SynGEC: Syntax-Enhanced Grammatical Error Correction with a Tailored GEC-Oriented Parser

Anonymous ACL submission

Abstract

Despite their tremendous success, current cutting-edge grammatical error correction (GEC) models make little use of syntactic knowledge, which plays an important role when humans try to understand and fix ungrammatical sentences. This work proposes a syntax-enhanced GEC approach (SynGEC), to incorporate syntactic information into the encoder part of GEC models. The key challenge for this idea is that the performance of off-the-shelf parsers dramatically drops since they are usually trained on clean grammatical sentences. To confront this challenge, we propose to build a tailored GEC-oriented parser (GOPar) using parallel GEC training data as a pivot. First, we present an extended annotation scheme that allows us to represent both grammatical errors and syntactic structure under a unified tree structure. Then, we obtain parse trees of the source incorrect sentences by projecting trees of the target correct sentences and using them for training GOPar. We employ graph convolution networks to encode tree structures produced by GOPar. Experiments on three English/Chinese benchmark datasets show that our proposed SynGEC approach consistently and substantially outperforms the strong baselines and achieves new single-model state-of-the-art performance on all datasets.

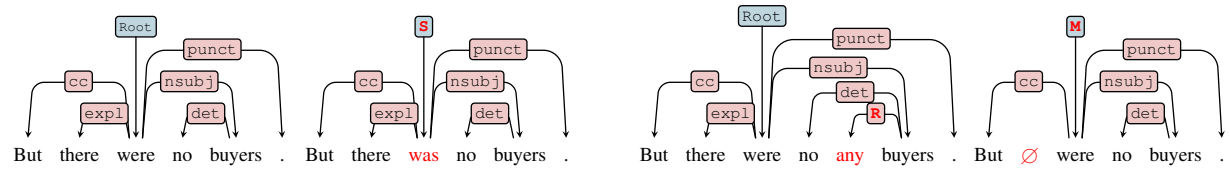
1 Introduction

Given a potentially ungrammatical sentence, the grammatical error correction (GEC) task aims to produce a grammatical target sentence (Grundkiewicz et al., 2020; Wang et al., 2021). Recent mainstream approaches treat GEC as a monolingual machine translation (MT) task (Yuan and Briscoe, 2016; Junczys-Dowmunt et al., 2018). Standard encoder-decoder based MT models, e.g., Transformer (Vaswani et al., 2017), have emerged as a dominant paradigm and achieved state-of-the-art (SOTA) results on various GEC datasets (Rothe et al., 2021; Stahlberg and Kumar, 2021; Sun et al., 2022; Zhang et al., 2022). Despite their impressive achievements, most works take the input sentence as a sequence of tokens, and totally ignore the internal syntactic or semantic structure.

Compared with MT, GEC has two peculiarities that directly motivate this work. First, the training data for GEC models is much less abundant, and the data sparseness problem may be alleviated by incorporating linguistic structure knowledge like syntax. As shown in Table 1 and Table 4, the English and Chinese GEC tasks only have about 126K and 157K high-quality labeled source/target sentence pairs for training, if not considering the highly noisy crowd-annotated Lang8 data (Mita et al., 2020). Second, many errors in ungrammatical sentences are intrinsically correlated with syntactic parsing. For example, some errors, such as inconsistency in tense or singular-vs-plural forms, can be better detected and corrected with the help of long-range syntactic dependencies.

In this paper, we propose SynGEC, an approach for injecting the syntactic structure of the input sentence into the encoder part of GEC models. The critical challenge for this idea is the unsatisfactory parsing performance on ungrammatical sentences. Typically, an off-the-shelf syntactic parser is trained upon clean treebanks, while the input sentences for GEC usually contain various grammatical errors. Such a big gap inevitably leads to many ill-formed parse trees (Hashemi and Hwa, 2016). As far as we know, two recent works try to incorporate syntactic knowledge into GEC, but both fail to directly handle this challenge (Wan and Wan, 2021; Li et al., 2022) (see Section 6 for more detailed discussion).

Meanwhile, there has been a line of works that try to adapt syntactic parsing for ungrammatical texts by extending the annotation guidelines and manually annotating data (Dickinson and Ragheb, 2009; Berzak et al., 2016; Nagata and Sakaguchi,



(a) The correct sentence. (b) Substituted errors. (c) Redundant errors. (d) Missing errors.

Figure 1: Illustration of our extended annotation scheme for all three kinds of grammatical errors.

2016). However, manual annotation is extremely expensive, and thus existing annotated datasets are of a very small scale.

To confront the challenge of unreliable performance of off-the-shelf parsers on ungrammatical sentences, we propose to train a tailored GEC-oriented parser (GOPar). The basic idea is utilizing parallel source/target sentence pairs in the GEC training data. First, we parse the target correct sentences using a vanilla off-the-shelf parser. Then, we construct the tree for the incorrect sentences via tree projection. To accommodate grammatical errors, we propose an extended annotation scheme based on several straightforward rules, which allows us to represent both grammatical errors and syntactic structure in a unified tree. Finally, we train GOPar directly on the automatically constructed trees of the source sentences in the GEC training data.

To incorporate syntactic information, we cascade several graph convolutional network (GCN) layers (Kipf and Welling, 2017) above the encoder of our baseline Transformer-based GEC model. We conduct experiments on two widely-used English GEC evaluation datasets, i.e., CoNLL-14 (Ng et al., 2014) and BEA-19 (Bryant et al., 2019), and the recently proposed Chinese MuCGEC dataset (Zhang et al., 2022). Results show that our SynGEC approach achieves consistent and substantial improvement on all datasets, even when the baseline model is enhanced with BART (Lewis et al., 2020). SynGEC achieves new single-model SOTA results: 69.6 $F_{0.5}$ score on CoNLL-14, 72.9 $F_{0.5}$ score on BEA-19, and 46.8 $F_{0.5}$ score on MuCGEC, outperforming previous SOTA under comparable settings by large margins (+3.5/+0.5/+7.3). We will release our code at `http://github.com`.

