Robust and Effective Grammatical Error Correction with Simple Cycle Self-Augmenting

Anonymous ACL submission

Abstract

Recent studies have revealed that grammatical error correction methods in the sequence-to-sequence paradigm are vulnerable to adversarial attack, and simply utilizing adversarial examples in the pre-training or post-training process can significantly enhance the robustness of GEC models to certain types of attack without suffering too much performance loss on clean data. In this paper, we further conduct a thorough robustness evaluation of cutting-edge GEC methods to four different types of adversarial attacks and propose a simple yet very effective Cycle Self-Augmenting (CSA) method accordingly. By leveraging the augmenting data from the GEC models themselves in the post-training process and introducing regularization data for cycle training, our proposed method can effectively improve model robustness of well-trained GEC models with only a few more training epochs as the extra cost. Experiments on four benchmark datasets and seven strong models indicate that our proposed training method can significantly enhance the robustness to four types of attacks without using purposely built adversarial examples in training. Evaluation results on clean data further confirm that our proposed CSA method significantly improves the performance of four baselines and yields nearly comparable results with other state-of-the-art models.[1]

1 Introduction

Grammatical error correction (GEC) is one of the most essential application tasks in the NLP community for its crucial values in many scenarios including, but not limited to, writing assistant (Napoles et al., 2019; Fitria, 2021), automatic speech recognition (Karat et al., 1999; Namazifar et al., 2021; Zhao et al., 2021; Wang et al., 2021; Zhang et al., 2021), information retrieval (Gao et al., 2010; Duan and Hsu, 2011; Hagen et al., 2017; Zhuang and Zuccon, 2021), which mainly aims to detect and correct various textual errors, such as spelling, punctuation, grammatical, word choice, and other article mistakes (Wang et al., 2020). Existing solutions to tackle this task can be roughly divided into two categories, i.e., sequence-to-sequence generation (*Seq2Seq*) (Ji et al., 2017; Chollampatt and Ng, 2018) and sequence-to-editing (*Seq2Edits*) (Awasthi et al., 2019; Li and Shi, 2021). The former group performs the translation from ungrammatical sentences to the corresponding error-free sentences, while the latter introduces tagging or sequence labeling to merely edit a small proportion of the input sentences, remaining the rest part unchanged.

With the well-tested encoder-decoder framework (Sutskever et al., 2014; Vaswani et al., 2017) as the backbone, GEC methods in the *Seq2Seq* paradigm can achieve promising performance but is sensitive to the quality and scale of training data. Thus, many recent works have studied the problem of automatically obtaining high-quality paired data to compensate for the lack of human-labeled data pairs (Zhao et al., 2019; Kiyono et al., 2019; Yasunaga et al., 2021). As for the *Seq2Edits* group, it performs better and faster than *Seq2Seq* methods under limited data resources but requires labeled data for intermediate tasks, e.g., tagging, sequence labeling. Existing literature has also revealed that incorporating large-scale pre-trained language models (PLMs) can enhance the GEC performance of both *Seq2Seq* (Kaneko et al., 2020) and *Seq2Edits* (Malmi et al., 2019; Omelianchuk et al., 2020) methods. However, recent studies have disclosed that *Seq2Seq* GEC models (even with data augmentation) are vulnerable to adversarial examples (Wang and Zheng, 2020). Studies on other classification tasks and PLMs further hint at the possible vulnerability of PLMs-based GEC methods (Li et al., 2021). In view of the above-mentioned facts, it is imperative to conduct a sys-

[1] Our code is available in the supplementary .zip file, which will be released after the anonymous period.

tematical evaluation of existing GEC methods to adversarial attacks, especially for the under-explored *Seq2Edits* paradigm and PLMs-based models.

To fill this gap, we propose to evaluate the robustness of cutting-edge GEC models to different adversarial attacks. More concretely, we introduce four textual adversarial attack methods to construct different variants for each original test set, including back-translation (Xie et al., 2018), antonym substitution (Ma, 2019), mapping & rules (Wang and Zheng, 2020), synonyms substitution (Li et al., 2021). Resembling the observation on the previous *Seq2Seq* method attacked by mapping & rules, cutting-edge GEC models are also very sensitive to the introduced attacks. Taking the BART-based method (Lewis et al., 2020; Katsumata and Komachi, 2020) for example, its performance ($F_{0.5}$) on CoNLL-2014 (Ng et al., 2014) decreases sharply, from 62.6 to 36.8. Intuitively, the dramatic performance decline can be mitigated by pre-training or post-training with a great number of adversarial examples for a certain type of attack (Wang and Zheng, 2020). However, such methods require preparing considerable data for each attack type in advance, which is infeasible for real-world scenarios. Another minor flaw of these methods is that the significant improvement in robustness is possibly accompanied by the performance decrease on clean data.

To avoid these problems, we propose a simple yet very effective cycle self-augmenting (CSA) method. Concretely, our proposed CSA is only introduced in the post-training process of a converged GEC model and merely needs the original training data. Through utilizing self-augmenting data pairs and the regularization data sub-sets in cycle training, our proposed simple CSA can significantly improve model robustness with only a few more training epochs as the extra cost. Since our CSA no longer requires well-crafted adversarial examples for model training, it is more feasible in applications and can generalize well to different GEC frameworks. Experimental results on **seven** strong models (e.g., BERT-fuse, BART, RoBERTa, XLNET) and **four** benchmark datasets (i.e., BEA, CoNLL, FCE, JFLEG) demonstrate the effectiveness of our proposed simple method. Our CSA method achieves significant robustness improvement on all settings and at the same time yields meaningful performance improvement on clean data (four out of seven tested models), with nearly comparable results for the left three SOTA baselines. Besides, we also observe that the trade-off between the robustness to attack and the performance on clean data is associated with regularization examples, where more regularization pairs in training lead to better robustness but with performance decline on clean data, and vise versa.

2 Preliminary

In this section, we briefly summarize the key components of cutting-edge GEC methods and present a few representative works correlated with the robustness of GEC models against adversarial attacks. We first review some typical methods for obtaining synthetic data and then introduce two most popular GEC model architectures in existing literature, i.e., *Seq2Seq* and *Seq2Edits*, following with the pilot studies of adversarial attack in GEC and a few widely-used attack methods for other NLP tasks.

2.1 Synthetic Data

The recent success of GEC models highly relies on the availability of massive training data pairs (Koehn and Knowles, 2017). Considering that human-labeled pairs are expensive to obtain, many efforts have been devoted to exploring the automatic generation of pseudo data pairs for GEC (Xie et al., 2018; Ge et al., 2018; Lichtarge et al., 2019; Awasthi et al., 2019; Grundkiewicz et al., 2019; Náplava and Straka, 2019), and the combination of synthetically generated data has almost been indispensable for recently proposed GEC models (Kiyono et al., 2019). Specifically, (Ge et al., 2018) propose a fluency boost learning method during the training stage to extend the dataset. Zhao et al. propose to directly inject noise into grammatical sentences. Xie et al.; Lichtarge et al.; Zhou et al. use translation methods to automatically generate the Poor-Good pairs. Wan et al. combine a classifier with a *Seq2Seq* model to generates specified types of errors. (Yasunaga et al., 2021) leverage a BIFI framework (Yasunaga and Liang, 2021) to generate more realistic ungrammatical sentences.

