concatenation order for the same pair of sentences

*Corresponding author

LREC-COLING 2024, pages 8818–8827

is used as the input, i.e., $[S_{corr}, S_{src}]$, the predicted probability may still be higher than 0.5 implying S_{src} is more correct than S_{corr}.

Therefore, to improve the robustness against different concatenation order for CAD task, we design a sentence pair correctness discriminator which is insensitive to the concatenation order when comparing the correctness of two sentences. Particularly, within the discriminator, we propose a symmetrical comparison operator to fuse two sentences' embeddings and apply a score function to output the correctness score for each sentence. Moreover, based on the discriminator, we further propose a pipeline method to improve the performance of existing GEC methods by removing the wrong corrections from their output results. In the pipeline, we first construct a set of sentence pairs by applying different correction combinations to the source sentence. Then each sentence pair will be fed into the discriminator and two correctness scores will be predicted. Finally, we select the correction combination that has the highest relative correctness score as the final result.

Our contributions can be summarized as follows:

- To compare and evaluate a pair of sentences from the perspective of sentence-level correctness, we propose a new task named Correction Acceptability Discrimination (CAD).
- To address the CAD task, we design a sentence pair correctness discriminator which can generate correctness score for each sentence. Users can judge which sentence is more correct by comparing their scores.
- We also design a pipeline method based on the discriminator to improve the performance of existing GEC systems.
- Experiments show our discriminator can achieve 94% accuracy for correctness comparison of sentence pairs. Then we apply our discriminator to further check the correction results of 13 GEC systems. The results show that our pipeline method can averagely improve $F_{0.5}$ score by 0.89% to each system in the BEA-2019 test set.

2. CAD Task

Given a source sentence S_{src} and its corrected version S_{pre} predicted by the existing GEC system, the predicted corrections $C_{pre} = \{c_1, ..., c_n\}$ can be extracted by comparing S_{src} with S_{pre}. Similarly, we can get ground truth corrections C_{gold} by comparing S_{src} with the target sentence S_{gold}. Each correction can be formulated as a tuple $(Start\_pos, End\_pos, Correct\_tokens)$ where the start position $Start\_pos$ and the end position End_pos is the error span detected in S_{src}, and $Correct\_tokens$ is the corresponding correction appeared in the output sentence S_{pre}. The correction is valid if it is in C_{gold}. Since C_{pre} may contain some invalid corrections, the task of correction acceptability discrimination aims to find the intersection set C_{opt} of C_{gold} and C_{pre}, i.e., remove corrections that are not in the C_{gold} from the C_{pre}.

S_{src}	Would do you please join us ?
S_{gold}	Would you like to join us ?
S_{pre}	Would you please joining us ?
S_{opt}	Would you please join us ?
C_{gold}	(1, 2, ''), (3, 4, 'like to')
C_{pre}	(1, 2, ''), (4, 5, 'joining')
C_{opt}	(1, 2, '')

Table 1: An example of CAD task. The $F_{0.5}$ score of S_{pre} with C_{pre} is 0.5 while S_{opt} with C_{opt} is 0.83.

Example. As illustrated in Table 1, by comparing S_{src} with S_{gold}, there are two ground truth corrections in C_{gold}: c_1 = (1, 2, '') and c_2 = (3, 4, 'like to'). However, the predicted sentence S_{pre} has an invalid correction over S_{src}, i.e., (4, 5, 'joining') in C_{pre}. After performing the CAD task, the subset of only valid corrections C_{opt} will be found and S_{opt} can be derived by applying C_{opt} over S_{src}.

3. Our Proposed Method

3.1. Overview

The key to solve the CAD task is to discriminate whether each correction is acceptable to the source sentence, hence we propose a 3-step pipeline to reject invalid corrections from the output sentence of the existing GEC system. The main workflow is shown in Figure 1:

Step 1: Sentence Pair Construction. The main purpose of this step is to build the input for correctness acceptability comparison in Step 2. To this end, we first extract all predicted corrections $C_{pre} = \{c_1, ..., c_n\}$ with ERRANT toolkit (Bryant et al., 2017) by comparing the source sentence S_{src} with the sentence S_{pre} output by the existing GEC system. Next, we further combine different corrections that n corrections can make 2^n sentence pairs and apply each combination C_{corr} ($C_{corr} \subset C_{pre}$) to S_{src} to form its corrected version S_{corr}. After that, a set of sentence pairs can be constructed by grouping each S_{corr} and S_{src}.

