Type-Driven Multi-Turn Corrections for Grammatical Error Correction

Shaopeng Lai[1]*, **Qingyu Zhou**[2], **Jiali Zeng**[2], **Zhongli Li**[2],
Chao Li[2], **Yunbo Cao**[2], **Jinsong Su**[1,3]†
[1]School of Informatics, Xiamen University, China
[2]Tencent Cloud Xiaowei, China
[3]Key Laboratory of Digital Protection and Intelligent Processing of Intangible Cultural Heritage of Fujian and Taiwan, Ministry of Culture and Tourism, China
splai@stu.xmu.edu.cn, {qingyuzhou, lemonzeng, neutrali, diegoli, yunbocao}@tencent.com, jssu@xmu.edu.cn

Abstract

Grammatical Error Correction (GEC) aims to automatically detect and correct grammatical errors. In this aspect, dominant models are trained by one-iteration learning while performing multiple iterations of corrections during inference. Previous studies mainly focus on the data augmentation approach to combat the exposure bias, which suffers from two drawbacks. First, they simply mix additionally-constructed training instances and original ones to train models, which fails to help models be explicitly aware of the procedure of gradual corrections. Second, they ignore the interdependence between different types of corrections. In this paper, we propose a *Type-Driven Multi-Turn Corrections* approach for GEC. Using this approach, from each training instance, we additionally construct multiple training instances, each of which involves the correction of a specific type of errors. Then, we use these additionally-constructed training instances and the original one to train the model in turn. By doing so, our model is trained to not only correct errors progressively, but also exploit the interdependence between different types of errors for better performance. Experimental results and in-depth analysis show that our approach significantly benefits the model training. Particularly, our enhanced model achieves state-of-the-art single-model performance on English GEC benchmarks. We release our code at https://github.com/DeepLearnXMU/TMTC.

1 Introduction

Grammatical Error Correction (GEC) aims at automatically detecting and correcting grammatical (and other related) errors in a text. It attracts much attention due to its practical applications in writing assistant (Napoles et al., 2017b; Ghufron and Rosyida, 2018), speech recognition systems (Karat et al., 1999; Wang et al., 2020; Kubis et al., 2020) etc. Inspired by the success of neural machine translation (NMT), some models adopt the same paradigm, namely NMT-based models. They have been quite successful, especially with data augmentation approach (Boyd, 2018; Ge et al., 2018; Xu et al., 2019; Grundkiewicz et al., 2019; Wang and Zheng, 2020; Takahashi et al., 2020). However, these models have been blamed for their inefficiency during inference (Chen et al., 2020; Sun et al., 2021). To tackle this issue, many researchers resort to the sequence-to-label (Seq2Label) formulation, achieving comparable or better performance with efficiency (Malmi et al., 2019; Awasthi et al., 2019; Stahlberg and Kumar, 2020; Omelianchuk et al., 2020).

Despite their success, both NMT-based and Seq2Label models are trained by one-iteration learning, while correcting errors for multiple iterations during inference. As a consequence, they suffer from exposure bias and exhibit performance degrade (Ge et al., 2018; Lichtarge et al., 2019; Zhao and Wang, 2020; Parnow et al., 2021). To deal with this issue, Ge et al. (2018) propose to generate fluency-boost pseudo instances as additional training data. Besides, Parnow et al. (2021) dynamically augment training data by introducing the predicted sentences with high error probabilities.

However, the above-mentioned approaches construct pseudo data based on a GEC model or an error-generation model, which extremely depends on the performance of these models. Thus, the error distribution of pseudo data is biased and lacks diversity and practicality. Moreover, they simply mix original and pseudo data to train models, which are unable to learn correcting errors progressively. Furthermore, they ignore the interdependence between different types of errors, which intuitively plays an important role on GEC. Taking Table 1 as example, correcting "*little*" with "*few*" or "*job*" with "*jobs*"

* Work is done during internship at Tencent Cloud Xiaowei
† Corresponding author

Findings of the Association for Computational Linguistics: ACL 2022, pages 3225 - 3236

Erroneous Sentence: *In my country there are **little job** because the economy is very bad .*
Reference Sentence: *In my country there are **few jobs** because the economy is very bad .*

Table 1: An example for the interdependence between corrections. Please note that whichever error is corrected first, the other error can be corrected more easily.

first can help the other error be better corrected. Therefore, we believe that how to construct and exploit pseudo data with editing-action corrections for GEC is still a problem worthy of in-depth study.

In this paper, we first conduct quantitative experiments to investigate the performance improvements of GEC model with providing different types of error corrections. Experimental results show that corrections of appending or replacing words first indeed benefit the corrections of other errors. Furthermore, we propose a **T**ype-Driven **M**ulti-**T**urn **C**orrections (TMTC) approach for GEC. Concretely, by correcting a certain type of errors with others unchanged, we construct an intermediate sentence for each training instance and pair it with its raw erroneous sentence and reference sentence respectively, forming two additional training instances. During the model training, using the former instance, we firstly guide the model to learn correcting the corresponding type of errors. Then, using the latter instance, we teach the model to correct other types of errors with the help of previous corrections. Overall, contributions of our work are three-fold:

- Through quantitative experiments, we investigate the interdependence between different types of corrections, with the finding that corrections of appending or replacing words significantly benefit correcting other errors.
- We propose a TMTC approach for GEC. To the best of our knowledge, our work is the first attempt to explore the interdependence between different types of errors for GEC.
- We conduct experiments and in-depth analysis to investigate the effectiveness of our proposed approach. Experimental results show that our enhanced model achieves the state-of-the-art performance.

2 Related Work

Generally, there are two categories of models in GEC: Transformer-dominant NMT-based models (Boyd, 2018; Ge et al., 2018; Xu et al., 2019; Grundkiewicz et al., 2019; Wang and Zheng, 2020; Takahashi et al., 2020) and GECToR-leading Seq2Label models (Malmi et al., 2019; Awasthi et al., 2019; Stahlberg and Kumar, 2020; Omelianchuk et al., 2020). The former models consider GEC as a machine translation task, where the model is fed with the erroneous sentence and then output the corrected sentence token by token. By comparison, Seq2Label models are able to correct grammatical errors more efficiently and even better. Among them, the GECToR models (Omelianchuk et al., 2020) obtain remarkable performance. Typically, they adopt a pre-trained language model as the encoder to learn word-level representations and utilize a softmax-based classifier to predict designed editing-action labels.