2 Our GEC-Oriented Parser

This section describes our tailored GOPar, which is more competent in parsing ungrammatical sentences than off-the-shelf parsers.

2.1 An Extended Annotation Scheme

The standard annotation scheme for representing syntax is designed for grammatical sentences, and thus does not cover many structures in grammatically erroneous sentences. Therefore, to obtain a tailored parser, our first task is to extend the annotation scheme and, more specifically, to design a complementary set of annotation guidelines to handle different grammatical mistakes. With this scheme, we can use unified tree structure to represent both grammatical errors and syntactic information.

As shown in Figure 1, we propose a simple extended annotation scheme based on several straightforward rules, corresponding to the three types of grammatical errors, i.e., *substituted*, *redundant* and *missing* (Bryant et al., 2017). Correspondingly, we add three labels into the original syntactic label set, i.e., "S", "R" and "M", to capture three kinds of errors. Since such categorization is also adopted in the grammatical error detection (GED) task (Yuan et al., 2021), we refer to them as GED labels.

Substituted errors (S) include spelling errors, tense errors, singular/plural inconsistency errors, etc. For simplicity, we do not consider such fine-grained categories, and use a single "S" label to indicate that the word should be replaced by another one, as shown in Figure 1(b).

Redundant errors (R) mean that some words should be deleted. For each redundant word, we let it depend on its right-side adjacent word, with a label "R", even if the right-side adjacent word is redundant also, as shown in Figure 1(c).

Missing errors (M) mean that some words should be inserted. For each missing word, we assign a label "M" to the incoming arc of its right-side adjacent word, unless the missing word is at the end of the sentence, as shown in Figure 1(d). If several consecutive words are missing, the annotation remains the same with when a single word is missing. Moreover, since a missing word may have children in the tree of the correct sentence, we let them depend on the head word of the missing

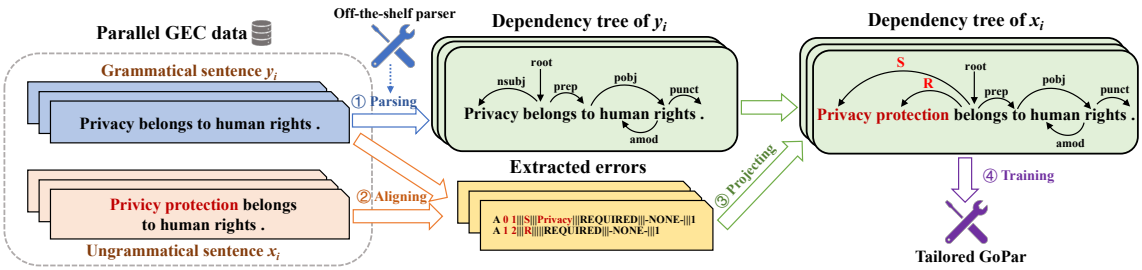


Figure 2: The workflow for obtaining our tailored GOPar.

word, without changing their syntactic labels.

Based on the above simple rules, we aim to perform minimal adjustments and make the resulting tree as consistent as possible with the syntactic tree of the correct sentence. Meanwhile, please kindly notice that our annotation scheme may encounter problems when different types of errors occur in consecutive positions of a sentence. Taking "But was no buyers" as an example, we need to replace "was" with "were" and then insert "there" before "were" at the same time. Therefore, according to our rules, the label of the incoming arc of "was" can be either "S" or "M", leading to a label conflict. To decide a unique label, we simply define a priority order: "S">"R">"M'.

2.2 Building GOPar

Based on the above annotation scheme, we propose to train our tailored GOPar using the parallel GEC training data $D = \{(x_i, y_i)\}$ as a pivot. The major goal is to automatically generate high-quality parse trees for large-scale sentences with realistic grammatical errors, and use them to train a parser suitable for parsing ungrammatical sentences. Figure 2 illustrates the workflow, consisting of the following four steps.

First, we use an off-the-shelf parser to parse the target correct sentences (e.g., y_i) of the GEC training data. Since such sentences are free from grammatical errors, the parser is more likely to produce reliable parse trees.

Second, we use ERRANT (Bryant et al., 2017)[1] to extract all grammatical errors in the source incorrect sentence (e.g., x_i) according to the alignments between x_i and y_i.

Third, we construct the parse tree of x_i by projecting the tree of y_i. For words that are not related to any errors, dependencies (arcs) and labels are directly copied; for those related to errors, dependencies and labels are assigned according to the rules introduced in Section 2.1.

Fourth, with constructed parse trees for all source-side sentences in D, we then use them as a treebank to train our tailored GOPar.

An alternative way to build a tailored GOPar is directly utilizing manually labeled treebanks. Since existing treebanks only contain grammatical sentences, we can inject synthetic errors based on rules or back-translation models (Foster et al., 2008; Cahill, 2015). Then, we can produce parse trees for ungrammatical sentences analogously through the above second and third steps. However, our preliminary experiments show that GOPar built in this way is much inferior and can only slightly improves our baseline GEC model. We suspect that the reasons are two-fold. On the one hand, there is a considerable gap between synthetic and real grammatical errors; on the other hand, the generated data is not enough to train GOPar adequately due to the limited scale of existing treebanks.

3 The DepGCN-based GEC Model

In this section, we describe our DepGCN-based GEC Model. Inspired by Wan and Wan (2021), we first adopt GCN (Kipf and Welling, 2017) to encode the dependency syntax trees of the source sentence. Then, we feed the encoded syntactic information into a Transformer-based GEC model.

3.1 Transformer Backbone

We model GEC as a sequence-to-sequence task and employ the commonly used Transformer model (Vaswani et al., 2017) as the backbone. The Transformer is composed of an encoder and a decoder. The encoder utilizes the multi-head self-attention mechanism to get the contextualized representations of each token in the source sentence. The decoder has a similar architecture while additionally containing a masked multi-head self-attention module to model the generated token information.