2.2 Model Architecture

The goal of Grammatical Error Correction is to map ungrammatical pieces x_i into grammatical ones y_i with the use of *Seq2Seq* model architecture (Sutskever et al., 2014; Vaswani et al., 2017; Lewis et al., 2020) or *Seq2Edits* framework (Awasthi et al., 2019; Devlin et al., 2019).

Seq2Seq Many researches (Xie et al., 2016; Yuan and Briscoe, 2016; Xie et al., 2018; Junczys-Dowmunt et al., 2018; Zhao et al., 2019; Sun et al., 2021; Kaneko et al., 2020) regard GEC as a natural language generation (NLG) task and utilize an encoder-decoder structure to complete the sequence-to-sequence (*Seq2Seq*) generation task.

Given an input sentence $\boldsymbol{x}$ of N tokens, the encoder first encodes it into the hidden representation $h^s_{1:N}$, and then the decoder outputs each token in an auto-regressive fashion. The output distribution over the vocabulary at the k-th decoding step is conditioned on $h^s_{1:N}$ from the encoder and the summarized representation of previously generated k-1 tokens $h^t_{1:k-1}$ from the decoder, formulated as $\Pr(y_k|\boldsymbol{y}_{<k}, \boldsymbol{x}) = \Pr(y_k|h^t_{1:k-1}, h^s_{1:N})$. The training objective of *Seq2Seq* model architecture is the negative log-likelihood, written by

$$\mathcal{L}(\theta) = -\frac{1}{|D|}\sum_{x,y\in D} log(p(y|x)) \quad (1)$$

where θ refers to trainable model parameters. To get a optimal output, beam search decoding (Yuan and Briscoe, 2016; Chollampatt and Ng, 2018) and its variation are also utilized (Sun et al., 2021). This architecture can achieve promising performance with a huge amount of data but will sacrifice inference efficiency owing to the iterative decoding.

Seq2Edits To alleviate the embarrassed situation of inference speed and data hungry problems in *Seq2Seq* model architecture, *Seq2Edits* provides another alternative that casts GEC into a tagging problem (Awasthi et al., 2019; Omelianchuk et al., 2020; Malmi et al., 2019) along with the non-autoregressive sequence prediction (Li and Shi, 2021). Instead of directly predicting the token, *Seq2Edits* architecture first predicts the edit operation type e_i for each input token x_i and then perform a series of transformation operations based on the predicted edit to realize the grammatical output. The training objective of tagging is formulated as,

$$\mathcal{C}(\phi) = -\frac{1}{|D|}\sum_{x\in D, e\in E} log(p(e|x)) \quad (2)$$

where ϕ corresponds to model parameters to be trained. This architecture can achieve competitive performance and faster inference speed with limited data but requires heuristic prior and human efforts to obtain labeled data for the tagging task.

2.3 Adversarial Attack

Recent studies on the GEC task have revealed that existing *Seq2Seq* methods are quite vulnerable to adversarial examples under the white-box setting. To obtain adversarial examples, Wan et al. propose to first identify the weak spots of a model and then replace the vulnerable tokens with two different strategies. One is to create a correct-to-error mapping from the GEC training set. Another is to present a series of substitution rules if there is no candidate in the mapping. Hereafter, we denote this method as Mapping & Rules for short. There are also other popular adversarial example construction methods for PLMs and other tasks but are less explored in GEC such as word substitutions (Ma, 2019; Dong et al., 2021; Li et al., 2021).

3 Cycle Self-Augmenting Method

In this section, we introduce our simple Cycle Self-Augmenting Method (**CSA**). We illustrate how Self-Augmenting and Cycle Training work under our settings in Section 3.1 and Section 3.2, respectively. In the cycle training process, We present the concept of regularization data for GEC, which is the key to robustness against adversarial attacks.

3.1 Self-Augmenting

To enhance model robustness, existing works mainly create well-crafted adversarial examples of considerable magnitude for certain types of adversarial attacks and use these data in the pre-training or/and post-training stages (Wang and Zheng, 2020; Li et al., 2021). Instead of carefully designing adversarial example generation strategies for each type of attack, we leverage the GEC model itself to perform self-augmenting to defend against various types of attack, which is more efficient and can generalize well to varied GEC models. To better utilize the capability of GEC models, we introduce our self-augmenting mechanism in post-training.

Concretely, the crux of Self-Augmenting is to obtain augmenting data pairs for post-training, in which the detailed process is illustrated in Figure 1. Given a well-trained GEC model $f(\cdot)$ and the original training dataset $\mathcal{D}$={(X, Y)}, we feed each input x into $f(\cdot)$ to obtain the corresponded output y' (step ①-② in Figure 1). Then, we compare the predicted y' with the golden sentence y of input x (step ③). If $y' \neq y$, we will collect (y', y) as augmenting pairs to further post-train the GEC model (step ④). After processing all the pairs in the origi-

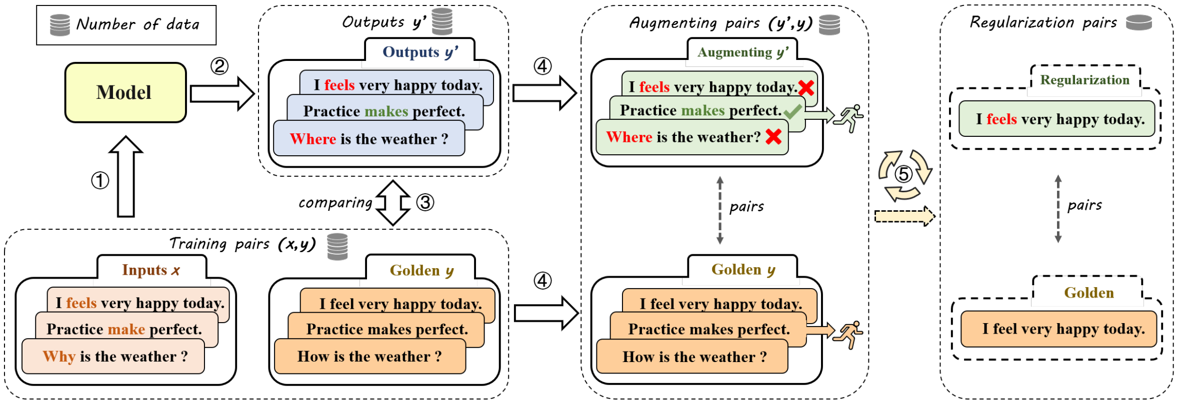


Figure 1: The overall framework of our proposed Cycle Self-Augmenting (CSA). The Self-Augmenting mechanism correlates with step ①-④. We launch cycle training in step ⑤, along with the utilization of regularization data.

nal training dataset $\mathcal{D}$, we can obtain a new dataset $\mathcal{D}_{Aug}$, comprising of augmenting pairs.