Since GEC models may fail to completely correct a sentence through just one-iteration inference, some researchers resort to data augmentation that has been widely used in other NLP studies (Song et al., 2020; Xu et al., 2020). For instance, Ge et al. (2018) propose to let the GEC model infer iteratively and design a fluency boost learning approach. Specifically, they establish new erroneous-reference sentence pairs by pairing predicted less fluent sentences with their reference sentences during training. Likewise, to solve the mismatches between training and inference of Seq2Label models, Parnow et al. (2021) apply a confidence-based method to construct additional training data by pairing low-confidence sentences with reference sentences. Note that these two methods also involve constructing pseudo data using sentences with partial errors. However, ours is still different from them in two aspects. First, these two methods simply mix their pseudo data with original data to still train models in a one-iteration learning manner. By contrast, we decompose the one-iteration corrections into multiple turns, so as to make the model aware of gradual corrections. Second, these two methods ignore the interdependence between different types of errors, which is exploited by our proposed approach to enhance the model.

3 Background

In this work, we choose GECToR (Omelianchuk et al., 2020) as our basic GEC model due to its efficiency and competitive performance. Typically, it considers the GEC task as a sequence-to-label task, where the candidate editing-action la-

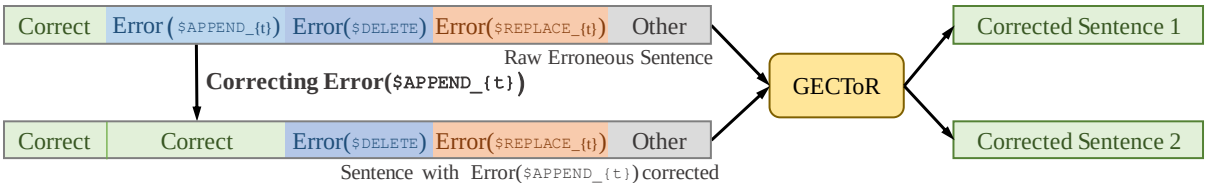

Figure 1: The procedure of our quantitative experiments. Each sentence is composed of five parts as illustrated, where Error(ACTION *label*) denote the erroneous words that can be corrected via corresponding editing-action label. We only correct one type of errors and compare the prediction results of other types of errors.

bels mainly include $KEEP (to keep the current word unchanged), $DELETE (to delete the current word), $APPEND_{t} (to append the word t after the current word), $REPLACE_{t} (to replace the current word with the word t) and some elaborate g-transformation labels (Omelianchuk et al., 2020) performing task-specific operations, such as $TRANSFORM_CASE_LOWER and $TRANSFORM_CASE_CAPITAL (to change the case of the current word).

On the whole, the GECToR model is composed of an encoder based on pre-trained language model and two linear classifiers: one for grammatical error detection (GED) and the other for GEC. The encoder reads the erroneous sentence X_e=$x_1, x_2, ..., x_N$ and represent words with hidden states $\{h_i\}_{i=1}^{N}$, which are fed into classifiers to predict the binary label sequence Y=$y_1, y_2, ..., y_N$ for GED and the editing-action label sequence T=$t_1, t_2, ..., t_N$ for GEC, respectively. Formally, the losses of two classifiers can be formulated as

$$L_d = -\sum_{i=1}^{N} \log p(y_i|X_e, \theta), \quad (1)$$

$$L_c = -\sum_{i=1}^{N} \log p(t_i|X_e, \theta), \quad (2)$$

where θ denotes model parameters. Usually, the GECToR model is trained to optimize the sum of two losses: L=L_d+L_c.

It is worth noting that the GECToR model is trained to correct all errors in a one-iteration manner, while correcting errors in a multiple-iteration way during inference (at most 5 iterations). Besides, there are three stages involved during the training of the GECToR model, as shown in Table 2.

4 Effect of the Interdependence between Different Types of Corrections

In this section, we conduct several groups of quantitative experiments to explore the interdependence between corrections.

Dataset	#Instance	Stage
PIE-synthetic (Awasthi et al., 2019)	9,000,000	I
Lang-8 (Tajiri et al., 2012)	947,344	II
NUCLE (Dahlmeier et al., 2013)	56,958	II
FCE (Yannakoudakis et al., 2011)	34,490	II
W&I+LOCNESS (Bryant et al., 2019)	34,304	II, III

Table 2: GECToR is trained on PIE-synthetic dataset for pre-training at Stage I. Then, it is fine-tuned on Lang-8, NUCLE, FCE, W&I+LOCNESS at Stage II. At Stage III, the final fine-tuning is conducted on W&I+LOCNESS.

We first train the GECToR model on *Stage II Only* for efficiency. All training settings are the same to published parameters.[1] Afterwards, we use the model to conduct corrections on the BEA-2019 (W&I+LOCNESS) dev set and CoNLL-2014 test set (Ng et al., 2014) and their variants with some errors corrected manually. For simplicity, we only consider the three most frequent editing-action labels: $APPEND_{t}, $DELETE and $REPLACE_{t}.

Figure 1 shows the procedure of quantitative experiments. Specifically, we separate each raw erroneous sentence into five parts: correct words, erroneous words that can be corrected by $APPEND_{t}/$DELETE/$REPLACE_{t}, and words with other types of errors. If we want to investigate the influence of $APPEND_{t}, we first select the data containing $APPEND_{t} labels and denote them as D(APPEND). Then we manually correct all the errors which should be corrected by $APPEND_{t} labels, obtaining the new subset D(APPEND✓). Afterwards, we use our model to correct erroneous sentences of subsets D(APPEND) and D(APPEND✓) for just one iteration, and finally we only evaluate and compare the model performance on the predictions of

[1]We use the codes of new version from https://github.com/grammarly/gector/pull/120 after contacting authors.