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

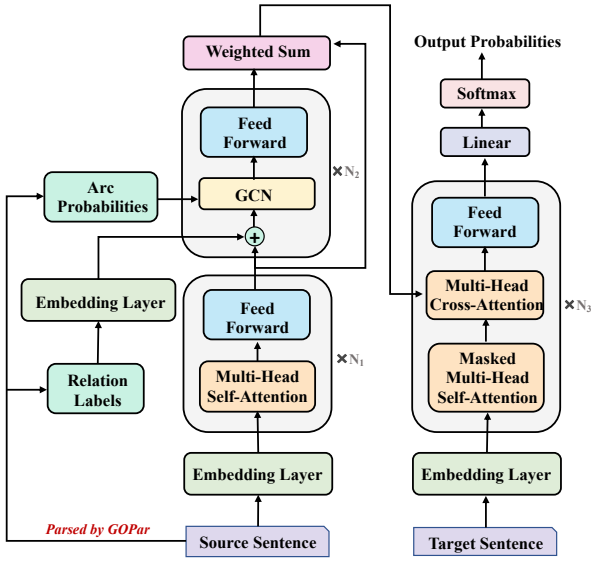


Figure 3: Overview of our DepGCN-based GEC model. The operation $\oplus$ denotes vector concatenation. N_1, N_2 and N_3 denote the number of identical Transformer encoder blocks, DepGCN blocks and Transformer decoder blocks, respectively. We also employ a residual connection (He et al., 2016) followed by layer normalization (Ba et al., 2016) around each sub-layer.

During training, the objective function is minimizing the teacher forcing negative log-likelihood loss (Williams and Zipser, 1989), formally:

$$\begin{aligned}\mathcal{L}(\theta) &= -log(P(y|x;\theta)) \\ &= -log\sum_{i=1}^{n}(P(y_i|y_{<i};x;\theta))\end{aligned} \quad (1)$$

where θ is trainable model parameters, x is the source sentence, $y = \{y_1, y_2, ..., y_n\}$ is the ground-truth target sentence with n tokens, and $y_{<i} = \{y_1, y_2, ..., y_{i-1}\}$ is the visible tokens in i-th training time step.

During inference, we utilize the beam search decoding (Wiseman and Rush, 2016) to find an optimal sequence o by maximizing the conditional probability $P(o|x;\theta)$.

Previous works show that PLMs, e.g., BART (Lewis et al., 2020) and T5 (Raffel et al., 2020), can improve GEC performance by a large margin (Rothe et al., 2021; Sun et al., 2022). We further use BART to build stronger baseline, since it shares the same model architecture with our Transformer backbone. Specifically, we use the BART parameters to initialize our Transformer backbone and then continue training on GEC training data. More details are discussed in Section 4.1 and Section 5.

3.2 Dependency GCN (DepGCN)

We employ GCN to encode the dependency syntax information, named DepGCN. The DepGCN module stacks several identical blocks, and each block has a GCN sub-layer and a feed-forward sub-layer.

For the GCN sub-layer, we introduce the information of the dependency arcs and dependency labels simultaneously. We compute the output $h_i^{(l)}$ of l-th GCN at the i-th token as:

$$h_i^{(l)} = \text{ReLU}(\sum_{j=1}^{n} A_{ij}W^{(l)}(h_j^{l-1} \oplus e_{ij}) + b^{(l)}) \quad (2)$$

where $A \in \mathbb{R}^{n\times n}$ denotes the adjacency matrix, $e_{ij} \in \mathbb{R}^d$ is the embedding of the dependency label between word w_i and word w_j, $W \in \mathbb{R}^{d\times 2d}$ and $b \in \mathbb{R}^d$ are model parameters. ReLU (Nair and Hinton, 2010) is the activation function, and $\oplus$ refers to the vector concatenation.

To reduce error propagation of GOPar, follow Zhang et al. (2020a), we use the arc probability matrix derived from GOPar as the adjacency matrix A, which can provide all latent syntactic structures.

We then feed the outputs of the GCN sub-layer to the feed-forward (FF) layer that contains two linear transformations with a ReLU activation function in between, as shown below:

$$\text{FF}(h) = \text{ReLU}(W_1 h + b_1)W_2 + b_2 \quad (3)$$

3.3 Integrating Syntactic Information

We put DepGCN at the junction between Transformer encoder and decoder, and build a DepGCN-based GEC model. As depicted in Figure 3, given an input sentence x and corresponding syntactic information x^{syn} (including dependency arcs and labels) from GOPar, we first obtain the contextualized representations H^c of x from Transformer encoder. Then, we feed H^c and x^{syn} into DepGCN and get the syntax-enhanced representations H^{syn}. Finally, we use the weighted-sum operation to derive the fusion representations H^f as the input of Transformer decoder:

$$H^f = \beta H^c + (1-\beta)H^{syn} \quad (4)$$

where $\beta \in (0, 1)$ is a hyper-parameter. The training and inference processes are consistent with the Transformer baseline.

Dataset	#Sentences	%Error	Usage
CLang8	2,372,119	57.8	Pre-training
FCE	34,490	62.6	Fine-tuning I
NUCLE	57,151	38.2	Fine-tuning I
W&I+LOCNESS	34,308	66.3	Fine-tuning I&II
BEA-19-*Dev*	4,384	65.2	Validation
CoNLL-14-*Test*	1,312	72.3	Testing
BEA-19-*Test*	4,477	-	Testing

Table 1: Statistics of English GEC datasets.