Intuitively, one can simply collect (x, y) as augmenting pair to perform further training or collect $(x, y^{'})$ for self-distillation (Mobahi et al., 2020). However, post-training converged GEC model with part of the original dataset $\mathcal{D}$ can lead to over-fitting, and self-distillation is not applicable for the GEC task, i.e., the target is to obtain grammatical outputs. Instead, utilizing $(y^{'}, y)$ for post-training can provide more feasible training pairs and is more tally with the GEC task, i.e., only part of the input sentence is edited. Besides, such a strategy enables GEC models to perform multiple refinements at inference by post-editing the unexpected output $y^{'}$ as golden sentence y. We will show the superiority of our self-augmenting in Section 5.2.

3.2 Cycle Training

To effectively utilize augmenting pairs from the above introduced self-augmenting process, we further present a cycle training strategy, which is sketched out in step ⑤ of Figure 1. Specifically, we use the self-augmenting mechanism to construct a new dataset $\mathcal{D}^{k}_{Aug}$ in each cycle k, where $0 < k \leq \epsilon$. Thus, we can leverage ϵ augmenting datasets $(\mathcal{D}^{1}_{Aug}, \dots, \mathcal{D}^{k}_{Aug}, \dots, \mathcal{D}^{\epsilon}_{Aug})$ in cycle training, where these datasets are divided into two groups for different training stages.

In **Stage I**, the obtained augmenting datasets contain many unseen data pairs in the original training dataset, which can be simply used by conducting further training to improve both model performance and robustness. Accordingly, we adopt the following training process for each cycle at the early stage, i.e., when $0 < k \leq \mathcal{P}$:

- Perform training on $\mathcal{D}^{k}_{Aug}$ until convergence.
- Conduct further tuning on a small high-quality GEC dataset $\mathcal{D}_{tune}$ to prevent over-fitting on the augmenting dataset.

Note that the improvement of performance and robustness is not caused by merely using the small dataset, which is discussed in Section 5.3.

Along with the model training, there are fewer and fewer unseen data pairs in the augmenting datasets. Simply utilizing the augmenting dataset in each cycle for model training might yield over-fitting on these datasets. Thus, we turn to focus on these hard-to-learn data, i.e., these data pairs that have not been learned after $\mathcal{P}$ cycles. Inspired by previous work (Zhou et al., 2021) that names some specific samples that are negatively associated with the performance of knowledge distillation as regularization examples, we treat these hard-to-learn data as **Regularization Data** for the GEC task. When $\mathcal{P} \leq k < \epsilon$, the regularization data of the k-th cycle is obtained as $\mathcal{D}^{k}_{Reg} = \mathcal{D}^{k-p+1}_{Aug} \cap \dots \cap \mathcal{D}^{k}_{Aug}$. In this stage (**Stage II**), the trained GEC model from **Stage I** is further trained as below:

- Perform training on $\mathcal{D}^{k}_{Reg}$ until convergence.
- Conduct further tuning on a small high-quality GEC dataset $\mathcal{D}_{tune}$.

The benefits of launching further training on regularization data are four-folds: 1) it prevents over-fitting on the easy-to-learn data pairs in the augmenting datasets; 2) it can reduce model capacity to improve its generalization ability and robustness; 3) it gives more opportunities for the model to address hard-to-learn pairs; 4) it can accelerate each training cycle by using fewer data pairs. More analysis of regularization data is given in Section 5.4.

4 Experiments

We conduct experiments on both clean data and attack sets to evaluate the effectiveness of our proposed CSA method. We first present necessary details about datasets and evaluations, following with the description of baselines and concrete implementation settings of all models. We then give the evaluation results on clean data and attack sets.

4.1 Datasets and Evaluations

Train Sets Table 1 describes all the datasets that are utilized in model training. Following the previous study (Omelianchuk et al., 2020), we leverage these datasets in two different training phases:

- *Pre-Training.* In this phase, we use 9M pseudo parallel sentences with synthetic errors for pre-training (Awasthi et al., 2019)[2].
- *Fine-Tuning.* During the fine-tuning phase, we use the official corpora from BEA-2019 shared task (Bryant et al., 2019)[3] for fine-tuning, which comprises four datasets, i.e., Lang-8 Corpus of Learner English (Lang-8) (Mizumoto et al., 2011; Tajiri et al., 2012), National University of Singapore Corpus of Learner English (NUCLE) (Dahlmeier et al., 2013), the First Certificate in English (FCE) (Yannakoudakis et al., 2011), and Cambridge English Write & Improve + LOCNESS Corpus (W&I+LOCNESS) (Granger, 1998; Yannakoudakis et al., 2018).

We split out validation data by random sampling from the official training corpora with a ratio of 2/98 and decompose the fine-tuning phase into two stages. In stage I, the model is fine-tuned on errorful-only sentences. In stage II, the model is tuned on a high-quality and more realistic dataset as in (Kiyono et al., 2019; Omelianchuk et al., 2020).

Attack Sets The core of building adversarial examples is to confuse the model. For this purpose, we introduce four textual adversarial attack methods to construct different variants for each original test sets, including back-translation (Xie et al., 2018), mapping & rules (Wang and Zheng, 2020), antonym substitution (Ma, 2019) and synonyms substitution (Li et al., 2021). The construction details of adversarial examples are as follows:

Dataset	#Sentences	Errors (%)	Usage
PIE-synthetic	9,000,000	100.0 %	Pre-training
Lang-8	1,102,868	51.1 %	Fine-tuning †
FCE	34,490	62.6 %	Fine-tuning †
NUCLE	57,151	38.2 %	Fine-tuning †
W&I+LOCNESS	34,308	66.3 %	Fine-tuning ‡

Table 1: Statistics of datasets used in our experiments. "†" denotes data used in fine-tuning stage I, while "‡" refers to data used in both fine-tuning stage I and II.

- *Back-Translation.* We reverse the examples from pre-training datasets to train a back-translation model which can generate errorful examples from a clean corpus. Then we implement a back-translation method variant (Xie et al., 2018) which adds $r\beta_{random}$ to penalize every hypothesis during the beam search step, where r is drawn uniformly from the interval [0, 1] and β_{random} is a hyper-parameter sampling from $\mathbb{R}$. We follow the previous work (Kiyono et al., 2019) to set β_{random} = 6.
- *Mapping & Rules.* We first build a $Poor \mapsto Good$ mapping from training datasets. Then, we utilize the method in previous work (Wan et al., 2020) to apply word substitution-based perturbations to each corpus.
- *Antonym and Synonyms Substitutions.* We detect the vulnerable tokens by using the method proposed by Wan et al. and simply use the open source tools NLPAug (Ma, 2019)[4] to substitute opposite meaning word according to WordNet antonym (Miller, 1995) or substitute similar word according to WordNet/PPDB (of Hertfordshire, 2007) synonym.

Evaluations We report results on the test sets of BEA, CoNLL-2014 (Ng et al., 2014), FCE, and JFLEG (Napoles et al., 2017). We measure the results of CoNLL-2014 and FCE by M^2 scorer (Dahlmeier and Ng, 2012). For JELEG results, we use the GLEU metric (Napoles et al., 2015, 2016). We report the scores measured by ERRANT (Bryant et al., 2017; Felice et al., 2016) for BEA-test. As the reference of the BEA-test are unavailable, we report results from CodaLab[5].