Dataset	Evaluation	RoBERTa							
		BEA-2019 (dev)				CoNLL-2014 (test)			
		Num.	Prec.	Rec.	F_1	Num.	Prec.	Rec.	F_1
Original Dataset	$APPEND_{t}	2609	53.43	35.22	42.46	621	27.46	23.35	25.24
	$DELETE	1403	56.04	23.81	33.42	1115	51.89	18.48	27.25
	$REPLACE_{t}	3495	50.87	23.32	31.98	1398	38.57	18.45	24.96
D(APPEND)	$DELETE	904	62.63	20.02	30.34	496	47.52	13.51	21.04
	$REPLACE_{t}	2079	49.71	20.30	28.83	660	28.57	11.21	16.10
D(APPEND✓)	$DELETE	904	68.84	26.88	38.66 (+8.32)	496	59.06	17.74	27.29 (+6.22)
	$REPLACE_{t}	2079	67.46	36.99	47.78 (+18.95)	660	48.96	28.64	36.14 (+20.04)
D(DELETE)	$APPEND_{t}	1024	52.69	25.78	34.62	332	18.93	13.86	16.00
	$REPLACE_{t}	1425	50.91	19.72	28.43	716	30.89	13.55	18.83
D(DELETE✓)	$APPEND_{t}	1024	57.14	27.73	37.34 (+2.72)	332	22.77	15.36	18.35 (+2.35)
	$REPLACE_{t}	1425	55.02	22.32	31.75 (+4.32)	716	36.17	16.62	22.78 (+3.95)
D(REPLACE)	$APPEND_{t}	1762	52.76	29.34	37.71	443	23.92	18.74	21.01
	$DELETE	996	56.19	18.67	28.03	767	47.10	15.91	23.78
D(REPLACE✓)	$APPEND_{t}	1762	68.05	49.21	57.11 (+19.40)	443	41.97	44.24	43.08 (+22.07)
	$DELETE	996	69.33	34.04	45.66 (+17.63)	767	61.08	25.16	35.64 (+11.86)

Table 3: Results of our quantitative experiments. D(ACTION) denotes a subset consisting of instances with ACTION label. D(ACTION✓) denotes another version of D(ACTION), where corresponding errors have been manually corrected.

$DELETE and $REPLACE_{t}. For example, by comparing the model performance with respect to the $DELETE label, we can draw the conclusion that appending some words first could help the model to achieve better predictions on $DELETE .

Likewise, we conduct experiments with respect to $DELETE and $REPLACE_{t} labels. Besides, we evaluate the performance for each type of labels on the raw dataset without any constraints. Experimental results of the RoBERTa-based GECToR model (Liu et al., 2019) are listed in Table 3. We can observe that the consistent performance improvements occur on both the W&I+LOCNESS dev set and the CoNLL-2014 test set, no matter which type of errors are corrected first. Moreover, it is surprising that if replacing words or appending words are conducted beforehand, the model performance is significantly improved on correcting other types of errors. Meanwhile, deleting words does not benefit others compared with other two kinds of corrections.

We also notice that the model improvements are positively associated with the number of manual corrections on the BEA-2019 dev set. However, the performance improvements on the CoNLL-2014 test set is not closely related to the number of manual corrections. Thus, we can conclude that the interdependence between different types of corrections indeed plays a more important role than the number of corrections on performance improvements. Having witnessed these experimental results, we can arrive at the following two conclusions:

- GEC models can better deal with errors when some types of errors have been corrected.
- Corrections of appending words or replacing words help the model correct other types of errors more than deleting words.

Please note that we also conduct experiments using the XLNet-based GECToR model (Yang et al., 2019). Similar trend can be observed from experimental results reported in Appendix §A.1.

5 Our Approach

In this section, we introduce our proposed Type-Driven Multi-Turn Corrections (TMTC) approach in detail. As concluded above, correcting certain types of errors first benefits correcting others, thus, we decompose one-iteration corrections of each training instance into multi-turn corrections, so as to make the trained model to learn performing corrections progressively.

The key step of our approach is to construct an intermediate sentence for each training instance. Formally, each training instance is a sentence pair (X_e, X_c) consisting an erroneous sentence X_e and a reference sentence X_c. To construct its intermediate sentence X', we randomly select partial grammatical errors and correct them manually while keeping others unchanged. Then, X' is paired with X_e and X_c to generate two new pairs: (X_e, X') and (X', X_c), respectively. Figure 2 illustrates an example of constructing two additional training instances from a sentence pair. In this example, for

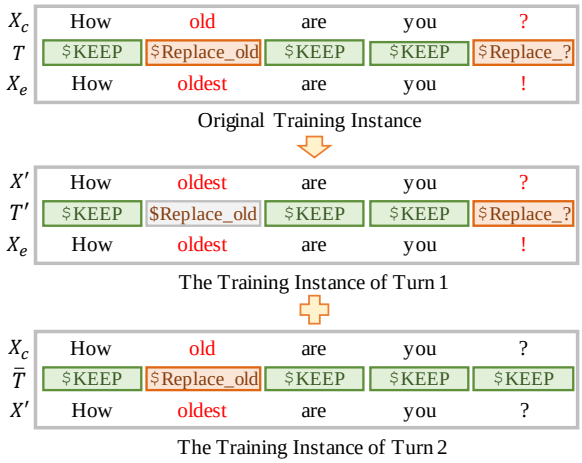

Figure 2: The procedure illustration of constructing additional training instances. Here, we construct an intermediate sentence X', which is paired with the raw erroneous sentence X_e and reference sentence X_c to form two additional training instances (X_e, X') and (X', X_c), respectively. Red squares mean labels correcting errors, while green ones mean the labels to keeping the current word unchanged. Losses of gray squares will be omitted in the first turn.

the erroneous sentence with two grammatical errors "*oldest*" and "*!*", we correct "*!*" by "*?*" manually to form the semi-corrected sentence "*How oldest are you ?*". It should be noted that our constructed training instances are derived from the original training corpus, and thus their grammatical errors are also human-making.