4 Experiments on English GEC

4.1 Settings

Datasets and evaluation. We first pre-train our model on the cleaned version of the Lang8 dataset (CLang8) [2] released by Rothe et al. (2021). Then, we use the FCE dataset (Yannakoudakis et al., 2011), the NUCLE dataset (Dahlmeier et al., 2013) and the W&I+LOCNESS train-set (Bryant et al., 2019) for model fine-tuning following previous studies. Like Omelianchuk et al. (2020), we decompose the fine-tuning procedure into two stages: 1) fine-tuning on FCE+NUCLE+W&I+LOCNESS; 2) further fine-tuning only on the small-scale but high-quality W&I+LOCNESS. For evaluation, we report average results over three distinct runs on the CoNLL-14 test-set [3] (Ng et al., 2014) evaluated by M2Scorer (Dahlmeier and Ng, 2012) and BEA-19 test-set (Bryant et al., 2019) evaluated by ERRANT (Bryant et al., 2017). The BEA-19 dev-set serves as validation data during the whole training. The statistics of above datasets are shown in Table 1.

GEC model details. We adopt Fairseq[4] (Ott et al., 2019) to build our Transformer baseline and DepGCN-based model. For the DepGCN-based model, we stack 3 DepGCN blocks and set aggregation factor β in Equation 4 to 0.5. We apply BPE (Sennrich et al., 2016) to generate a 32K shared subword vocabulary. We re-implement the Dropout-Src mechanism proposed by Junczys-Dowmunt et al. (2018) and perform it on the source word embedding matrix to alleviate over-fitting. More model details are discussed in Appendix A.

GOPar details. We employ the biaffine parser (Dozat and Manning, 2017) as the model structure of GOPar. For implementation, we adopt the parsing toolkit Supar[5] (Zhang et al., 2020b) and set hyper-parameters following their paper. After compare several popular PLMs, we use ELECTRA (Clark et al., 2020) to provide the contextual token representations[6]. The GOPar training data is generated from CLang8 GEC dataset (Rothe et al., 2021) by the data construction approach described in detail in Section 2. For using GOPar in GEC, since the inputs of GEC models are subword sequences while the basic token unit in syntax trees is the word, we add arcs from all subwords of w to all subwords of v if there exits an arc from word w to word v in the word-level tree before feeding this tree into GEC models.

Incorporating BART. To verify that the syntactic knowledge is still useful after incorporating a powerful PLM, we use BART (Lewis et al., 2020) to initialize the Transformer backbone of our models. It is noteworthy that we adopt a two-stage training procedure to keep the training stable when incorporating BART into our DepGCN-based model. Firstly, we fine-tune the BART-initialized Transformer backbone until it converges. Secondly, we add an auxiliary DepGCN module into the converged Transformer backbone and only tune the DepGCN parameters on the same training data.

4.2 Main Results

The main results are listed in Table 2. In the top group of results without PLMs, our SynGEC approach achieves 65.2/68.4 $F_{0.5}$ scores on CoNLL-14 and BEA-19 test-sets, respectively, outperforming all other systems utilizing syntax. The performance of SynGEC is only lower than Stahlberg and Kumar (2021) on both test-sets as they use extra huge synthetic corpus with 540M sentences while we only use 2.4M. The incorporation of syntactic information provided by GOPar leads to 3.5/4.2 $F_{0.5}$ improvements over our baseline, which demonstrates that the syntactic knowledge from GOPar is quite helpful for GEC and our DepGCN-based GEC model can effectively capture it.

In the bottom group, our SynGEC approach augmented with BART achieves 69.6/72.9 $F_{0.5}$ scores, continuously outperforming other PLM-augmented models under similar sizes[7]. After removing syntax, the $F_{0.5}$ scores decline by 0.9 on both datasets,

[2] CLang8 can be downloaded from https://github.com/google-research-datasets/clang8

[3] We use the *official-2014.combined-withalt.m2* version of CoNLL-14 following previous SOTA (Rothe et al., 2021).

[4] https://github.com/pytorch/fairseq

[5] https://github.com/yzhangcs/parser

[6] The download links of all PLMs used in our experiments can be found in Appendix B.

[7] Rothe et al. (2021) also build a lager GEC model based on the T5-11B (Raffel et al., 2020) and achieve 68.9/75.9 $F_{0.5}$ scores. For a fair comparison, we do not list this result in Table 2 as this model is about 24× larger than ours.

	System	Extra Data Size	Transformer Layer, Hidden, FFN	CoNLL-14-*test* P	R	$F_{0.5}$	BEA-19-*test* P	R	$F_{0.5}$
w/o PLM	**w/o syntax**								
	Kiyono et al. (2019)$^{\circ}$	70M	12+12,1024,4096	67.9	44.1	61.3	65.5	59.4	64.2
	Lichtarge et al. (2020)$^{\triangle\blacktriangle}$	340M	12+12,1024,4096	69.4	43.9	62.1	67.6	62.5	66.5
	Stahlberg and Kumar (2021)$^{\triangle\blacktriangle\square}$	540M	12+12,1024,4096	72.8	49.5	**66.6**	72.1	64.4	70.4
	Our Baseline$^{\heartsuit}$	2.4M	6+6,512,2048	69.6	42.4	61.7	66.8	55.5	64.2
	w/ syntax								
	Wan and Wan (2021)$^{\blacklozenge}$	10M	6+6,512,2048	74.4	39.5	63.2	74.5	48.6	67.3
	Li et al. (2022)$^{\spadesuit}$	30M	12+12,1024,4096	66.7	38.3	58.1	-	-	-
	SynGEC$^{\heartsuit}$	2.4M	6+6,512,2048	71.6	47.9	**65.2**	70.9	59.9	**68.4**
	GOPar→Off-the-shelf Parser	2.4M	6+6,512,2048	70.5	42.6	62.4	67.3	55.4	64.5
w/ PLM	**w/o syntax**								
	Kaneko et al. (2020)$^{\circ}$	70M	12+12,1024,4096	69.2	45.6	62.6	67.1	60.1	65.6
	Katsumata and Komachi (2020)	-	12+12,1024,4096	69.3	45.0	62.6	68.3	57.1	65.6
	Omelianchuk et al. (2020)$^{\diamondsuit}$	9M	12+0,768,3072	77.5	40.1	65.3	79.2	53.9	**72.4**
	Rothe et al. (2021)$^{\heartsuit}$	2.4M	12+12,1024,4096	-	-	66.1	-	-	72.1
	Our Baseline$^{\heartsuit}$	2.4M	12+12,1024,4096	75.6	50.4	**68.7**	74.0	64.9	72.0
	w/ syntax								
	Li et al. (2022)$^{\spadesuit}$	30M	12+12,1024,4096	68.1	44.1	61.4	-	-	-
	SynGEC$^{\heartsuit}$	2.4M	12+12,1024,4096	76.6	51.1	**69.6**	75.1	65.5	**72.9**
	GOPar→Off-the-shelf Parser	2.4M	12+12,1024,4096	76.1	50.0	68.9	74.6	64.1	72.3