Step 2: Correctness Discrimination. As the core of the whole pipeline, this step uses a discriminator model to return the correctness comparison results for each sentence pair from Step 1. Within the discriminator, we design a symmetrical scoring operator to produce correctness scores P_{src} and

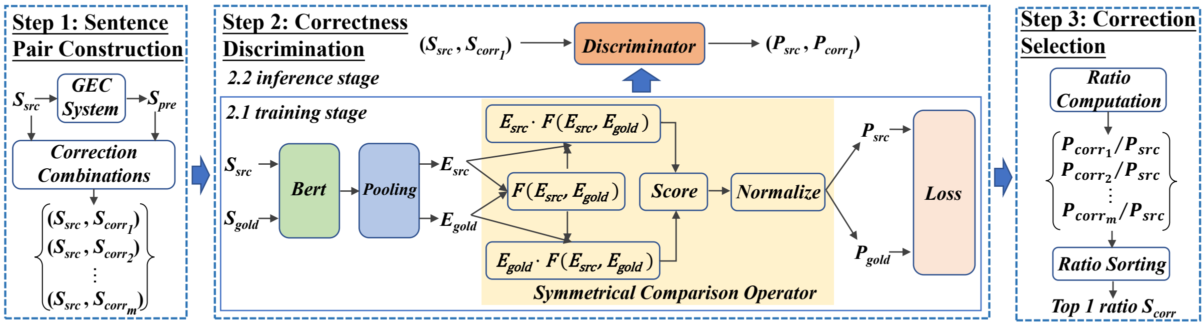

Figure 1: The framework of proposed pipeline method for CAD task

P_{corr} for S_{src} and S_{corr} respectively without considering their concatenation order. By comparing P_{src} and P_{corr}, we know which sentence is more correct. Note that, the discriminator should be trained with the existing GEC datasets in advance.

Step 3: Correction Selection. This step aims to select the best correction combination based on C_{pre} and generate the optimized version S_{opt} for S_{pre}. To get the best correction combination, we further calculate the relative correctness score P_{corr}/P_{src} for each sentence pair and select the highest one.

Next, we discuss the implementation details of the proposed discriminator model and the correction selection strategy in the following sections.

3.2. Discriminator Implementation

The discriminator is the main component of Step 2 and it will be iteratively invoked to compare which sentence is more correct for each sentence pair from Step 1. To this end, we propose to train a discriminator model with the set of sentence pairs (S_{src}, S_{gold}) from existing GEC datasets. First, we use BERT (Devlin et al., 2019) to encode two input sentences separately, thus we can get each token's hidden state in the sentence. Then following the previous work of Sentence-BERT (Reimers and Gurevych, 2019), we apply the mean pooling to all tokens' hidden states of a sentence and get two sentence embeddings E_{src} and E_{gold}.

After representing two sentences, a simple way to find which sentence is more correct is to concatenate two sentence embeddings and use a full connection layer for binary classification. However, as mentioned in the introduction, the performance of this idea cannot be guaranteed once the concatenation order is switched. Therefore, as highlighted with the yellow color background in Figure 1, we design a symmetrical score operator motivated by the work of (Reimers and Gurevych, 2019) where no concatenation is involved. The computation for the operator is defined as follows:

$$E_{fusion} = F(E_{src}, E_{gold}) \qquad (1)$$

$$P_i = Normalize(score(E_i, E_{fusion})) \qquad (2)$$

F is the fusion strategy as shown in Table 3, i.e., Hadamard product and add. E_{fusion} is a vector containing fused information of E_{src} and E_{gold}, and it plays the role of anchor to help discriminate between the error sentence E_{src} and the target sentence E_{gold}. Particularly, since we know in advance that the target sentence S_{gold} is more correct than the source sentence S_{src}, we define the score function $score$ as shown in Sec 4.5 to scale the output into a probability. Then we further calculate the correctness score P_i for a sentence S_i in Equation 2, where we use softmax or sigmoid function for normalization. Hence, P_i is in the range of $[0, 1]$ and the larger P_i denotes the higher correctness acceptability.

The training goal of the discriminator is to maximize the difference between the S_{src} and S_{gold} in terms of the correctness acceptability. Hence, we define the loss function as follows:

$$loss = -\log(\frac{P_{gold} - P_{src} + 1}{2}) \times \frac{1}{1 - wer} \qquad (3)$$

where two parts are involved: (1) $\frac{P_{gold}-P_{src}+1}{2}$ denotes the correctness difference based on the correctness scores obtained by Equation 2; (2)Similar to the edit distance(Ristad and Yianilos, 1998), $\frac{1}{1-wer}$ can be viewed as an inflation factor which makes low-correctness sentence with large word error rate wer (Klakow and Peters, 2002) farther away from the high-correctness sentence. wer is calculated by the following equation:

$$wer = \frac{r + d + a}{l_{src} + l_{gold}} \qquad (4)$$

where r, d and a represent the number of words substitution, deletion, and insertion by S_{gold} to S_{src}, respectively. l_{src}, l_{gold} are the lengths of S_{src} and S_{gold}. Dividing by the sum of the lengths of two sentences is to limit the wer to be between 0 and 1.

Based on the above training mechanism, our discriminator is able to effectively discriminate each sentence pair (S_{src}, S_{corr}) from Step 1 by comparing their correctness scores, and a larger difference indicates more correctness of one sentence against the other. But it is also important to note that for the same source sentence S_{src} in different sentence pairs, it may have different correctness scores. Hence, to find the best correction combination C_{opt}, directly choosing the highest P_{corr} among different sentence pairs will not work. We present our correction selection strategy in the following section.