Based on the above findings mentioned in Section §4, we apply our approach to design three training strategies: **APPEND-first**, **DELETE-first** and **REPLACE-first**. Here, the ACTION-first strategy means that the model is trained to learn ACTION corrections in the first turn and then the others in the second turn. For example, when using the DELETE-first strategy, we keep the errors with "$DELETE" as target labels unchanged during the constructions of intermediate sentences. Using additionally-constructed training instances involving these sentences, the trained model will be encouraged to focus on performing corrections first via $DELETE. Table 4 lists the numbers of additionally-constructed training instances using these strategies. According to our findings concluded in Section §4, the models trained using APPEND-first and REPLACE-first strategies should perform better.

Using our approach, we adopt different objectives to successively train our model. Specifically,

Strategy	#Additional Instance
RANDOM	367,814
APPEND-first	311,348
DELETE-first	326,100
REPLACE-first	296,683

Table 4: Numbers of additionally-constructed training instances. We also explore the training strategy that randomly corrects partial errors first. For convenience, we name this training strategy as RANDOM.

we define the following training objectives $L_c^{(1)}$ and $L_c^{(2)}$ in the first and second turns, respectively:

$$L_c^{(1)} = -\sum_{i=1}^{N} \mathbb{1}(t'_i = t_i) \cdot \log p(t'_i | X_e, \theta), \quad (3)$$

$$L_c^{(2)} = -\sum_{i=1}^{\bar{N}} \log p(\bar{t}_i | X', \theta), \quad (4)$$

where $\{t'_i\}_{i=1}^{N}$ and $\{\bar{t}_i\}_{i=1}^{\bar{N}}$ are the editing-action label sequences of additionally-constructed training instances (X_e, X') and (X', X_c) respectively.

Notably, there remain some grammatical errors within intermediate sentences which not be learned by the model in the first turn. Therefore, we omit the incorrect supervisal signals in the definition of $L_c^{(1)}$ via an indicator function $\mathbb{1}(*)$, which is used to shield the effect of incorrect losses. However, because our additionally-constructed training instances contain less grammatical errors compared with original ones, which causes the trained model to correct less errors. To address this defect, we still use the original training instances to continuously train model in the third turn.

Formally, we finally, we use all training instances to continuously train our model with the following objective L'=$L_c^{(1)}$+$L_c^{(2)}$+L. Our experimental results presented in Section §6 show that our additionally-constructed training instances and original ones are complementary to each other.

6 Experiment

6.1 Setup

To ensure fair comparison, we train the GECToR models using the same training datasets and parameters as (Omelianchuk et al., 2020), and then evaluate them on the BEA-2019 (W&I+LOCNESS) dev, test set and the CoNLL-2014 test set. The details of the training data are listed in Table 2. Following (Omelianchuk et al., 2020), we conduct

Model	Pre-trained	BEA-2019 (dev)			CoNLL-2014 (test)		
		Prec.	Rec.	$F_{0.5}$	Prec.	Rec.	$F_{0.5}$
GECToR(Omelianchuk et al., 2020)†	RoBERTa	50.30	30.50	44.50	67.50	38.30	58.60
	XLNet	47.10	34.20	43.80	64.60	42.60	58.50
GECToR	RoBERTa	49.80	37.61	46.77	66.56	45.08	60.77
	XLNet	45.55	39.81	44.27	64.04	48.67	60.24
GECToR(RANDOM)	RoBERTa	52.88	36.05	48.37 (+1.60)	69.54	44.32	62.43 (+1.66)
GECToR(APPEND-first)	RoBERTa	54.92	35.30	**49.43** (+2.66)	70.73	43.88	**63.01** (+2.24)
GECToR(DELETE-first)	RoBERTa	53.85	35.13	48.67 (+1.90)	70.57	42.78	62.45 (+1.68)
GECToR(REPLACE-first)	RoBERTa	54.78	34.82	49.14 (+2.37)	70.2	43.92	62.70 (+1.93)
GECToR(RANDOM)	XLNet	49.74	38.47	46.99 (+2.72)	67.41	46.68	61.91 (+1.67)
GECToR(APPEND-first)	XLNet	51.10	37.72	47.71 (+3.44)	67.74	46.39	62.03 (+1.79)
GECToR(DELETE-first)	XLNet	50.48	37.49	47.21 (+2.97)	67.33	46.42	61.79 (+1.55)
GECToR(REPLACE-first)	XLNet	51.96	37.19	**48.14** (+3.87)	69.36	46.30	**63.08** (+2.84)

Table 5: Results of models in the dataset setting of Stage II Only. † indicates scores reported in previous papers.

Model	Pre-trained	BEA-2019 (test)			CoNLL-2014 (test)		
		Prec.	Rec.	$F_{0.5}$	Prec.	Rec.	$F_{0.5}$
Dual-boost(Ge et al., 2018)†	-	-	-	-	64.47	30.48	52.72
GECToR(Omelianchuk et al., 2020)†	RoBERTa	77.2	55.1	71.5	72.1	42.0	63.0
	XLNet	79.2	53.9	72.4	77.5	40.1	65.3
GECToR(GST)(Parnow et al., 2021)†	RoBERTa	77.5	55.7	71.9	74.1	42.2	64.4
	XLNet	79.4	54.5	72.8	78.4	39.9	65.7
SAD((12+2)(Sun et al., 2021)†	BART	-	-	**72.9**	71.0	52.8	**66.4**
GECToR	RoBERTa	78.02	53.49	71.53	72.93	40.02	63.11
	XLNet	80.23	51.76	72.36	77.63	40.11	65.57
GECToR(RANDOM)	RoBERTa	79.85	51.53	71.94 (+ 0.41)	75.39	41.57	64.84 (+ 1.73)
GECToR(APPEND-first)	RoBERTa	80.31	51.14	72.08 (+0.55)	76.77	40.95	65.34 (+2.23)
GECToR(DELETE-first)	RoBERTa	79.39	52.25	71.92 (+0.39)	75.70	39.85	64.16 (+1.05)
GECToR(REPLACE-first)	RoBERTa	81.27	50.67	**72.51** (+0.98)	77.36	40.35	**65.37** (+ 2.26)
GECToR(RANDOM)	XLNet	81.14	50.83	72.49 (+0.13)	77.08	42.03	66.06 (+0.49)
GECToR(APPEND-first)	XLNet	81.89	50.55	72.85 (+0.49)	78.18	42.67	**67.02** (+1.45)
GECToR(DELETE-first)	XLNet	82.35	49.52	72.71 (+0.35)	77.05	42.03	66.04 (+0.47)
GECToR(REPLACE-first)	XLNet	81.33	51.55	**72.91** (+0.55)	77.83	41.82	66.40 (+0.83)

Table 6: Results of models at the dataset setting of Three Stages of Training.

experiments in two dataset settings: Stage II Only and Three Stages of Training. Notably, in the latter setting, we only apply our approach at Stage II and Stage III for efficiency. Finally, we evaluate the model performance in terms of official ERRANT (Bryant et al., 2017) and M^2 scorer (Dahlmeier and Ng, 2012) respectively.