Table 2: **Single-model** results on English GEC test-sets. Our results are averaged over three runs with different random seeds. **Layer**, **Hidden** and **FFN** denote the depth, hidden size and feed-forward network size of Transformer. "**w/ PLM**" means using pre-trained language models. "**w/ syntax**" means using syntactic knowledge. Besides the public human-annotated training data, current GEC systems variously use private and/or artificial data, including: $^{\circ}$artificial Gigaword (70M sentences), $^{\triangle}$Wikipedia revision histories (170M), $^{\blacktriangle}$artificial Wikipedia (170M), $^{\square}$artificial Colossal Clean Crawled Corpus (200M), †non-public CLC (2M), $^{\diamondsuit}$artificial one-billion-word (9M), $^{\blacklozenge}$artificial one-billion-word (10M), $^{\spadesuit}$artificial one-billion-word (30M) , $^{\heartsuit}$cleaned version of Lang8 (2.4M).

which reveals that the contribution from adaptive syntax and PLMs does not fully overlap.

4.3 Analysis and Discussion

Effectiveness of GOPar. In Table 2, we present the results of using an off-the-shelf parser[8] to provide syntactic knowledge (GOPar → Off-the-shelf Parser). After changing the parser, the impact of syntax becomes marginal under all settings. This observation implies that the performance gains contributed from syntax are highly contingent on the quality of parses. We look further into the parses and find that GOPar is more robust when facing grammatical errors and can further identify such errors, while the off-the-shelf parser is vulnerable and tends to provide incorrect parses. So we can draw a conclusion that the task adaptation of parsers is essential when applying syntax to the GEC task.

Decomposition of syntactic information. To gain more insights on how adaptive syntactic information works, we decompose it into three parts: 1) the arc information, which means only using the topological structure of the syntax tree; 2) the GED label information, which refers to the special labels "S", "R" and "M" for marking erroneous tokens; and 3) the syntax label information, such as "subj" and "iobj" for different syntactic relations. We conduct an ablation study to explore the effect of each kind of information for GEC[9], as shown in Table 3. First, removing GED labels or syntax labels reduce the performance of the SynGEC model to the similar extent, which indicates that they are equally important to GEC. Second, when we only use the arc information, i.e., removing all labels, the recall drops sharply while the precision increases notably compared with the baseline. We speculate that the contribution from arc information is mainly on preventing GEC models from being misled by inappropriate context. Third, the full SynGEC model utilizing all three information achieves the best performance, which implies that they have intrinsic complementary strengths.

Error type performance. Figure 4 shows more fine-grained evaluation results on different error types on BEA-19-*dev*. The results support that syntactic information from GOPar is beneficial for most error types. Specifically, syntactic knowledge

[8] We use *biaffine-dep-roberta-en* model provided by Supar.

[9] For "w/o GED Labels", we force the parser to skip GED labels and select the syntax label with the highest probability when predicting. For "w/o Syntax Labels", we replace all syntax labels with "O" in the results. For "w/o All Labels", we do not feed the label embeddings into the DepGCN module.

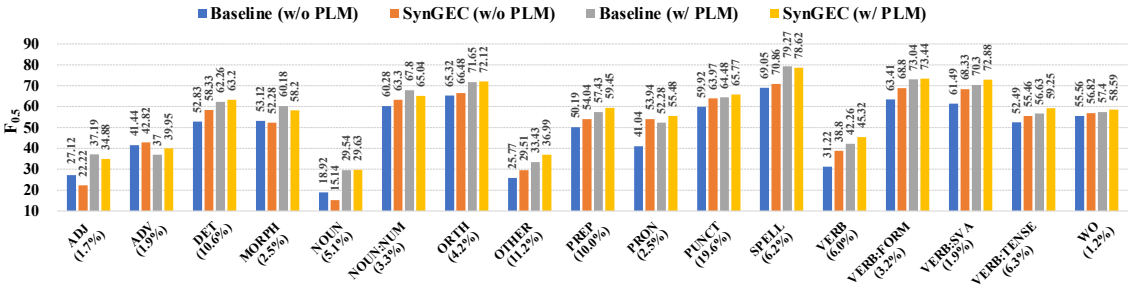


Figure 4: Performance on a selection of error types from ERRANT (Bryant et al., 2017) on BEA-19-*dev*. Numbers in parentheses represent the percentages of error types. We exclude error types that account for less than 1%.

	BEA-19-*dev* P/R/$F_{0.5}$	CoNLL-14-*test* P/R/$F_{0.5}$
	w/o PLM	
SynGEC	60.84/**39.76/55.01**	71.62/**47.92/65.17**
w/o GED Labels	59.47/38.03/53.44	72.28/45.43/64.64
w/o Syntax Labels	59.31/39.02/53.72	71.61/46.65/64.69
w/o All Labels	**61.62**/34.21/53.11	**73.16**/41.64/63.54
Baseline	57.99/35.77/51.58	69.60/42.37/61.67
	w/ PLM	
SynGEC	64.51/**45.73/59.62**	76.59/**51.09/69.64**
w/o GED Labels	63.59/45.23/58.82	76.79/50.45/69.53
w/o Syntax Labels	64.20/45.51/59.33	76.32/50.91/69.39
w/o All Labels	**64.90**/42.57/58.74	**77.01**/48.82/69.04
Baseline	63.09/44.80/58.32	75.61/50.42/68.74

Table 3: Effect of different labels in our scheme.