3.3. Correction Selection Strategy

To get the best correction combination, we can refer to the relative correctness score which can be defined either by $(P_{corr} - P_{src})$ or $\frac{P_{corr}}{P_{src}}$. Since two scores are derived by softmax function, we have $P_{corr} + P_{src} = 1$ and using either above definition both work theoretically. However, in practical implementations, due to the precision of floating-point number specification (IEEE 754) (Kahan, 1996), the results of $(P_{corr} - P_{src})$ might be the same and some correction combinations cannot be differentiated. Hence, we finally adopt the ratio $\frac{P_{corr}}{P_{src}}$ as the computation for the relative correctness score for each sentence pair.

As a result, the final sentence S_{opt} for S_{src} will be found by selecting the highest relative correctness score among all sentence pairs.

In addition, considering that n corrections can make 2^n sentence pairs in Step 1, the calculation cost is unacceptable when n is large. So in our selection strategy, we manually set a threshold $T(T < n)$. Specifically, we first sort the correction combinations in decreasing order based on the correction number each C_{corr} has, then select correction combinations that contain more than n' corrections, where n' is defined as the minimal value that satisfies $\sum_{i=n'}^{n} \binom{n}{i} \leq 2^T - 1$. We will later discuss the empirical determination of T in our experiments.

4. Experiments

4.1. Datasets

According to the pipeline shown in Figure 1, we divide existing datasets to the following four groups and summarize them in Table 2.

Training Datasets for Discriminator. Following the datasets setting in BEA-2019 shared task for GEC (Bryant et al., 2019), we use the following datasets for training our discriminator: FCE (Yannakoudakis et al., 2011), NUCLE (Dahlmeier et al., 2013), LANG-8 (Mizumoto et al., 2011), and W&I+LOCNESS (Bryant et al., 2019) Corpus.

Validation and Test Datasets for Discriminator. We split W&I validation set into two parts, one with FCE's validation set for discriminator validation, and the other with FCE's test set for discriminator evaluation.

Corpus	Sentences	Tok.	Corr.
Training Sets for Discriminator			
FCE (train)	17,715	19.6	2.43
LANG-8	498,359	13.5	2.38
NUCLE	21,354	26.0	2.03
W&I	22,737	21.3	2.73
Validation Sets for Discriminator			
FCE (dev)	1,370	19.7	2.50
W&I (dev P1)	1,408	23.6	2.74
Test Sets for Discriminator			
FCE (test)	1,792	18.3	2.54
W&I (dev P2)	1,411	21.9	2.55
Test Sets for CAD Task			
CoNLL-2014 (test)	1,313	23.0	4.58
BEA-2019 (test)	4,478	19.1	-

Table 2: Statistic of used datasets. **Tok.** means the average token number in each example. **Corr.** means the average correction number in each example.

Test Datasets for CAD Task. As a post-processing task to improve the performance of GEC systems, we follow Omelianchuk et al. (2020); Awasthi et al. (2019) to report results on CoNLL-2014 test set (Ng et al., 2014) evaluated by official M^2 score (Dahlmeier and Ng, 2012) and BEA-2019 test set evaluated by ERRANT (Bryant et al., 2019). We first use GEC systems to correct the source sentence S_{src} in test sets and obtain the predicted sentence S_{pre}. Then based on Step 1 introduced in Section 3.1, we construct a set of corrected sentences S_{corr} as the test set for the CAD task.

4.2. Evaluation Metrics

Discriminator Evaluation Metrics. To test our discriminator, we define two metrics: (1) $Acc_{gold}^{1\text{vs}1}$, the ratio of the number of correct discriminations to the total number of input sentence pairs of (S_{src}, S_{gold}). This metric can be used to measure the model's ability of distinguishing incorrect sentences and corresponding ground truth ones. (2) $Acc_{gold}^{1\text{vs}N}$, the ratio of number of found S_{gold} to the total number of S_{gold} and the input sentence pairs to the discriminator are (S_{src}, S_{corr}) where S_{corr} represents corresponding sentences with different correction combinations from S_{gold}. As S_{gold} is directly related to C_{opt}, the metric can measure the ability of filtering invalid corrections.

Pipeline Evaluation Metrics. Since our proposed pipeline can be used to improve the performance of existing GEC systems, following previous GEC work (Omelianchuk et al., 2020; Awasthi et al., 2019), we use the precision P, recall R and $F_{0.5}$

as the evaluation metrics.

4.3. GEC Systems

We select a variety of popular GEC systems to verify the effectiveness of our CAD task: (1) **GECToR** (Tarnavskyi et al., 2022), which designs custom token-level transformations for GEC tasks and some SOTA results can be achieved based on it (Omelianchuk et al., 2020). (2) **PIE** (Awasthi et al., 2019), which presents a new parallel-iterative edit (PIE) architecture and uses an iterative predictive editing approach. (3) **UEdin-MS** (Grundkiewicz et al., 2019), which proposes a simple unsupervised synthetic error generation method to increase the amount of training data. (4) **Kakao** (Choe et al., 2019) uses noise to construct large amounts of fake data and uses transfer learning to build synthetic models for GEC tasks. (5) **PretLarge** (Kiyono et al., 2019), which conducts research on how to generate and use pseudo-data. (6) **IBM** (Kantor et al., 2019), which ensembles GEC systems in a nonlinear combinatorial fashion. (6) **Scoring** (Sorokin, 2022), which identifies the edits as positive or negative and calculates the probabilities to combine them.