6.2 Main Results and Analysis

Stage II Only. In this setting, we compare the performance of GECToR with or without applying our approach[2].

Results are presented on Table 5. Notably, the results are consistent with our findings in Section §4. That is, since correcting some types of errors benefit the corrections of other errors, all models trained with our approach significantly perform better than their corresponding baselines. Moreover, the GECToR models trained by the APPEND-first or REPLACE-first strategies are superior to models trained by DELETE-first or RANDOM, echoing the conclusions mentioned in Section §4.

Three Stages of Training. In this setting, we compare our enhanced models with more baselines under the setting of the single model, including the most related work, Dual-boost (Ge et al., 2018), GECToR(GST) (Parnow et al., 2021) and the current best NMT-based model SAD(12+2) (Sun et al., 2021).

As reported in Table 6, we obtain the similar results to Stage II Only. Our approach promotes models to obtain desirable improvements, where the APPEND-first and REPLACE-first strategies perform better. Overall, the GECToR models trained by our approach are comparable or even better than SAD(12+2). Particularly, when ensembling our

[2]Please note that previous studies do not provide the performance of other baselines under the setting of Stage II Only.

Dataset	Strategy	Evaluation	RoBERTa							
			BEA-2019 (dev)				CoNLL-2014 (test)			
			Num.	Prec.	Rec.	F_1	Num.	Prec.	Rec.	F_1
D(APPEND)	APPEND-first	$DELETE	904	64.03	19.69	30.12	496	45.45	9.07	15.13
		$REPLACE_{t}	2079	52.54	19.38	28.32	660	34.83	9.39	14.80
D(APPEND✓)	APPEND-first	$DELETE	904	79.17	33.63	47.20 (+17.08)	496	68.18	18.15	28.66 (+13.53)
		$REPLACE_{t}	2079	73.49	36.80	49.04 (+20.72)	660	60.84	28.48	38.80 (+24.00)
D(DELETE)	DELETE-first	$APPEND_{t}	1024	54.31	22.75	32.07	332	24.53	11.75	15.89
		$REPLACE_{t}	1425	52.75	18.88	27.80	716	35.19	10.61	16.31
D(DELETE✓)	DELETE-first	$APPEND_{t}	1024	60.28	25.49	35.83 (+3.76)	332	30.32	14.16	19.30 (+3.41)
		$REPLACE_{t}	1425	59.16	22.67	32.78 (+4.98)	716	40.32	13.97	20.75 (+4.44)
D(REPLACE)	REPLACE-first	$APPEND_{t}	1762	55.32	27.13	36.41	443	28.74	16.03	20.58
		$DELETE	996	58.13	19.38	29.07	767	50.00	11.34	18.49
D(REPLACE✓)	REPLACE-first	$APPEND_{t}	1762	73.57	47.56	57.77 (+21.36)	443	53.82	42.89	47.74 (+27.16)
		DELETE	996	77.99	36.65	49.86 (+20.79)	767	71.75	25.16	37.26 (+18.77)

Table 7: Results of our quantitative experiments using models enhanced by our approach. Three groups of experiments are conducted on the same data subset as Table 3.

Model	BEA-2019 (dev)			CoNLL-2014 (test)		
	Prec.	Rec.	$F_{0.5}$	Prec.	Rec.	$F_{0.5}$
GECToR	49.80	37.61	46.77	66.56	45.08	60.77
w/ TMTC	54.92	35.30	49.43	70.73	43.88	63.01
w/o turn 1	51.29	37.01	47.03	68.99	45.45	62.51
w/o turn 2	50.43	37.3	47.12	66.94	44.60	61.31
w/o orignal	55.21	32.5	48.44	71.22	41.55	62.32
mix data	53.04	31.00	46.44	71.31	40.59	61.84
w/o $\mathbb{1}(*)$	53.23	33.49	47.62	71.31	42.16	62.64

Table 8: Ablation study. Our model is based on RoBERTa and trained using APPEND-first. The $\mathbb{1}(*)$ is the indicator function mentioned in Equation 3.

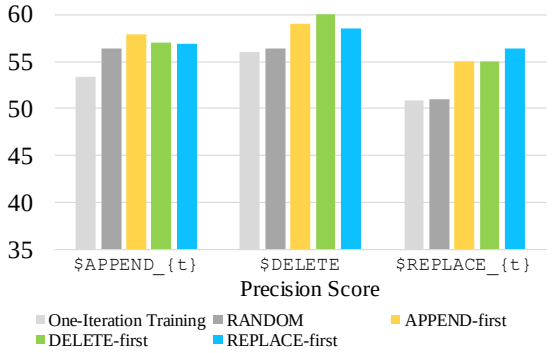

Figure 3: Label predictions of the RoBERTa-based model on the BEA-2019 dev set in the first iteration of prediction.

enhanced models with competitive GEC models, we obtain 77.93 $F_{0.5}$, achieving SOTA score on the BEA-2019 test set.

Moreover, we find that our approach allows the trained models to correct more cautiously. That is, the trained models tend to perform less but more precise corrections, compared with the basic GECToR models. One of underlying reasons is that our additionally-constructed training instances contain more $KEEP labels especially in the second turn, which makes the label predictions of the model biased.

6.3 Ablation Study

Then, we conduct more experiments to investigate the effectiveness of various details on our proposed approach.