We utilize the obtained attack sets for each test set to evaluate the defense capability of different models. As each variant of the attack set is constructed from the original test set, we leverage the

[2]https://drive.google.com/open?id=1bl5reJ-XhPEfEaPjvO45M7w0yN-0XGOA
[3]https://www.cl.cam.ac.uk/research/nl/bea2019st/
[4]https://github.com/makcedward/nlpaug
[5]https://competitions.codalab.org/competitions/20228

Model	BEA (ERRANT)			CoNLL-2014 (M^2)			FCE (M^2)			JELEG
	Prec.	**Rec.**	**F_0.5**	**Prec.**	**Rec.**	**F_0.5**	**Prec.**	**Rec.**	**F_0.5**	**GLEU**
Transformer (Kiyono et al., 2019)	65.5	59.4	64.2	68.9	43.9	61.8	59.4*	39.5*	54.0*	59.7
+ **Ours** (4 Cycles)	68.4	65.5	**67.9**	67.5	49.4	**62.9**	60.5	43.4	**56.1**	**61.4**
BERT-fuse (Kaneko et al., 2020)	67.1	60.1	65.6	69.2	45.6	62.6	59.8	46.9	56.7	61.3
+ **Ours** (3 Cycles)	68.9	64.5	**68.0**	69.4	49.8	**64.4**	64.4	46.6	**59.9**	**62.5**
BART (Katsumata and Komachi, 2020)	68.3	57.1	65.6	69.3	45.0	62.6	59.6*	40.3*	54.4*	57.3
+ **Ours** (2 Cycles)	68.8	63.4	**67.7**	68.8	48.6	**63.5**	65.2	34.4	**55.3**	**59.4**
RoBERTa(Omelianchuk et al., 2020)	68.4	60.8	66.8	68.7	47.2	**62.9**	61.6*	45.3*	57.5*	**59.1***
+ **Ours** (1 Cycle)	68.8	60.3	**66.9**	68.0	46.9	62.4	62.7	44.8	**58.0**	58.6
BERT(Omelianchuk et al., 2020)	71.5	55.7	**67.6**	72.1	42.0	**63.0**	66.2*	42.0*	**59.4***	57.5*
+ **Ours** (1 Cycle)	67.7	57.2	65.3	70.0	44.3	62.3	64.0	43.1	58.3	**57.8**
XLNet (Omelianchuk et al., 2020)	79.2	53.9	**72.4**	77.5	40.1	**65.3**	71.9*	41.3*	62.7*	56.0*
+ **Ours** (1 Cycle)	77.8	55.0	71.8	75.3	41.6	64.8	71.5	42.7	**63.1**	**56.5**
LM-Critic (Yasunaga et al., 2021)	51.6	24.7	42.4	64.4	35.6	55.5	49.6*	24.6*	41.2*	51.4*
+ **Ours** (2 Cycles)	67.0	46.5	**61.6**	65.7	47.4	**61.0**	58.0	39.6	**53.1**	**59.1**

Table 2: Evaluation results on clean data. The numbers labeled with "*" refer to the results tested by ourselves with the released checkpoints from the original papers, while all the left numbers are copied from the original papers. We also present the cycle times for each model. The blackened fonts are the optimal performance of each comparison.

same metrics to calibrate model robustness, i.e., M^2 scorer, GLEU metric, and ERRANT.

4.2 Baselines and Settings

Note that our proposed CSA method is a post-training strategy, which can be utilized upon any neural GEC model. We leverage seven cutting-edge models as our baselines to conduct experiments under the supervised setting. It should be clarified that if there exists a publicly available checkpoint for each baseline model, we will use it directly. Otherwise, we will follow the original settings to train a model by ourselves. Specifically, we carry out experiments on **Transformer** (Kiyono et al., 2019)[6], **BERT-fuse** (Kaneko et al., 2020)[7], **BART** large (Katsumata and Komachi, 2020)[8], three model variants (**RoBERTa**, **BERT**, **XLNet**) based on large-scale pre-trained language models in GECToR (Omelianchuk et al., 2020)[9], and the **LM-Critic** method (Yasunaga et al., 2021)[10]. As for CSA, we set max cycle times $\epsilon = 5$ and patience $\mathcal{P} = 2$. If the model performance does not improve over two consecutive cycles, the training process is stopped. During the post-training stage, all hyper-parameter settings are the same with baselines.

[6] https://github.com/butsugiri/gec-pseudodata
[7] https://github.com/bert-nmt/bert-nmt
[8] https://github.com/Katsumata420/generic-pretrained-GEC
[9] https://github.com/grammarly/gector
[10] https://github.com/michiyasunaga/LM-Critic

4.3 Main Results

GEC Results We first present the experimental results on four clean sets to calibrate the influence of our proposed CSA to baseline models, where the detailed numbers are shown in Table 2. It can be seen that our proposed simple CSA method yields impressive performance improvement on four baselines, i.e., Transformer, BERT-fuse, BART, and LM-Critic. For the rest three strong baselines, our proposed method also does not degrade model performance by achieving comparable scores [11].

Attack Results In table 3, we report the evaluation results on the attack sets. Recall that we construct four variants of attack sets with different methods for each original test set. To better show the effectiveness of our proposed CSA method, we utilize the averaged results of four attack sets for each original test set, where more detailed results are given in the Appendix. It can be observed that our proposed simple CSA method yields robustness improvement on all baseline methods. In particular, our CSA leads to the improvements of 4.9 ($F_{0.5}$) points over BERT-fuse and 5.1 ($F_{0.5}$) points over the BART model on the CoNLL-2014 attack sets.

5 Analysis and Discussion

In this section, we conduct extensive studies from different perspectives to better understand our CSA

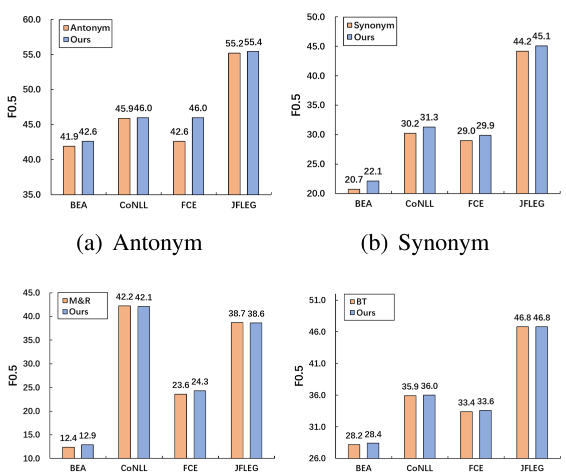
[11] The released checkpoints of baselines have already been meticulously trained on existing datasets, and any further post-training may hurt their performance.