Dataset	#Sentences	%Error	Usage
Lang8	1,220,906	89.5	Training
HSK	15,6870	60.8	Training
MuCGEC-*Dev*	1,125	95.1	Validation
MuCGEC-*Test*	5,938	92.2	Testing

Table 4: Statistics of Chinese GEC datasets.

	PLM	Syntax	P	R	$F_{0.5}$
Zhang et al. (2022)	✓	✗	43.81	28.56	39.58
Baseline	✗	✗	43.79	25.93	38.49
SynGEC	✗	✓	46.88	27.68	**40.58**
Baseline	✓	✗	54.21	28.51	45.93
SynGEC	✓	✓	55.59	28.72	**46.83**

Table 5: Single-model results on MuCGEC-*test*.

significantly improves the GEC model's ability to correct context-aware errors, such as DET, PREP, PUCNT, VERB:SVA, and VERB:TENSE. Correcting such errors requires long-distance information, which syntax can effectively provide. The syntax also helps solve word-ordering (WO) errors, which need structural information to correct. Besides, the performance on PRON, OTHER, VERB is also substantially improved. Meanwhile, we note that a small subset of types are negatively affected, like ADJ, MORPH, SPELL, and NOUN. After more careful observation, we find that their corrections mainly depend on local information, where syntactic knowledge may not help much.

5 Experiments on Chinese GEC

Datasets and evaluation. We directly follow the recent work of Zhang et al. (2022), who propose a multi-reference multi-source Chinese GEC evaluation dataset named MuCGEC. We report P/R/$F_{0.5}$ values on MuCGEC-*test* using the official char-based evaluation tool. MuCGEC-*dev* is used for hyper-parameter tuning. For *training* data, we use the Chinese Lang8 dataset (Zhao et al., 2018) and HSK dataset (Zhang, 2009). The statistics of above-mentioned datasets are shown in Table 4.

Char-based GOPar. Current Chinese GEC models usually treat the input sentence as a character sequence and do not perform word segmentation. In contrast, syntactic parsers typically treat the input sentence as a word sequence. To handle this mismatch, we follow Yan et al. (2020) and build a char-based GOPar. The basic idea is to convert a word-based tree into a char-based one by letting each char depends on its right-hand one inside multi-char words.

Use of BART. We employ the recently proposed Chinese BART (Shao et al., 2021), which is originally implemented with the HuggingFace Transformers toolkit[10] (Wolf et al., 2020). We manage to wrap their code and use it on our Fairseq implementation, which is faster than the original implementation. More importantly, we find many common characters are missing in its vocabulary and thus add 3,866 Chinese characters and punctuation marks from Chinese Gigaword and Wikipedia corpora, leading to a substantial performance boost according to our preliminary experiments.

Results are presented in Table 5. When not

[10] https://huggingface.co/

using BART, our SynGEC outperforms the Transformer baseline by 2.09 $F_{0.5}$ score on MuCGEC-*test*. When using BART, our baseline already outperforms the previous SOTA system (Zhang et al., 2022), and SynGEC further improves the $F_{0.5}$ score by 0.90. These results indicate that our proposed SynGEC approach can be steadily effective for different languages.

6 Related Works

Grammatical error correction. Recent works mainly formulate GEC as a monolingual translation task and handle it with encoder-decoder-based MT models (Yuan and Briscoe, 2016; Junczys-Dowmunt et al., 2018), among which Transformer (Vaswani et al., 2017) has become a dominant paradigm. With the help of synthetic training data (Lichtarge et al., 2019; Yasunaga et al., 2021) and large PLMs (Kaneko et al., 2020; Katsumata and Komachi, 2020), Transformer-based GEC models have achieved SOTA performance on various benchmark datasets (Rothe et al., 2021; Stahlberg and Kumar, 2021).

Meanwhile, the sequence-to-edit (Seq2Edit) approach emerges as a competitive alternative, which predicts a sequence of edit operations to achieve correction (Gu et al., 2019; Awasthi et al., 2019; Omelianchuk et al., 2020). Although this work adopts the Transformer-based GEC models as the baseline, our SynGEC approach can also be applied to Seq2Edit models straightforwardly, which we leave to the future work.

Parsing ungrammatical sentences. Despite the success of syntactic parsing on clean sentences (Dozat and Manning, 2017; Zhang et al., 2020b), the performance of off-the-shelf parsers dramatically degrades when confronting ungrammatical sentences (Hashemi and Hwa, 2016). Previous studies mainly tackle this problem by annotating small-scale ungrammatical sentences and fine-tuning an off-the-shelf parser on them (Dickinson and Ragheb, 2009; Petrov and McDonald, 2012; Cahill, 2015; Berzak et al., 2016). In contrast, we propose to train a tailored parser on automatically generated syntax trees from parallel GEC data, which avoids the laborious manual annotation.

Syntax-enhanced GEC. Many previous works have demonstrated the effectiveness of utilizing syntactic information for various NLP tasks, such as machine translation (Bastings et al., 2017; Zhang et al., 2019), opinion role labeling (Zhang et al., 2020a), and semantic role labeling (Xia et al., 2019; Sachan et al., 2021). Meanwhile, we have found two recent works on syntax-enhanced GEC (Wan and Wan, 2021; Li et al., 2022). Both works directly produce the dependency tree of the input sentence using an off-the-shelf parser, without tailoring parsers for ungrammatical sentences. They both use graph attention networks (GAT) for tree encoding (Velickovic et al., 2018). Besides the dependency tree, Li et al. (2022) exploits the constituent tree of the input sentence as well.

Compared with the above two works, the major contribution of our work is directly dealing with the severe performance drop issue via our tailored GOPar. We adopt GCN for tree encoding because our preliminary experiments show that it achieves similar performance but is faster. Moreover, our baseline GEC models achieve much higher performance than theirs, as shown in Table 2.