All experimental results are provided in Table 8. Results of lines 3-5 ("w/o turn 1", "w/o turn 2", "w/o original") demonstrate that our additionally-constructed training instances are complementary to original ones. In addition, we also directly mix the additionally-constructed training instances and the original ones to train a GECToR model. However, such a training strategy does not promote the model to learn much better, showing the advantage of gradual learning error corrections. Finally, as mentioned in Section §5, some grammatical errors should not be learned within intermediate sentence. Here, we also report the performance of the GECToR model without omitting incorrect supervisal signals. As shown in the line 7 ("w/o $\mathbb{1}(*)$") of Table 8, the lower recall values indicate these incorrect $KEEP labels make the model to infer more conservatively.

6.4 Analysis

Correction Trend. Here, we use the models trained under different strategies to not only evaluate the one-iteration performance with respect to our investigated three types of labels, but also conduct quantitative experiments again. By doing so, we can investigate if our approach indeed guides the model to correct some types of error first.

Model	BEA-2019 (dev)			CoNLL-2014 (test)		
	Prec.	Rec.	$F_{0.5}$	Prec.	Rec.	$F_{0.5}$
GECToR	49.80	37.61	46.77	66.56	45.08	60.77
GECToR(APP+REP+DEL)	59.26	31.70	50.48	74.08	40.37	63.48
GECToR(APP+DEL+REP)	58.38	32.06	50.15	73.26	40.89	63.24
GECToR(REP+APP+DEL)	57.75	30.95	49.23	74.36	39.19	63.05
GECToR(REP+DEL+APP)	57.72	31.44	49.66	73.86	39.87	62.88
GECToR(DEL+APP+REP)	59.13	31.52	50.04	74.28	39.61	63.15
GECToR(DEL+REP+APP)	58.51	31.83	50.18	73.34	40.55	63.06

Table 9: Results of more fine-grained strategies. We conduct experiments by the model trained at Stage II Only based on RoBERTa.

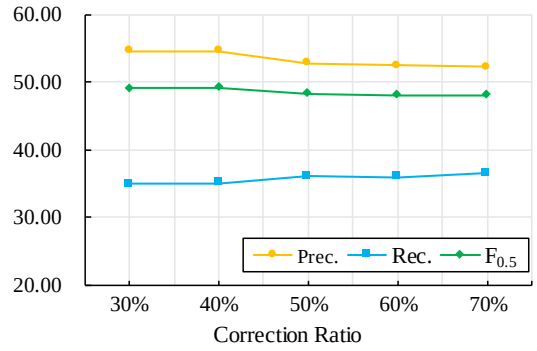

Figure 4: The precision, recall and $F_{0.5}$ values with respect to different correction ratios.

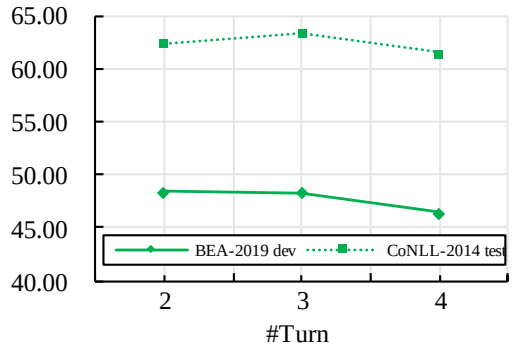

Figure 5: The $F_{0.5}$ scores of GECToR(RANDOM) with more turns of corrections.

As shown in Figure 3, we find our strategies indeed guide model to correct corresponding errors more precisely in the first iteration. Meanwhile, the less but more precise predictions occur again with respect to corresponding labels. For example, when only considering the model performance with respect to $APPEND_{t}, we observe that the model trained by APPEND-first obtains the highest precision score.

More importantly, back to Table 7, the phenomenon that correcting some types of errors benefits the others is highlighted. It indicates that our approach indeed allows the trained model to capture the interdependence between different types of corrections.

Effect of Correction Ratio. As described in Section §5, the correction ratio is an important hyper-parameter that determines the numbers of manual corrections. Thus, we try different correction ratio values to investigate its effect on our approach. Figure 4 shows the performance of the trained model with varying correction ratios. Apparently, with the correction ratio increasing, the precision score drops and recall score rises. By contrast, the overall $F_{0.5}$ scores are always steady.

Effect of More Turns of Corrections. The above experimental results show that decomposing the conventional one-iteration training of into the two-turn training is useful to improve model training. A natural problem arises: can the trained model be further improved if we use more turns of training?

To answer this question, we use the model trained by the RANDOM strategy to conduct experiments. Specifically, we decompose the one-iteration corrections into K turns of corrections, where we construct intermediate sentence by accumulatively correct $\frac{1}{K}$ errors during each turn of corrections. From Figure 5, we can observe that more turns of corrections do not benefit our models over two-turn corrections under the RANDOM strategy while with more training cost.

Also, we conduct experiments using more fine-grained strategies. For example, we can design a training strategy: after learning corrections of $APPEND_{t}, the model learns to correct errors of $REPLACE_{t} and then to correct others. For convenience, we name this strategy as APP+REP+DEL, where APP, REP and DEL are abbreviations of $APPEND_{t}, $REPLACE_{t} and $DELETE, respectively. As illustrated in Table 9, all models trained by our approach obtain slightly better performance when introducing more iterations of corrections. However, they require

almost 1.5x training time compared with our standard TMTC approach.

7 Conclusion

In this paper, we have firstly conducted quantitative experiments to explore the interdependence between different types of corrections, with the finding that performing some types of corrections such as appending or replacing words first help models to correct other errors. Futhermore, we propose a Type-Driven Multi-Turn Corrections (TMTC) approach for GEC, which allows the trained model to be not only explicitly aware of the progressive corrections, but also exploit the interdependence between different types of corrections. Extensive experiments show that our enhanced model is able to obtain comparable or better performance compared with the SOTA GEC model.

In the future, we plan to apply bidirectional decoding (Zhang et al., 2018; Su et al., 2019; Zhang et al., 2019) to further improve our approach. Besides, inspired by the recent syntax-aware research (Li et al., 2021), we will explore the interdependence between corrections from other perspectives for GEC such as syntax.

Acknowledgment

The project was supported by National Key Research and Development Program of China (No. 2020AAA0108004), National Natural Science Foundation of China (No. 61672440), Natural Science Foundation of Fujian Province of China (No. 2020J06001), and Youth Innovation Fund of Xiamen (No. 3502Z20206059). We also thank the reviewers for their insightful comments.