Model	BEA (ERRANT)			CoNLL-2014 (M^2)			FCE (M^2)			JFLEG
	Prec.	Rec.	F_0.5	Prec.	Rec.	F_0.5	Prec.	Rec.	F_0.5	GLEU
Transformer (Kiyono et al., 2019)	21.0	48.0	23.4	34.1	39.7	34.9	29.7	34.2	30.3	45.4
+ Ours (4 Cycles)	23.7	53.2	**26.4**	37.7	45.5	**38.9**	32.5	38.8	**33.5**	**46.5**
BERT-fuse (Kaneko et al., 2020)	20.4	46.1	22.6	33.5	38.2	34.1	31.0	34.5	31.4	45.4
+ Ours (3 Cycles)	23.8	53.7	**26.6**	37.9	45.5	**39.0**	33.7	40.0	**34.6**	**47.0**
BART (Katsumata and Komachi, 2020)	20.9	44.7	23.0	34.5	38.8	35.0	30.1	31.5	30.0	43.8
+ Ours (2 Cycles)	25.0	53.9	**27.7**	39.1	46.1	**40.1**	32.1	37.5	**32.8**	**45.8**
RoBERTa (Omelianchuk et al., 2020)	24.8	52.4	27.4	38.2	44.1	39.0	33.9	39.9	34.8	46.3
+ Ours (1 Cycle)	24.9	52.7	**27.5**	38.7	44.5	**39.5**	34.3	40.3	**35.2**	**46.5**
BERT (Omelianchuk et al., 2020)	23.1	50.2	25.7	35.6	42.9	37.4	33.2	39.4	34.2	45.7
+ Ours (1 Cycle)	23.4	51.2	**26.0**	37.3	41.8	**38.3**	33.7	40.3	**34.7**	**45.9**
XLNet (Omelianchuk et al., 2020)	25.7	54.6	28.4	38.9	46.6	40.1	36.5	44.9	37.7	47.3
+ Ours (1 Cycle)	25.8	54.8	**28.6**	39.0	46.6	**40.1**	36.6	44.9	**37.9**	**47.5**
LM-Critic (Yasunaga et al., 2021)	18.6	39.0	20.5	34.5	35.9	34.5	23.6	24.7	23.5	41.1
+ Ours (2 Cycles)	24.6	52.1	**27.2**	41.1	46.1	**41.8**	31.9	36.2	**32.5**	**46.1**

Table 3: The average of evaluation results on four attack sets (i.e., each test set corresponds to four variants for attack), where the detailed evaluation results against attack sets are given in the Appendix . We also give the cycle times for each model. The blackened fonts indicate the optimal performance of each comparison.

method. We first compare the defense capability of CSA with a recently proposed defense method for the GEC task (Wan et al., 2020). We then conduct experiments to study the effects of self-augmenting, followed by hyper-parameter analysis and a preliminary study for regularization data. These studies are mainly taken on CoNLL-2014, and its correlated attack set constructed by the Mapping & Rules unless there is a clear explanation. All the experiments are launched on the Transformer.

5.1 Defence Capability Comparison



(a) Antonym (b) Synonym

(c) Mapping & Rules (d) Back-Translation

Figure 2: Comparison between our **CSA** and the adversarial training method on four different attack sets.

To calibrate the capability of our CSA against adversarial attacks, we introduce the Mapping & Rules method (Wan et al., 2020) for comparison. Figure 2 presents the evaluation results on four test sets under the aforementioned four types of adversarial attack. We can clearly observe that our CSA has better defense capability than the baseline model under three types of attack and achieves comparable results under the Mapping & Rules attack, which is also the data augmentation strategy for the baseline model. In other words, our CSA can achieve competitive results with the defense method that uses well-crafted adversarial examples at scale for the same type of attack. For other attacks without specifically designed adversarial training examples, our CSA achieves much better model robustness. These results demonstrate the effectiveness and generalization ability of our CSA.

5.2 The Effects of Self-Augmenting

As mentioned before, there are different strategies during the self-augmenting process. One is to directly use the failed pairs (x, y) from the original training datasets to re-train the model, which serves as the baseline. The other is to utilize the outputs $y^{'}$ from the well-trained GEC model as new ungrammatical input sentences to generate augmenting pairs $(y^{'}, y)$, which is used in our CSA. We implement these two strategies under the same settings, and the results on clean data are reported in table 4. We find that the baseline model can improve GEC

Strategies	#Cycles	#Pairs	CoNLL-2014 (M^2)		
			Prec.	Rec.	F_0.5
$x \mapsto y$	0	625,467	68.9	43.9	61.8
	1	511,006	66.5	51.1	62.6
	2	436,229	65.2	46.3	60.3
$y' \mapsto y$	0	625,467	68.9	43.9	61.8
	1	506,572	67.2	49.4	62.6
	2	263,993	67.3	49.3	62.7

Table 4: Results of two strategies for self-augmenting. The first group refers to the strategy of using failed pairs ($x \mapsto y$) from the original training sets $\mathcal{D}$ to re-train the model. The second group corresponds to our strategy of using the model outputs to construct ($y' \mapsto y$) pairs.

#Cycles	0	1	2	3	4	5
Clean Data	61.8	56.2	57.0	56.5	57.1	55.2
+ Ours	61.8	62.5	62.6	62.7	62.9	62.7
Attack Set	36.7	37.4	37.0	36.6	37.0	37.1
+ Ours	36.7	42.2	41.3	41.6	41.8	42.2

Table 5: Comparison between re-training the GEC model on $\mathcal{D} \cup \mathcal{D}_{tune}$ with the same epochs as our CSA.

performance after the first training cycle but will decrease model performance with one more training cycle. As for our introduced strategy in the self-augmenting process, the model performance rises continuously after two training cycles, using fewer pairs than the baseline method. The reason behind this is the baseline method will cause overfitting on the original training datasets by simply re-training the model with part of the same data.

5.3 Hyper-Parameter Analysis

Table 5 presents the results of different training cycles, which correlates with the hyper-parameters of ϵ. We can see that, with the increasing of training cycles, our CSA continuously achieves better performance than the last training cycle on the attack set, i.e., better robustness, with stable performance on the clean data. To explore whether the performance improvement in cycle training is from the introduced small dataset $\mathcal{D}_{tune}$, we train the baseline model on $\mathcal{D} \cup \mathcal{D}_{tune}$ with the same training epochs as our CSA. The dramatic performance decrease of baseline along with the cycle training proves that the performance gains are not brought by $\mathcal{D}_{tune}$ in the cycle training process. Table 6 shows the influence of patience $\mathcal{P}$ to model performance. It can be seen that $\mathcal{P}$=2 is sufficient to achieve competitive performance, which is used as the standard setting in our implementations.

$\mathcal{P}$	CoNLL-2014			CoNLL-2014 (ATK)		
	Prec.	Rec.	F_0.5	Prec.	Rec.	F_0.5
-	67.9	44.1	61.3	34.1	39.7	34.9
2	67.2	49.4	62.6	40.6	44.3	41.3
3	68.1	48.5	63.2	40.3	43.6	40.9
4	68.2	48.9	63.2	40.6	43.7	41.2
5	68.6	48.6	63.4	40.0	43.4	40.7
6	66.2	48.9	61.8	40.2	43.2	40.8

Table 6: The influence of $\mathcal{P}$ to model performance. "**-**" denotes baseline, and (ATK) refers to the attack set.