7 Conclusions

This paper presents a SynGEC approach that effectively incorporates syntactic knowledge into GEC models. The key idea is adjusting vanilla parsers to accommodate ungrammatical sentences. To achieve this goal, we first extend the standard syntactic annotation scheme to use a unified tree structure to encode both grammatical errors and syntactic structure. Then we obtain high-quality parse trees of ungrammatical sentences by projecting target-sentence trees into the source side in GEC training data, which are used for training our tailored GOPar. To incorporate parse trees produced by GOPar, we present a DepGCN-based GEC model based on Transformer. Experiments on three English/Chinese datasets show that our proposed SynGEC approach is effective and achieves new SOTA results under comparable settings.

Limitations and future work. We observe two possible limitations of our present work. First, off-the-shelf parsers may still produce noisy parse trees for the target correct sentences in the GEC training data, which further leads to noise in our projected trees for the source sentences. Second, so far, we only use three coarse-grained labels to distinguish grammatical errors in our annotation scheme, while fine-grained categories may further benefit GEC. As discussed above, both limitations may be strengthened by integrating ideas and resources achieved by previous works on manually annotating syntactic trees for ungrammatical sentences.

References

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

Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Jasmijn Bastings, Ivan Titov, Wilker Aziz, Diego Marcheggiani, and Khalil Sima'an. 2017. Graph convolutional encoders for syntax-aware neural machine translation. In *Proceedings of EMNLP*, pages 1957–1967.

Samuel Bell, Helen Yannakoudakis, and Marek Rei. 2019. Context is key: Grammatical error detection with contextual word representations. In *Proceedings of BEA@ACL*, pages 103–115.

Yevgeni Berzak, Jessica Kenney, Carolyn Spadine, Jing Xian Wang, Lucia Lam, Keiko Sophie Mori, Sebastian Garza, and Boris Katz. 2016. Universal dependencies for learner english. In *Proceedings of ACL*, pages 737–746.

Christopher Bryant, Mariano Felice, Øistein E Andersen, and Ted Briscoe. 2019. The bea-2019 shared task on grammatical error correction. In *Proceedings of 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 *Proceedings of ACL*, pages 793–805.

Aoife Cahill. 2015. Parsing learner text: to shoehorn or not to shoehorn. In *Proceedings of Linguistic Annotation Workshop*, pages 144–147.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: pre-training text encoders as discriminators rather than generators. In *Proceedings of ICLR*.

Yiming Cui, Wanxiang Che, Ting Liu, Bing Qin, Shijin Wang, and Guoping Hu. 2020. Revisiting pre-trained models for chinese natural language processing. In *Proceedings of EMNLP: findings*, pages 657–668.

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

Daniel Dahlmeier, Hwee Tou Ng, and Siew Mei Wu. 2013. Building a large annotated corpus of learner english: The nus corpus of learner english. In *Proceedings of BEA@NAACL-HLT*, pages 22–31.

Markus Dickinson and Marwa Ragheb. 2009. Dependency annotation for learner corpora. In *Proceedings of International Workshop on Treebanks and Linguistic Theories*, page 59.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *Proceedings of ICLR*.

Jennifer Foster, Joachim Wagner, and Josef Van Genabith. 2008. Adapting a wsj-trained parser to grammatically noisy text. In *Proceedings of ACL (short)*, pages 221–224.

Roman Grundkiewicz, Christopher Bryant, and Mariano Felice. 2020. A crash course in automatic grammatical error correction. In *Proceedings of COLING: Tutorial Abstracts*, pages 33–38.

Jiatao Gu, Changhan Wang, and Jake Zhao Junbo. 2019. Levenshtein transformer. In *Proceedings of NIPS*, pages 11181–11191.

Homa B Hashemi and Rebecca Hwa. 2016. An evaluation of parser robustness for ungrammatical sentences. In *Proceedings of EMNLP*, pages 1765–1774.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Identity mappings in deep residual networks. In *Proceedings of ECCV*, pages 630–645.

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 NAACL-HLT*, pages 595–606.

Masahiro Kaneko and Mamoru Komachi. 2019. Multi-head multi-layer attention to deep language representations for grammatical error detection. *Computing Research Repository*, pages 883–891.

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 ACL*, pages 4248–4254.

Satoru Katsumata and Mamoru Komachi. 2020. Stronger baselines for grammatical error correction using a pretrained encoder-decoder model. In *Proceedings of AACL*, pages 827–832.

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

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *Proceedings of ICLR*.

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 EMNLP-IJCNLP*, pages 1236–1242.

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 ACL*, pages 7871–7880.

Zuchao Li, Kevin Parnow, and Hai Zhao. 2022. Incorporating rich syntax information in grammatical error correction. *Information Processing and Management*.

Jared Lichtarge, Chris Alberti, and Shankar Kumar. 2020. Data weighted training strategies for grammatical error correction. *TACL*, pages 634–646.

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

Masato Mita, Shun Kiyono, Masahiro Kaneko, Jun Suzuki, and Kentaro Inui. 2020. A self-refinement strategy for noise reduction in grammatical error correction. In *Proceedings of EMNLP (Findings)*, pages 267–280.

Ryo Nagata and Keisuke Sakaguchi. 2016. Phrase structure annotation and parsing for learner english. In *Proceedings of ACL*, pages 1837–1847.

Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of ICML*, pages 807–814.

Courtney Napoles, Keisuke Sakaguchi, Matt Post, and Joel Tetreault. 2015. Ground truth for grammatical error correction metrics. In *Proceedings of ACL (short)*, pages 588–593.

Courtney Napoles, Keisuke Sakaguchi, and Joel Tetreault. 2017. Jfleg: A fluency corpus and benchmark for grammatical error correction. In *Proceedings of 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 CoNLL: Shared Task*, pages 1–14.

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

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT(Demo)*, pages 48–53.