References

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

Adriane Boyd. 2018. Using wikipedia edits in low resource grammatical error correction. In *NUT@EMNLP*, pages 79–84.

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

Meng Hui Chen, Tao Ge, Xingxing Zhang, Furu Wei, and M. Zhou. 2020. Improving the efficiency of grammatical error correction with erroneous span detection and correction. In *EMNLP*, pages 7162–7169.

Daniel Dahlmeier and Hwee Tou Ng. 2012. Better evaluation for grammatical error correction. In *NACCL*, 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 *BEA@NAACL-HLT*, pages 22–31.

Tao Ge, Furu Wei, and M. Zhou. 2018. Fluency boost learning and inference for neural grammatical error correction. In *ACL*, pages 1055–1065.

M. Ghufron and Fathia Rosyida. 2018. The role of grammarly in assessing english as a foreign language (efl) writing. *Lingua Cultura*, 12:395–403.

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

Michael Heilman, Aoife Cahill, Nitin Madnani, Melissa Lopez, Matthew Mulholland, and Joel Tetreault. 2014. Predicting grammaticality on an ordinal scale. In *ACL*, pages 174–180.

Clare-Marie Karat, Christine Halverson, Daniel B. Horn, and John Karat. 1999. Patterns of entry and correction in large vocabulary continuous speech recognition systems. In *CHI '99*, pages 568–575.

Marek Kubis, Zygmunt Vetulani, Mikolaj Wypych, and Tomasz Zietkiewicz. 2020. Open challenge for correcting errors of speech recognition systems. *ArXiv*, abs/2001.03041.

Zhongli Li, Qingyu Zhou, Chao Li, Ke Xu, and Yunbo Cao. 2021. Improving BERT with syntax-aware local attention. In *Findings of ACL*, pages 645–653.

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

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*, abs/1907.11692.

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

Courtney Napoles, Keisuke Sakaguchi, and Joel Tetreault. 2017a. Jfleg: A fluency corpus and benchmark for grammatical error correction. In *EACL*, pages 229–234.

Courtney Napoles, Keisuke Sakaguchi, and Joel R. Tetreault. 2017b. Jfleg: A fluency corpus and benchmark for grammatical error correction. In *EACL*, pages 229–234.

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.

Kostiantyn Omelianchuk, Vitaliy Atrasevych, Artem Chernodub, and Oleksandr Skurzhanskyi. 2020. GECToR – grammatical error correction: Tag, not rewrite. In *BEA@ACL*, pages 163–170.

Kevin Parnow, Zuchao Li, and Hai Zhao. 2021. Grammatical error correction as gan-like sequence labeling. In *Findings of ACL*, pages 3284–3290.

Linfeng Song, Ante Wang, Jinsong Su, Yue Zhang, Kun Xu, Yubin Ge, and Dong Yu. 2020. Structural information preserving for graph-to-text generation. In *ACL*, pages 7987–7998.

Felix Stahlberg and Shankar Kumar. 2020. Seq2Edits: Sequence transduction using span-level edit operations. In *EMNLP*, pages 5147–5159.

Jinsong Su, Xiangwen Zhang, Qian Lin, Yue Qin, Junfeng Yao, and Yang Liu. 2019. Exploiting reverse target-side contexts for neural machine translation via asynchronous bidirectional decoding. *Artif. Intell.*, 277:103168.

Xin Sun, Tao Ge, Furu Wei, and Houfeng Wang. 2021. Instantaneous grammatical error correction with shallow aggressive decoding. In *ACL/IJCNLP*, pages 5937–5947.

Toshikazu Tajiri, Mamoru Komachi, and Yuji Matsumoto. 2012. Tense and aspect error correction for ESL learners using global context. In *ACL*, pages 198–202.

Yujin Takahashi, Satoru Katsumata, and Mamoru Komachi. 2020. Grammatical error correction using pseudo learner corpus considering learner's error tendency. In *ACL SRW*, pages 27–32.

Haoyu Wang, Shuyan Dong, Yue Liu, James Logan, Ashish Kumar Agrawal, and Yang Liu. 2020. Asr error correction with augmented transformer for entity retrieval. In *INTERSPEECH*, pages 1550–1554.

Lihao Wang and Xiaoqing Zheng. 2020. Improving grammatical error correction models with purpose-built adversarial examples. In *EMNLP*, pages 2858–2869.

Kun Xu, Linfeng Song, Yansong Feng, Yan Song, and Dong Yu. 2020. Coordinated reasoning for cross-lingual knowledge graph alignment. In *AAAI*, pages 9354–9361.

Shuyao Xu, Jiehao Zhang, Jin Chen, and Longlu Qin. 2019. Erroneous data generation for grammatical error correction. In *BEA@ACL*, pages 149–158.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *NeurIPS*, pages 5754–5764.

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

Biao Zhang, Deyi Xiong, Jinsong Su, and Jiebo Luo. 2019. Future-aware knowledge distillation for neural machine translation. *TASLP*, 27(12):2278–2287.

Xiangwen Zhang, Jinsong Su, Yue Qin, Yang Liu, Rongrong Ji, and Hongji Wang. 2018. Asynchronous bidirectional decoding for neural machine translation. In *AAAI*, pages 5698–5705.

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

A Appendix

A.1 Quantitative Experiments on XLNet

We also conduct quantitative experiments described in Section §4 using model trained based on XLNet. The overall results are closely similar to Table 3, which indicates that our findings and conclusions are not specific to a certain model or a certain dataset, but common among realistic human-making datasets.

A.2 Evaluation on JFLEG

Suggested by reviewers, we evaluate our approach on the JFLEG (Napoles et al., 2017a) dataset which focus on fluency. As shown in Table 11 and Table 12, models trained by our approach obtain higher GLEU (Heilman et al., 2014) compared with baselines, which demonstrate the effectiveness of decomposing one-iteration correction into multiple turns. However, editing-action based interdependence seems not very beneficial from the view of fluency.