4.4. Implementation Details

Discriminator. We implement our discriminator model with the base-cased version of BERT[1] which contains 110M parameters. We truncate the sentence with a limit of 50 tokens and use a batch size of 128. We apply AdamW optimizer with a learning rate 1e-5 for 5 epochs and select the checkpoint with the highest $Acc_{gold}^{1\text{vs}N}$ in development sets as our final model's parameters.

GEC Systems. We use GECToR (Omelianchuk et al., 2020) based on different pre-trained models, including BERT, RoBERTa (Liu et al., 2019), XLNet (Yang et al., 2019), RoBERTa-large, RoBERTa+XLNet (R+X) and BERT+RoBERTa+XLNet (B+R+X) for comparison. For PretLarge, we follow Kiyono et al. (2019) to incorporate it with the following techniques: Synthetic Spelling Error (SSE) (Lichtarge et al., 2019), Right-to-left Re-ranking (R2L) (Sennrich et al., 2016) and Sentence-level Error Detection (SED) (Asano et al., 2019).

Threshold Setting. The only hyperparameter in our pipeline is the threshold T, i.e., the number of corrections for each source sentence S_{src}. Figure 2 shows ratios of different correction numbers in training sets, which can reflect the distribution of error numbers in sentences. To balance the cost of calculation and the coverage of all correction's

[1] https://huggingface.co/bert-base-cased

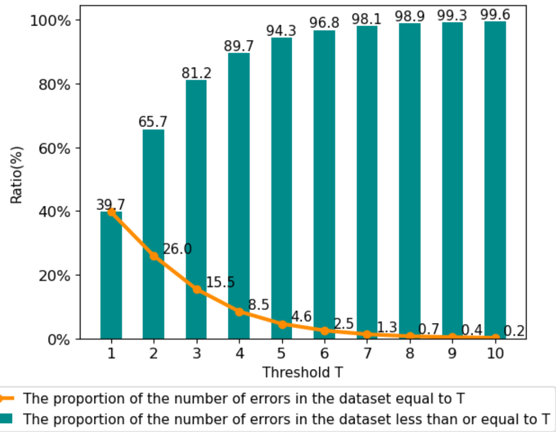

Figure 2: Ratios of correction numbers in training sets

combinations, we set the threshold T to 8 in our experiments which can cover 98.9% cases.

4.5. Ablation Study for Discriminator

We provide an ablation analysis of our proposed discriminator by using different components on two test datasets. The experimental results are shown in Tables 3. Overall, our proposed method outperforms other variants for all test cases. Specifically, for different method variants:

- ***Pooling strategy***, where we use BERT to obtain sentence embeddings with two pooling strategies: using the output's CLS-token and computing the mean of all output vectors (MEAN-strategy). We can observe that using MEAN-strategy is better than CLS-token on two test sets.
- ***Embedding fusion strategy***, where we fuse the embeddings of two sentences with three embedding fusion strategies: Hadamard product, add, subtract and take the absolute value. The results suggest that using subtract and take the absolute value operation is better than the others. The most significant improvement is gained on W&I dev P2 test set with up to a 6.73% increase of $Acc_{gold}^{1\text{vs}N}$.
- ***Symmetrical score operator***, where we use two score functions ($\frac{E_i \cdot E_{fusion}}{\sqrt{k}}$ where k is the embedding dimension of E_i and linear layer with learnable weight $W \in \mathbb{R}^{2d\times 1}$ where d is the dimension of sentence) and two normalization functions (softmax and sigmoid function) to output correctness score. The decrease in $Acc_{gold}^{1\text{vs}N}$ suggests that using softmax function is beneficial for our discriminator. Moreover, we can observe that up to 0.42% and 1.45% improvement on $Acc_{gold}^{1\text{vs}N}$ can be obtained on two test sets respectively. For W&I dev P2 and FCEtest set, the accuracy of using $\frac{E_i \cdot E_{fusion}}{\sqrt{k}}$

Pooling strategy	Fusion strategy	Score + Norm function	wer	W&I dev P2 $Acc_{gold}^{1vsN}(\%)$	FCE test $Acc_{gold}^{1vsN}(\%)$
mean	$\|s_1 - s_2\|$	cdot + softmax	✓	**80.65**	**81.64**
CLS	$\|s_1 - s_2\|$	cdot + softmax	✓	75.41	77.18
mean	$\frac{s_1+s_2}{2}$	cdot + softmax	✓	79.59	79.46
mean	$s_1 \odot s_2$	cdot + softmax	✓	73.92	73.60
mean	$\|s_1 - s_2\|$	linear + softmax	✓	80.37	80.52
mean	$\|s_1 - s_2\|$	linear + sigmoid	✓	78.32	75.43
mean	$\|s_1 - s_2\|$	cdot + sigmoid	✓	80.23	80.19
mean	$\|s_1 - s_2\|$	cdot + softmax	✗	80.01	80.29

Table 3: Ablation study results for discriminator in terms of Acc_{gold}^{1vsN} (%) on two test sets.