Reserving Rates (%)	CoNLL-2014 (M^2)			CoNLL-2014 (ATK)		
	P	R	F_0.5	P	R	F_0.5
0 %	68.6	48.6	63.4	40.0	43.4	40.7
25 %	68.5	48.6	63.3	40.1	43.3	40.7
50 %	68.3	48.8	63.2	40.3	43.7	40.9
75 %	68.3	48.5	63.1	40.5	43.8	41.1
100 %	67.5	49.4	62.9	40.9	44.5	41.6

Table 7: The influence of regularization data amount to model performance and defence capability.

5.4 The Influence of Regularization Data

We launch a preliminary experiment to show the correlation between regularization data and the trade-off of model performance and robustness. Table 7 presents the experimental results of removing different proportions of regularization data in the last training cycle. It can be seen that more regularization data can improve the model robustness but suffer performance decreases on the clean data.

6 Conclusion

In this paper, we further study the robustness of advanced GEC models to various types of adversarial attacks and put forward a simple yet very effective cycle self-augmenting method accordingly to improve model robustness. By only utilizing self-augmenting data pairs from a well-learned GEC model and its original training set, our proposed method can improve model performance and robustness without requiring well-crafted adversarial examples at scale for a specific type of adversarial attack. Through leveraging the regularization subset of the self-augmenting data in the cycle training process, our presented method gains additional robustness improvement. Experimental results on seven strong baselines and four benchmark test sets as well as four types of adversarial attacks confirm the effectiveness of our proposed method, which can generalize well to various GEC models with only a few more training epochs as the extra cost.

References

Abhijeet Awasthi, Sunita Sarawagi, Rasna Goyal, Sabyasachi Ghosh, and Vihari Piratla. 2019. Parallel iterative edit models for local sequence transduction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4260–4270.

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

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

Shamil Chollampatt and Hwee Tou Ng. 2018. A multilayer convolutional encoder-decoder neural network for grammatical error correction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

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

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 *Proceedings of the eighth workshop on innovative use of NLP for building educational applications*, pages 22–31.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Xinshuai Dong, Anh Tuan Luu, Rongrong Ji, and Hong Liu. 2021. Towards robustness against natural language word substitutions. *International Conference of Representation Learning*.

Huizhong Duan and Bo-June Hsu. 2011. Online spelling correction for query completion. In *Proceedings of the 20th international conference on World wide web*, pages 117–126.

Mariano Felice, Christopher Bryant, and Ted Briscoe. 2016. Automatic extraction of learner errors in esl sentences using linguistically enhanced alignments. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 825–835.

Tira Nur Fitria. 2021. Grammarly as ai-powered english writing assistant: Students' alternative for writing english. *Metathesis: Journal of English Language, Literature, and Teaching*, 5(1):65–78.

Jianfeng Gao, Xiaolong Li, Daniel Micol, Chris Quirk, and Xu Sun. 2010. A large scale ranker-based system for search query spelling correction. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 358–366.

Tao Ge, Furu Wei, and Ming Zhou. 2018. Fluency boost learning and inference for neural grammatical error correction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1055–1065.

Sylviane Granger. 1998. Prefabricated patterns in advanced efl writing: Collocations and lexical phrases. *Phraseology: Theory, analysis and applications*, pages 145–160.

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

Matthias Hagen, Martin Potthast, Marcel Gohsen, Anja Rathgeber, and Benno Stein. 2017. A large-scale query spelling correction corpus. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1261–1264.

Jianshu Ji, Qinlong Wang, Kristina Toutanova, Yongen Gong, Steven Truong, and Jianfeng Gao. 2017. A nested attention neural hybrid model for grammatical error correction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 753–762.

Marcin Junczys-Dowmunt, Roman Grundkiewicz, Shubha Guha, and Kenneth Heafield. 2018. Approaching neural grammatical error correction as a low-resource machine translation task. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 595–606.

Masahiro Kaneko, Masato Mita, Shun Kiyono, Jun Suzuki, and Kentaro Inui. 2020. Encoder-decoder models can benefit from pre-trained masked language models in grammatical error correction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4248–4254.

Clare-Marie Karat, Christine Halverson, Daniel Horn, and John Karat. 1999. Patterns of entry and correction in large vocabulary continuous speech recognition systems. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pages 568–575.

Satoru Katsumata and Mamoru Komachi. 2020. Stronger baselines for grammatical error correction using a pretrained encoder-decoder model. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 827–832.

Shun Kiyono, Jun Suzuki, Masato Mita, Tomoya Mizumoto, and Kentaro Inui. 2019. An empirical study of incorporating pseudo data into grammatical error correction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1236–1242.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 28–39.

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

Piji Li and Shuming Shi. 2021. Tail-to-tail non-autoregressive sequence prediction for Chinese grammatical error correction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4973–4984.

Zongyi Li, Jianhan Xu, Jiehang Zeng, Linyang Li, Xiaoqing Zheng, Qi Zhang, Kai-Wei Chang, and Cho-Jui Hsieh. 2021. Searching for an effective defender: Benchmarking defense against adversarial word substitution. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3137–3147.

Jared Lichtarge, Chris Alberti, Shankar Kumar, Noam Shazeer, Niki Parmar, and Simon Tong. 2019. Corpora generation for grammatical error correction. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3291–3301.

Edward Ma. 2019. Nlp augmentation. https://github.com/makcedward/nlpaug.

Eric Malmi, Sebastian Krause, Sascha Rothe, Daniil Mirylenka, and Aliaksei Severyn. 2019. Encode, tag, realize: High-precision text editing. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5054–5065.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Tomoya Mizumoto, Mamoru Komachi, Masaaki Nagata, and Yuji Matsumoto. 2011. Mining revision log of language learning sns for automated japanese error correction of second language learners. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 147–155.

Hossein Mobahi, Mehrdad Farajtabar, and Peter L Bartlett. 2020. Self-distillation amplifies regularization in hilbert space. *arXiv preprint arXiv:2002.05715*.

Mahdi Namazifar, John Malik, Li Erran Li, Gokhan Tur, and Dilek Hakkani Tür. 2021. Correcting Automated and Manual Speech Transcription Errors Using Warped Language Models. In *Proc. Interspeech 2021*, pages 2037–2041.

Jakub Náplava and Milan Straka. 2019. Grammatical error correction in low-resource scenarios. In *Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT 2019)*, pages 346–356.

Courtney Napoles, Maria Nădejde, and Joel Tetreault. 2019. Enabling robust grammatical error correction in new domains: Data sets, metrics, and analyses. *Transactions of the Association for Computational Linguistics*, 7:551–566.

Courtney Napoles, Keisuke Sakaguchi, Matt Post, and Joel Tetreault. 2015. Ground truth for grammatical error correction metrics. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 588–593.

Courtney Napoles, Keisuke Sakaguchi, Matt Post, and Joel Tetreault. 2016. Gleu without tuning. *arXiv preprint arXiv:1605.02592*.

Courtney Napoles, Keisuke Sakaguchi, and Joel Tetreault. 2017. Jfleg: A fluency corpus and benchmark for grammatical error correction. *EACL 2017*, page 229.

Hwee Tou Ng, Siew Mei Wu, Ted Briscoe, Christian Hadiwinoto, Raymond Hendy Susanto, and Christopher Bryant. 2014. The conll-2014 shared task on grammatical error correction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–14.