Dataset	Evaluation	XLNet							
		BEA-2019 (dev)				CoNLL-2014 (test)			
		Num.	Prec.	Rec.	F1	Num.	Prec.	Rec.	F1
Original Dataset	$APPEND_{t}	2609	50.61	38.06	43.45	621	24.4	26.09	25.21
	$DELETE	1403	52.79	25.66	34.53	1115	49.65	19.01	27.50
	$REPLACE_{t}	3495	49.10	24.12	32.35	1398	37.06	20.89	26.72
D(APPEND)	$DELETE	904	61.89	21.02	31.38	496	46.34	15.32	23.03
	$REPLACE_{t}	2079	50.65	20.68	29.37	660	32.30	14.24	19.77
D(APPEND✓)	$DELETE	904	72.66	30.86	43.32 (+11.94)	496	68.18	18.15	28.66 (+5.63)
	$REPLACE_{t}	2079	67.13	36.84	47.58 (+18.21)	660	60.84	28.48	38.80 (+19.03)
D(DELETE)	$APPEND_{t}	1024	50.27	27.44	35.50	332	18.09	16.57	17.30
	$REPLACE_{t}	1425	49.57	20.00	28.50	716	28.12	14.80	19.40
D(DELETE✓)	$APPEND_{t}	1024	54.91	28.42	37.45 (+1.95)	332	30.32	14.16	19.30 (+2.00)
	$REPLACE_{t}	1425	51.40	21.89	30.71 (+2.21)	716	40.32	13.97	20.75 (+1.35)
D(REPLACE)	$APPEND_{t}	1762	55.32	31.38	38.85	443	20.28	19.86	20.07
	$DELETE	996	56.37	20.88	30.48	767	45.16	16.43	24.09
D(REPLACE✓)	$APPEND_{t}	1762	65.47	50.91	57.28 (+18.43)	443	53.82	42.89	47.74 (+27.67)
	$DELETE	996	70.89	35.94	47.70 (+17.22)	767	71.75	25.16	37.26 (+16.51)

Table 10: Results of our control expriment. Four groups of results are obtained by the same re-implemented GECToR model.

Model	Pre-trained	BEA-2019 (dev)			CoNLL-2014 (test)			JFLEG (test)
		Prec.	Rec.	$F_{0.5}$	Prec.	Rec.	$F_{0.5}$	GLEU
GECToR(Omelianchuk et al., 2020)†	RoBERTa	50.30	30.50	44.50	67.50	38.30	58.60	-
	XLNet	47.10	34.20	43.80	64.60	42.60	58.50	-
GECToR	RoBERTa	49.80	37.61	46.77	66.56	45.08	60.77	42.75
	XLNet	45.55	39.81	44.27	64.04	48.67	60.24	42.90
GECToR(RANDOM)	Roberta	52.88	36.05	48.37 (+1.60)	69.54	44.32	62.43 (+1.66)	**56.64**
GECToR(APPEND-first)	Roberta	54.92	35.30	**49.43** (+2.66)	70.73	43.88	**63.01** (+2.24)	56.61
GECToR(DELETE-first)	Roberta	53.85	35.13	48.67 (+1.90)	70.57	42.78	62.45 (+1.68)	56.48
GECToR(REPLACE-first)	Roberta	54.78	34.82	49.14 (+2.37)	70.2	43.92	62.70 (+1.93)	55.97
GECToR(RANDOM)	XLNet	49.74	38.47	46.99 (+2.72)	67.41	46.68	61.91 (+1.67)	56.84
GECToR(APPEND-first)	XLNet	51.10	37.72	47.71 (+3.44)	67.74	46.39	62.03 (+1.79)	**57.15**
GECToR(DELETE-first)	XLNet	50.48	37.49	47.21 (+2.97)	67.33	46.42	61.79 (+1.55)	56.60
GECToR(REPLACE-first)	XLNet	51.96	37.19	**48.14** (+3.87)	69.36	46.30	**63.08** (+2.84)	56.73

Table 11: Results of models under the settings of Stage II Only. † indicates scores reported in previous papers.

Model	Pre-trained	BEA-2019 (test)			CoNLL-2014 (test)			JFLEG (test)
		Prec.	Rec.	$F_{0.5}$	Prec.	Rec.	$F_{0.5}$	GLEU
Dual-boost(Ge et al., 2018)†		-	-	-	64.47	30.48	52.72	
GECToR(Omelianchuk et al., 2020)†	RoBERTa	77.2	55.1	71.5	72.1	42.0	63.0	-
	XLNet	79.2	53.9	72.4	77.5	40.1	65.3	-
GECToR(GST)(Parnow et al., 2021)†	RoBERTa	77.5	55.7	71.9	74.1	42.2	64.4	-
	XLNet	79.4	54.5	72.8	78.4	39.9	65.7	-
SAD((12+2)(Sun et al., 2021)†	BART	-	-	**72.9**	71.0	52.8	**66.4**	-
GECToR	RoBERTa	78.02	53.49	71.53	72.93	40.02	63.11	42.96
	XLNet	80.23	51.76	72.36	77.63	40.11	65.57	43.11
GECToR(RANDOM)	Roberta	79.85	51.53	71.94 (+ 0.41)	75.39	41.57	64.84 (+ 1.73)	**59.05**
GECToR(APPEND-first)	Roberta	80.31	51.14	72.08 (+0.55)	76.77	40.95	65.34 (+2.23)	58.88
GECToR(DELETE-first)	Roberta	79.39	52.25	71.92 (+0.39)	75.70	39.85	64.16 (+1.05)	58.94
GECToR(REPLACE-first)	Roberta	81.27	50.67	**72.51** (+0.98)	77.36	40.35	**65.37** (+ 2.26)	59.03
GECToR(RANDOM)	XLNet	81.14	50.83	72.49 (+0.13)	77.08	42.03	66.06 (+0.49)	**58.73**
GECToR(APPEND-first)	XLNet	81.89	50.55	72.85 (+0.49)	78.18	42.67	**67.02** (+1.45)	58.64
GECToR(DELETE-first)	XLNet	82.35	49.52	72.71 (+0.35)	77.05	42.03	66.04 (+0.47)	58.45
GECToR(REPLACE-first)	XLNet	81.33	51.55	**72.91** (+0.55)	77.83	41.82	66.40 (+0.83)	58.42

Table 12: Results of models under the settings of Three Stages of Training.