(cdot) is higher than that of using linear layer. Therefore, the symmetrical score operator is defined as follows:

$$E_{fusion} = |E_{src} - E_{gold}| \quad (5)$$

$$P_i = softmax(\frac{E_i \cdot E_{fusion}}{\sqrt{k}}) \quad (6)$$

- ***Loss function w/o wer***, where we don't use the word error rate wer in loss function. The degraded performance proves that using wer for training is effective.

4.6. Comparison Study for Discriminator

Considering that the goal of CAD task is to find the optimal sentence S_{opt} with the best correction combination C_{opt} in terms of the correctness acceptability, existing linguistic quality evaluation methods as mentioned in Section 1 can also achieve this purpose to some extent. Hence, we compare our discriminator with following baselines:

- GRUEN (Zhu and Bhat, 2020), which comprehensively considers the linguistic quality of a sentence from the four aspects of Grammaticality, Non-redundancy, Focus, Structure and Coherence.
- PPL-GPT2 (Radford et al., 2019), which calculates each sentence's perplexity (PPL) with GPT-2 (Radford et al., 2019) and the sentence with the lowest PPL is selected as the result. In our tests, we use the GPT-2 with 117M parameters[2].
- Single-Sent is a regression method based on the BERT model and its input is a single sentence and the output is its corresponding correctness score. The training examples consist of two groups: (1) S_{src} with the regression target 0, and (2) S_{gold} with the regress target 1. BERT is used to derive representations of S_{src} and S_{gold}. In testing, we select the sentence with the highest regression score among S_{src} and its corrected versions S_{corr} as the result.

[2]https://huggingface.co/gpt2

Method	W&I dev P2 Acc_{gold}^{1vs1}	W&I dev P2 Acc_{gold}^{1vsN}	FCE test Acc_{gold}^{1vs1}	FCE test Acc_{gold}^{1vsN}
GRUEN	84.76	61.80	85.04	58.71
PPL-GPT2	87.88	67.90	88.39	66.69
PPL-GPT2 (fine-tuned)	92.22	75.76	91.57	71.99
Single-Sent	92.84	77.18	93.86	77.62
Joint-Sents (src first)	**94.54**	69.81	94.70	69.75
Joint-Sents (corr first)	93.69	67.26	94.31	66.63
Ours	93.91	**80.65**	**95.03**	**81.64**

Table 4: Comparing our discriminator with four baselines in terms of Acc_{gold}^{1vs1} (%) and Acc_{gold}^{1vsN} (%).

- Joint-Sents is a binary classification method based on BERT. S_{src} and one of its corrected version S_{corr} are concatenated first and embedded by BERT. Then we classify the sentence pair with special token CLS's embedding. The selection step is the same as Step 3 mentioned in Section 3.3.

Note that, Single-Sent and Joint-Sents are two alternatives that constructed by us to generate correctness scores. Moreover, since Joint-Sents is sensitive to the sentence concatenation order, we train the model with different concatenation orders for a pair of sentences (S_{src}, S_{gold}). Specifically, when S_{src} occurs first, the target label is 1, and when S_{gold} is concatenated before S_{src}, the target label becomes 0.

Comparison Results. We compare our discriminator with the above baselines on two test datasets and report their results in Table 4. Obviously, for Acc_{gold}^{1vsN} which reflects the key to improve the performance of GEC systems, our discriminator achieves the best performance for all test cases. For example, on two datasets of W&I dev P2 and FCE test, our discriminator significantly outperforms GRUEN by up to 18.85% and 22.93%, and PPL-GPT2 by up to 12.75% and 14.95%. This is because GRUEN and PPL-GPT2 methods only consider the linguistic quality of one single sentence without considering the comparison with the source sentence. Our discriminator is also much better than baselines of Single-Sent and Joint-Sent in terms of Acc_{gold}^{1vsN}, which further proves its effectiveness for capturing the sentence with better correctness.

Our method also performs relatively better than

	CoNLL-2014 (MaxMatch)				BEA-2019 (ERRANT)			
System	P (%)	R (%)	$F_{0.5}$ (%)	Δ	P (%)	R (%)	$F_{0.5}$ (%)	Δ
GECToR (RoBERTa)	*73.91*	*41.66*	*64.00*		*77.13*	*55.26*	*71.47*	
GECToR + gruen	75.67	37.21	62.70	-1.30	79.8	50.22	71.39	-0.08
GECToR + gruen(fine-tuned)	75.61	37.37	62.76	-1.24	79.74	50.44	71.44	-0.03
GECToR + ppl	**77.18**	38.27	64.14	0.14	79.77	51.13	71.73	0.26
GECToR +ppl(fine-tuned)	76.68	38.73	63.92	-0.08	79.63	51.90	71.94	0.47
GECToR + Single-sent	76.81	38.02	63.79	-0.21	79.12	52.64	71.89	0.42
GECToR + Joint-sents(src first)	75.88	38.62	63.60	-0.40	**79.83**	50.68	71.59	0.12
GECToR + ours	75.05	**40.84**	**64.28**	**0.28**	78.85	**54.29**	**72.24**	**0.77**

Table 5: Results for different correctness discrimination methods over GECToR (RoBERTa). Δ is the improvement of $F_{0.5}$.