University of Hertfordshire. 2007. Ppdb: Pesticide properties database.

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

Xin Sun, Tao Ge, Furu Wei, and Houfeng Wang. 2021. Instantaneous grammatical error correction with shallow aggressive decoding. *arXiv preprint arXiv:2106.04970*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6000–6010.

Zhaohong Wan, Xiaojun Wan, and Wenguang Wang. 2020. Improving grammatical error correction with data augmentation by editing latent representation. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2202–2212.

Lihao Wang and Xiaoqing Zheng. 2020. Improving grammatical error correction models with purpose-built adversarial examples. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2858–2869.

Xiaoqiang Wang, Yanqing Liu, Sheng Zhao, and Jinyu Li. 2021. A Light-Weight Contextual Spelling Correction Model for Customizing Transducer-Based Speech Recognition Systems. In *Proc. Interspeech 2021*, pages 1982–1986.

Yu Wang, Yuelin Wang, Jie Liu, and Zhuo Liu. 2020. A comprehensive survey of grammar error correction. *arXiv preprint arXiv:2005.06600*.

Ziang Xie, Anand Avati, Naveen Arivazhagan, Dan Jurafsky, and Andrew Y Ng. 2016. Neural language correction with character-based attention. *arXiv preprint arXiv:1603.09727*.

Ziang Xie, Guillaume Genthial, Stanley Xie, Andrew Y Ng, and Dan Jurafsky. 2018. Noising and denoising natural language: Diverse backtranslation for grammar correction. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 619–628.

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

Helen Yannakoudakis, Ted Briscoe, and Ben Medlock. 2011. A new dataset and method for automatically grading esol texts. In *Proceedings of the 49th annual meeting of the association for computational linguistics: human language technologies*, pages 180–189.

Michihiro Yasunaga, Jure Leskovec, and Percy Liang. 2021. Lm-critic: Language models for unsupervised grammatical error correction. *arXiv preprint arXiv:2109.06822*.

Michihiro Yasunaga and Percy Liang. 2021. Break-it-fix-it: Unsupervised learning for program repair. *arXiv preprint arXiv:2106.06600*.

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

Shuai Zhang, Jiangyan Yi, Zhengkun Tian, Ye Bai, Jianhua Tao, Xuefei Liu, and Zhengqi Wen. 2021. End-to-End Spelling Correction Conditioned on Acoustic Feature for Code-Switching Speech Recognition. In *Proc. Interspeech 2021*, pages 266–270.

Wei Zhao, Liang Wang, Kewei Shen, Ruoyu Jia, and Jingming Liu. 2019. Improving grammatical error correction via pre-training a copy-augmented architecture with unlabeled data. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 156–165.

Yun Zhao, Xuerui Yang, Jinchao Wang, Yongyu Gao, Chao Yan, and Yuanfu Zhou. 2021. BART Based Semantic Correction for Mandarin Automatic Speech Recognition System. In *Proc. Interspeech 2021*, pages 2017–2021.

Helong Zhou, Liangchen Song, Jiajie Chen, Ye Zhou, Guoli Wang, Junsong Yuan, and Qian Zhang. 2021. Rethinking soft labels for knowledge distillation: A bias-variance tradeoff perspective. *arXiv preprint arXiv:2102.00650*.

Wangchunshu Zhou, Tao Ge, Chang Mu, Ke Xu, Furu Wei, and Ming Zhou. 2020. Improving grammatical error correction with machine translation pairs. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 318–328.

Shengyao Zhuang and Guido Zuccon. 2021. Dealing with typos for bert-based passage retrieval and ranking. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 2836–2842.

Examples of Regularization Data	
Poor:	I thought it was a very good idea .
Reg:	I thought it was a very good idea .
Good:	I think it was a very good idea .
Poor:	I hope you 'll attend my suggestions .
Reg:	I hope you 'll follow my suggestions .
Good:	I hope you 'll act on my suggestions .
Poor:	I was very frethend , but I knew , that I should do something .
Reg:	I was very free , but I knew that I should do something .
Good:	I was very frightened , but I knew that I had to do something .

Table 8: Examples of regularization data, where *Poor*, *Reg* and *Good* represent for ungrammatical sentence, regularization data and grammatical sentence respectively. We set patience $\mathcal{P} = 10$ and store augmenting dataset in each cycle. Finally, we sample the regularization data from the intersection of those stored datasets. We summarize three main types of regularization data. (a) The golden sentences are mislabeled as shown in the first group of the table. (b) The augmenting data have a similar meaning with the golden sentences as shown in the second group of the table. (c) The augmenting data are grammatically correct but lack context as shown in the third group of the table.

Model	**BEA** (ERRANT)			**CoNLL** (M^2)			**FCE** (M^2)			**JFLEG**
	Prec.	**Rec.**	**F_0.5**	**Prec.**	**Rec.**	**F_0.5**	**Prec.**	**Rec.**	**F_0.5**	**GLEU**
Transformer (Kiyono et al., 2019)	24.3	57.1	27.4	33.0	48.9	35.3	31.3	42.7	33.1	46.3
+ Ours	25.3	57.9	**28.4**	33.7	49.8	**36.0**	31.9	42.9	**33.6**	**46.8**
Bert-fuse (Kaneko et al., 2020)	24.2	56.6	27.3	32.6	48.2	34.9	31.8	43.1	33.6	46.6
+ Ours	25.2	58.6	**28.5**	33.7	50.3	**36.1**	32.5	44.5	**34.4**	**47.1**
BART (Katsumata and Komachi, 2020)	24.2	57.2	27.4	33.2	48.9	35.5	31.2	42.0	32.9	46.3
+ Ours	25.4	59.8	**28.7**	34.2	51.2	**36.6**	31.9	43.7	**33.7**	**46.6**
RoBERTa (Omelianchuk et al., 2020)	25.4	59.3	28.7	34.4	51.1	36.8	32.7	45.1	34.6	46.5
+ Ours	25.5	59.4	**28.8**	34.6	51.3	**37.0**	33.1	45.1	**34.9**	**46.7**
BERT (Omelianchuk et al., 2020)	24.7	58.8	28.0	33.5	50.2	35.9	32.5	45.0	34.5	46.7
+ Ours	24.9	58.7	**28.2**	33.8	50.6	**36.2**	32.8	45.7	**34.8**	**46.8**
XLNet (Omelianchuk et al., 2020)	25.8	60.0	29.1	34.4	51.8	36.9	33.8	47.5	35.8	47.5
+ Ours	25.7	60.8	**29.2**	34.6	52.0	**37.1**	34.1	47.3	**36.1**	**47.8**
lm (Yasunaga et al., 2021)	23.9	56.7	27.0	32.9	49.1	35.3	30.2	41.7	32.0	46.0
+ Ours	25.0	58.8	**28.3**	34.2	50.6	**36.5**	32.1	44.2	**34.0**	**47.1**

Table 9: Evaluation results on *Back-Translation* attack sets. We generate the attack sets by using *Back-Translation* method as aforementioned.