	CoNLL-2014 (MaxMatch)			BEA-2019 (ERRANT)		
System	P (%)	R (%)	$F_{0.5}$ (%)	P (%)	R (%)	$F_{0.5}$ (%)
GECToR (BERT)	72.07+1.17	42.13-0.75	63.06+0.41	71.41+2.02	55.96-0.85	67.67+1.18
GECToR (RoBERTa)	73.91+0.95	41.66-0.47	64.00+0.28	77.13+1.61	55.26-0.84	71.47+0.77
GECToR (XLNet)	77.49+1.31	40.15-0.61	65.34+0.40	79.18+1.84	54.11-1.08	72.46+0.82
GECToR-large (RoBERTa)	76.47+0.98	37.78-0.40	63.47+0.30	80.67+2.09	53.47-0.81	73.22+1.05
PIE	65.99+2.67	43.69-0.95	59.88+1.35	-	-	-
UEdin-MS	-	-	-	72.28+2.06	60.12-1.02	69.47+1.22
Kakao	-	-	-	75.19+2.00	51.91-0.52	69.00+1.15
Scoring (combined)	79.10+0.63	38.30-0.25	65.20+0.21	82.40+0.72	54.50-0.45	74.70+0.34
GECToR (R+X)	76.56+0.65	42.63-0.46	66.05+0.16	79.34+1.18	57.46-0.83	73.72+0.54
GECToR (B+R+X)	77.11+0.42	43.28-0.33	66.68+0.09	78.81+1.07	58.42-0.76	73.67+0.49
PretLarge(SSE+R2L)	72.40+0.81	46.07-0.43	64.97+0.35	72.14+1.70	61.77-0.79	69.80+1.05
PretLarge(SSE+R2L+SED)	73.26+0.78	44.17-0.40	64.73+0.31	74.71+1.19	56.67-0.68	70.24+0.62
IBM (UEdin-MS+Kakao)	-	-	-	78.31+1.70	58.00-1.00	73.18+0.85
Δ	+1.37	-0.51	+0.39	+1.92	-0.96	+1.01

Table 6: Results for improvements in GEC systems. Original results for GEC systems are copied from original papers. The values after '+' and '-' symbols mean the improvement and deterioration, respectively.

other baselines in terms of Acc_{gold}^{1vs1} except for the case of Joint-Sent on W&I dev P2 test dataset. One possible reason is that Joint-Sent uses a linear layer to classify the high-quality sentence which is good for Acc_{gold}^{1vs1}. However, as mentioned above, the metric Acc_{gold}^{1vsN} plays a more important role in the setting of GEC, hence it is reasonable to infer that our discriminator is advantageous for improving the performance of existing GEC systems. The results of Table 5 prove the inference.

In Table 5, we list the performance of GECToR (RoBERTa) in the first row as the reference, and then we apply different discrimination baselines to GECToR (RoBERTa) and list their testing results below. We can easily observe that compared-with the reference performance, both the precision scores P and $F_{0.5}$ have been increased while recall scores R are all dropped. This is because adding the discrimination step will inevitably remove some valid corrections, but the precision in the meantime can be greatly improved due to the fact that many invalid corrections are also rejected from the results of GECToR (RoBERTa). Moreover, we also find that our discriminator gets the best performance of $F_{0.5}$ among all baseline methods, which demonstrates the ability of our discriminator for improving GECToR (RoBERTa). In the next section, we further explore the ability of our discriminator for more GEC systems.

4.7. Improvements for GEC Systems

We select 13 existing GEC systems in total to integrate with our pipeline method. Due to some GEC systems do not report their results on the testing datasets, finally we have 10 GEC systems on the CoNLL-2014 test set and 12 GEC systems on the BEA-2019 test set. Table 6 shows the improvements for GEC systems by applying our pipeline on two test datasets.

Our pipeline improves both P and $F_{0.5}$ for all test cases, which demonstrates that our discriminator is effective in removing invalid corrections. Results show our method decreases on metric R, one reason is that our discriminator may mistakenly remove some valid corrections. However, $F_{0.5}$ metric which gives more weight to precision P than to recall R is more emphasized on GEC tasks(Ren et al., 2018). Hence our pipeline is of great significance to GEC tasks.

Particularly, for GEC systems that have relatively lower performance, our pipeline can greatly improve their results. For example, as shown in Table 6, the GEC systems PIE and UEdin-MS have been improved by up to 2.67%, 2.06% of P on CoNLL-2014 and BEA-2019, respectively.

5. Related Work

Many post-processing approaches have been proposed to improve GEC performance by removing error corrections. In this section, we note several prior works from the perspective of ensemble learning for GEC task and the linguistic quality evaluation.