Model	BEA (ERRANT)			CoNLL (M^2)			FCE (M^2)			JFLEG
	Prec.	**Rec.**	**F_0.5**	**Prec.**	**Rec.**	**F_0.5**	**Prec.**	**Rec.**	**F_0.5**	**GLEU**
Transformer (Kiyono et al., 2019)	8.4	37.3	9.9	36.5	37.8	36.7	19.7	27.1	20.8	37.7
+ Ours	11.0	44.2	**12.9**	41.3	45.9	**42.1**	22.8	32.5	**24.3**	**38.6**
Bert-fuse (Kaneko et al., 2020)	7.9	36.0	9.3	36.7	37.5	36.9	19.9	27.5	21.1	37.4
+ Ours	11.3	45.5	**13.3**	41.5	45.0	**42.1**	23.9	34.3	**25.4**	**38.6**
BART (Katsumata and Komachi, 2020)	8.1	34.8	9.6	36.7	37.4	36.8	18.7	24.9	19.7	35.8
+ Ours	12.1	46.65	**14.2**	42.3	45.7	**43.0**	22.8	32.7	**24.3**	**37.7**
RoBERTa (Omelianchuk et al., 2020)	11.8	43.8	13.8	41.3	42.6	41.6	24.5	34.2	26.0	40.0
+ Ours	12.1	44.5	**14.2**	42.0	43.3	**42.2**	24.6	34.6	**26.1**	**40.2**
BERT (Omelianchuk et al., 2020)	10.8	40.9	12.7	35.6	41.6	39.9	23.9	34.0	25.4	39.0
+ Ours	11.4	42.5	**13.4**	40.3	43.2	**40.8**	24.7	34.8	**26.2**	**39.2**
XLNet (Omelianchuk et al., 2020)	13.2	46.8	15.4	42.1	45.3	42.7	27.4	39.3	29.1	40.7
+ Ours	13.4	47.3	**15.7**	42.2	45.9	**42.9**	27.4	39.4	**29.2**	**40.8**
lm (Yasunaga et al., 2021)	6.4	26.7	7.6	33.1	30.8	32.6	14.0	17.1	14.5	34.4
+ Ours	11.4	43.1	**13.3**	40.7	43.9	**41.3**	21.9	29.8	**23.1**	**38.6**

Table 10: Evaluation results on *Mapping & Rules* attack sets. We generate the attack sets by using *Mapping & Rules* method as aforementioned.

Model	BEA (ERRANT)			CoNLL (M^2)			FCE (M^2)			JFLEG
	Prec.	**Rec.**	**F_0.5**	**Prec.**	**Rec.**	**F_0.5**	**Prec.**	**Rec.**	**F_0.5**	**GLEU**
Transformer (Kiyono et al., 2019)	35.3	53.4	37.8	42.9	39.0	42.0	41.5	36.0	40.3	53.7
+ Ours	39.7	59.9	**42.6**	46.1	45.5	**46.0**	46.3	44.3	**46.0**	**55.4**
Bert-fuse (Kaneko et al., 2020)	34.6	50.3	36.9	41.8	36.2	40.5	46.1	37.5	44.1	53.5
+ Ours	39.4	59.5	**42.3**	46.5	45.8	**46.3**	48.3	44.0	**47.4**	**56.3**
BART (Katsumata and Komachi, 2020)	36.0	47.3	37.8	44.9	37.6	43.2	45.2	32.0	41.7	51.2
+ Ours	42.0	58.7	**44.5**	48.8	45.7	**48.1**	44.8	38.9	**43.5**	**54.2**
RoBERTa (Omelianchuk et al., 2020)	42.1	57.0	44.4	48.1	43.8	47.2	47.8	42.8	46.7	53.9
+ Ours	42.0	57.4	**44.4**	48.4	44.0	**47.5**	48.2	43.5	**47.2**	**54.3**
BERT (Omelianchuk et al., 2020)	38.8	54.5	41.1	45.9	42.3	45.1	46.6	41.9	45.6	53.3
+ Ours	38.7	55.9	**41.3**	46.5	44.1	**46.0**	46.6	43.0	**45.9**	**53.6**
XLNet (Omelianchuk et al., 2020)	43.0	60.3	45.6	48.3	47.1	48.1	50.6	49.5	50.4	55.4
+ Ours	43.1	60.3	**45.7**	48.4	46.7	**48.1**	50.8	49.5	**50.5**	**55.5**
lm (Yasunaga et al., 2021)	32.3	41.3	33.8	38.9	32.3	37.3	32.2	21.7	29.4	46.4
+ Ours	42.2	57.5	**44.6**	48.7	45.8	**48.0**	45.5	38.1	**43.8**	**54.5**

Table 11: Evaluation results on *Antonym Substitutions* attack sets. We generate the attack sets by using *Antonym Substitutions* method as aforementioned.

Model	BEA (ERRANT)			CoNLL (M^2)			FCE (M^2)			JELEG
	Prec.	**Rec.**	**F_0.5**	**Prec.**	**Rec.**	**F_0.5**	**Prec.**	**Rec.**	**F_0.5**	**GLEU**
Transformer (Kiyono et al., 2019)	15.9	44.3	18.3	24.1	33.1	25.5	26.3	31.0	27.1	43.8
+ Ours	19.3	50.7	**22.1**	29.6	40.6	**31.3**	28.8	35.3	**29.9**	**45.1**
Bert-fuse (Kaneko et al., 2020)	14.8	41.4	17.0	22.7	31.0	24.0	26.2	29.8	26.9	44.0
+ Ours	19.3	51.1	**22.1**	29.7	40.9	**31.4**	30.1	37.0	**31.3**	**45.8**
BART (Katsumata and Komachi, 2020)	15.1	39.4	17.3	23.1	31.2	24.4	25.1	27.0	25.5	41.8
+ Ours	20.5	50.5	**23.3**	30.9	41.8	**32.6**	28.8	34.5	**29.8**	**44.5**
RoBERTa (Omelianchuk et al., 2020)	19.8	49.3	22.5	28.9	38.8	30.5	30.7	37.3	31.8	44.6
+ Ours	19.9	49.4	**22.6**	29.6	39.3	**31.1**	31.2	38.1	**32.4**	**44.8**
BERT (Omelianchuk et al., 2020)	18.2	46.7	20.8	27.3	37.3	28.8	29.9	36.6	31.1	43.6
+ Ours	18.6	47.8	**21.2**	28.7	29.3	**30.3**	30.5	37.6	**31.7**	**43.8**
XLNet (Omelianchuk et al., 2020)	20.8	51.1	23.6	30.7	42.0	32.5	34.0	43.2	35.5	45.7
+ Ours	20.8	50.9	**23.6**	30.7	41.9	**32.4**	34.1	43.2	**35.6**	**45.9**
lm (Yasunaga et al., 2021)	11.7	31.4	13.4	33.0	31.3	32.7	17.8	18.4	17.9	37.7
+ Ours	19.7	49.1	**22.4**	40.7	43.9	**41.3**	28.1	32.7	**29.0**	**44.2**

Table 12: Evaluation results on *Synonyms Substitutions* attack sets. We generate the attack sets by using *Synonyms Substitutions* method as aforementioned.