The most widely applied post-processing method for GEC is ensemble learning because each GEC system has its pros and cons for different error types. One simple ensemble method is independently training several GEC models, which have the same architecture with different initial parameters, and averaging all models' probability distributions in inference (Zhao et al., 2019b; Awasthi et al., 2019; Omelianchuk et al., 2020). Tarnavskyi et al. (2022) find ensemble by voting with predicted edits works better than averaging probabilities to their sequence tagging approach. In the supervised ensemble, integer linear programming is used to optimize the $F_{0.5}$ score by combining different GEC systems and reviewing proposed edits (Kantor et al., 2019; Lin and Ng, 2021; Qorib et al., 2022). However, these existing post-processing methods only focus on each correction itself and ignore the relevant semantic information of these corrections combined with the source sentence. Moreover, these methods only work with more than one GEC system, which require huge resources that are unacceptable in some real scenarios. Our proposed pipeline method can work with all existing GEC techniques whether they are ensemble based or not.

There exist several linguistic quality evaluation methods that can also be adapted for post-processing the results of GEC systems. (Zhu and Bhat, 2020) utilizes a BERT-based model and four manually set features (grammaticality, non-redundancy, focus, structure, and coherence) to evaluate the sentence quality. (Ludwig et al., 2021) uses Transformer to vectorize essays, and proposes to use classification or regression to output essays to obtain essay evaluations. (Wang et al., 2022) use BERT to represent articles, and use multiple losses and transfer learning from out-of-domain essays to further improve the performance of essay scoring. However, when using these existing LQE methods to compare the S_{src} with the S_{corr}, they focus more on the sentence itself and ignore the semantic relation between the two sentences. Our correctness discriminator model can evaluate the grammatical acceptability by comparing with the source sentence with its corrected version, which is more suitable for GEC tasks.

6. Conclusion

This paper has presented a new task of CAD, and the proposed pipeline method for solving the task is "plug-and-play" with existing GEC systems. As the core of the pipeline, the discriminator we designed can effectively remove invalid corrections from the output of a GEC system. The extensive experiments have shown that our discriminator has an obvious advantage over existing linguistic quality evaluation methods for correctness acceptability comparison. Moreover, the $F_{0.5}$ scores of all 13 selected GEC systems have been improved after applying our pipeline.

In the future, we will explore more effective strategy for optimizing the discriminator, and we are interested in adapting CAD task to other languages like Chinese and Russian.

7. Acknowledgments

This research was partially supported by STI 2030-Major Projects 2021ZD0200400, National Natural Science Foundation of China (62276233 and 62072405) and Key Research Project of Zhejiang Province (2023C01048).

8. Bibliographical References

Hiroki Asano, Masato Mita, Tomoya Mizumoto, and Jun Suzuki. 2019. The aip-tohoku system at the bea-2019 shared task. In *BEA 2019*, pages 176–182.

Abhijeet Awasthi, Sunita Sarawagi, Rasna Goyal, Sabyasachi Ghosh, and Vihari Piratla. 2019. Parallel iterative edit models for local sequence transduction. In *EMNLP/IJCNLP*.

Christopher Bryant, Mariano Felice, Øistein E Andersen, and Ted Briscoe. 2019. The bea-2019 shared task on grammatical error correction. In *ACL*, pages 52–75.

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

BSI. 1973a. *Natural Fibre Twines*, 3rd edition. British Standards Institution, London. BS 2570.

BSI. 1973b. Natural fibre twines. BS 2570, British Standards Institution, London. 3rd. edn.

A. Castor and L. E. Pollux. 1992. The use of user modelling to guide inference and learning. *Applied Intelligence*, 2(1):37–53.

J.L. Chercheur. 1994. *Case-Based Reasoning*, 2nd edition. Morgan Kaufman Publishers, San Mateo, CA.

Yo Joong Choe, Jiyeon Ham, Kyubyong Park, and Yeoil Yoon. 2019. A neural grammatical error correction system built on better pre-training and sequential transfer learning. *arXiv preprint arXiv:1907.01256*.

N. Chomsky. 1973. Conditions on transformations. In *A festschrift for Morris Halle*, New York. Holt, Rinehart & Winston.

John Conroy and Hoa Trang Dang. 2008. Mind the gap: Dangers of divorcing evaluations of summary content from linguistic quality. In *COLING*, pages 145–152.

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

Daniel Dahlmeier, Hwee Tou Ng, and Siew Mei Wu. 2013. Building a large annotated corpus of learner english: The NUS corpus of learner english. *In NAACL/HLT*, page 22.

Hoa Trang Dang. 2006. Overview of duc 2006. In *Proceedings of the document understanding conference*.

Vidas Daudaravicius, Rafael E Banchs, Elena Volodina, and Courtney Napoles. 2016. A report on the automatic evaluation of scientific writing shared task. In *Innovative Use of NLP for Building Educational Applications*, pages 53–62.

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

Umberto Eco. 1990. *The Limits of Interpretation*. Indian University Press.

Mariano Felice, Zheng Yuan, Øistein E Andersen, Helen Yannakoudakis, and Ekaterina Kochmar. 2014. Grammatical error correction using hybrid systems and type filtering. *In CoNLL*, page 15.

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

Paul Gerhard Hoel. 1971a. *Elementary Statistics*, 3rd edition. Wiley series in probability and mathematical statistics. Wiley, New York, Chichester. ISBN 0 471 40300.

Paul Gerhard Hoel. 1971b. *Elementary Statistics*, 3rd edition, Wiley series in probability and mathematical statistics, pages 19–33. Wiley, New York, Chichester. ISBN 0 471 40300.

Otto Jespersen. 1922. *Language: Its Nature, Development, and Origin*. Allen and Unwin.

Marcin Junczys-Dowmunt, Roman Grundkiewicz, Shubha Guha, and Kenneth Heafield. 2018. Approaching neural grammatical error correction as a low-resource machine translation task. In *NAACL*, pages 595–606. Association for Computational Linguistics.

William Kahan. 1996. Ieee standard 754 for binary floating-point arithmetic. *Lecture Notes on the Status of IEEE*, 754(94720-1776):11.

Yoav Kantor, Yoav Katz, Leshem Choshen, Edo Cohen-Karlik, Naftali Liberman, Assaf Toledo, Amir Menczel, and Noam Slonim. 2019. Learning to combine grammatical error corrections. *arXiv preprint arXiv:1906.03897*.

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

Dietrich Klakow and Jochen Peters. 2002. Testing the correlation of word error rate and perplexity. *In Speech Communication*, 38(1-2):19–28.

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

Ruixi Lin and Hwee Tou Ng. 2021. System combination for grammatical error correction based on integer programming. In *RANLP*, pages 824–829.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Sabrina Ludwig, Christian Mayer, Christopher Hansen, Kerstin Eilers, and Steffen Brandt. 2021. Automated essay scoring using transformer models. *In Psych*, 3(4):897–915.

André FT Martins, Marcin Junczys-Dowmunt, Fabio N Kepler, Ramón Astudillo, Chris Hokamp, and Roman Grundkiewicz. 2017. Pushing the limits of translation quality estimation. *In TACL*, 5:205–218.

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

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 *CoNLL*, pages 1–14.

Kostiantyn Omelianchuk, Vitaliy Atrasevych, Artem Chernodub, and Oleksandr Skurzhanskyi. 2020. Gector–grammatical error correction: Tag, not rewrite. *In ACL*, page 163.

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.

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

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *EMNLP-IJCNLP*, pages 3982–3992.

Hongkai Ren, Liner Yang, and Endong Xun. 2018. A sequence to sequence learning for chinese grammatical error correction. In *NLPCC*, pages 401–410. Springer.

Eric Sven Ristad and Peter N Yianilos. 1998. Learning string-edit distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(5):522–532.

Sascha Rothe, Jonathan Mallinson, Eric Malmi, Sebastian Krause, and Aliaksei Severyn. 2021. A simple recipe for multilingual grammatical error correction. In *ACL/IJCNLP*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *ACL*.

Charles Joseph Singer, E. J. Holmyard, and A. R. Hall, editors. 1954–58. *A history of technology*. Oxford University Press, London. 5 vol.

Alexey Sorokin. 2022. Improved grammatical error correction by ranking elementary edits. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11416–11429.

Jannik Strötgen and Michael Gertz. 2012. Temporal tagging on different domains: Challenges, strategies, and gold standards. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, pages 3746–3753, Istanbul, Turkey. European Language Resource Association (ELRA).

S. Superman, B. Batman, C. Catwoman, and S. Spiderman. 2000. *Superheroes experiences with books*, 20th edition. The Phantom Editors Associates, Gotham City.

Raymond Hendy Susanto, Peter Phandi, and Hwee Tou Ng. 2014. System combination for grammatical error correction. In *EMNLP*, pages 951–962.

Maksym Tarnavskyi, Artem Chernodub, and Kostiantyn Omelianchuk. 2022. Ensembling and knowledge distilling of large sequence taggers for grammatical error correction. In *ACL*, pages 3842–3852.

Yongjie Wang, Chuan Wang, Ruobing Li, and Hui Lin. 2022. On the use of bert for automated essay scoring: Joint learning of multi-scale essay representation. *arXiv preprint arXiv:2205.03835*.

Stratos Xenouleas, Prodromos Malakasiotis, Marianna Apidianaki, and Ion Androutsopoulos. 2019. Sum-qe: a bert-based summary quality estimation model. In *EMNLP-IJCNLP*. The Association for Computational Linguistics.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *In NIPS*, 32.

Helen Yannakoudakis, Ted Briscoe, and Ben Medlock. 2011. A new dataset and method for automatically grading esol texts. In *ACL*, pages 180–189.

Wei Zhao, Maxime Peyrard, Fei Liu, Yang Gao, Christian M Meyer, and Steffen Eger. 2019a. Moverscore: Text generation evaluating with contextualized embeddings and earth mover distance. In *EMNLP-IJCNLP*, pages 563–578.

Wei Zhao, Liang Wang, Kewei Shen, Ruoyu Jia, and Jingming Liu. 2019b. Improving grammatical error correction via pre-training a copy-augmented architecture with unlabeled data. In *NAACL-HLT*.

Wanzheng Zhu and Suma Bhat. 2020. Gruen for evaluating linguistic quality of generated text. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 94–108.