Slav Petrov and Ryan McDonald. 2012. Overview of the 2012 shared task on parsing the web.

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

Marek Rei and Helen Yannakoudakis. 2016. Compositional sequence labeling models for error detection in learner writing. In *Proceedings of ACL*, pages 1181–1191.

Sascha Rothe, Jonathan Mallinson, Eric Malmi, Sebastian Krause, and Aliaksei Severyn. 2021. A simple recipe for multilingual grammatical error correction. In *Proceedings of ACL-IJCNLP*, pages 702–707.

Devendra Sachan, Yuhao Zhang, Peng Qi, and William L Hamilton. 2021. Do syntax trees help pre-trained transformers extract information? In *Proceedings of EACL*, pages 2647–2661.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of ACL*, pages 1715–1725.

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

Felix Stahlberg and Shankar Kumar. 2021. Synthetic data generation for grammatical error correction with tagged corruption models. In *Proceedings of BEA@EACL*, pages 37–47.

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

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of ICCV*, pages 2818–2826.

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 NIPS*, pages 5998–6008.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *Proceedings of ICLR*.

Zhaohong Wan and Xiaojun Wan. 2021. A syntax-guided grammatical error correction model with dependency tree correction. *arXiv preprint arXiv:2111.03294*.

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

Ronald J Williams and David Zipser. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280.

Sam Wiseman and Alexander M. Rush. 2016. Sequence-to-sequence learning as beam-search optimization. In *Proceedings of EMNLP*, pages 1296–1306.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of EMNLP (Demo)*, pages 38–45.

Qingrong Xia, Zhenghua Li, Min Zhang, Meishan Zhang, Guohong Fu, Rui Wang, and Luo Si. 2019. Syntax-aware neural semantic role labeling. In *Proceedings of AAAI*, pages 7305–7313.

Hang Yan, Xipeng Qiu, and Xuanjing Huang. 2020. A graph-based model for joint chinese word segmentation and dependency parsing. *TACL*, pages 78–92.

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

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

Zheng Yuan and Ted Briscoe. 2016. Grammatical error correction using neural machine translation. In *Proceedings of NAACL-HLT*, pages 380–386.

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

Baolin Zhang. 2009. Features and functions of the hsk dynamic composition corpus. *International Chinese Language Education*, 4:71–79.

Bo Zhang, Yue Zhang, Rui Wang, Zhenghua Li, and Min Zhang. 2020a. Syntax-aware opinion role labeling with dependency graph convolutional networks. In *Proceedings of ACL*, pages 3249–3258.

Meishan Zhang, Zhenghua Li, Guohong Fu, and Min Zhang. 2019. Syntax-enhanced neural machine translation with syntax-aware word representations. In *Proceedings of NAACL-HLT*, pages 1151–1161.

Yu Zhang, Zhenghua Li, and Min Zhang. 2020b. Efficient second-order treecrf for neural dependency parsing. In *Proceedings of ACL*, pages 3295–3305.

Yue Zhang, Zhenghua Li, Zuyi Bao, Jiacheng Li, Bo Zhang, Chen Li, Fei Huang, and Min Zhang. 2022. MuCGEC: a multi-reference multi-source evaluation dataset for chinese grammatical error correction. In *Proceedings of NAACL-HLT*.

Yuanyuan Zhao, Nan Jiang, Weiwei Sun, and Xiaojun Wan. 2018. Overview of the nlpcc 2018 shared task: Grammatical error correction. In *CCF International Conference on Natural Language Processing and Chinese Computing (NLPCC)*, pages 439–445.

	Train-set	BEA-19-*dev* $F_{0.5}$	FCE-*test* $F_{0.5}$
Bell et al. (2019)	FCE-train	48.50	57.28
Kaneko and Komachi (2019)	FCE-train	–	61.65
Yuan et al. (2021)	FCE-train	65.54	72.93
GOPar	FCE-train	**66.32**	**74.10**
GOPar	CLang8	**72.13**	**71.53**

Table 6: Binary GED performance.

A Hyper-parameters

The main hyper-parameters used in our experiments are presented in Table 7. When not using PLMs, the total training time is about 3 hours. When using PLMs, the training costs about 7 hours.

B Download links of PLMs

The download links of PLMs used in our experiments are listed below. We employ ELECTRA (Clark et al., 2020; Cui et al., 2020) to build GOPar and BART (Lewis et al., 2020; Shao et al., 2021) to enhance our GEC model.

- ELECTRA-English-Large.
- ELECTRA-Chinese-Large.
- BART-English-Large.
- BART-Chinese-Large.

Configuration	Value
Pre-training	
Base architecture	Transformer-base (w/o PLM) Transformer-large (w/ PLM)
Pretrained Language model	BART-large (Lewis et al., 2020)
Number of epochs	60
Devices	8 Tesla V100 GPU (32GB)
Batch size per GPU	8096 tokens
Optimizer	Adam (Kingma and Ba, 2014) ($\beta_1 = 0.9, \beta_2 = 0.98, \epsilon = 1 \times 10^{-8}$)
Learning rate	5×10^{-4}
Warmup updates	4000
Max source length	64 (English); 128 (Chinese)
Number of DepGCN layers	3
Dual context aggregation β	0.5
Loss function	Label smoothed cross entropy (label-smoothing=0.1) (Szegedy et al., 2016)
Dropout	0.1 (w/o PLM); 0.3 (w/ PLM)
Dropout-src	0.2
Fine-tuning	
Learning rate	5×10^{-5}
Warmup updates	1000
Generation	
Beam size	12
Max input length	64 (English); 128 (Chinese)

Table 7: Hyper-parameter values used in our experiments.

	PLM	Syntax	GLUE	Δ
Baseline	✗	✗	58.15	-
SynGEC	✗	✓	60.14	+1.99
Baseline	✓	✗	61.53	-
SynGEC	✓	✓	62.15	+0.62

Table 8: The GLUE scores of different models on JFLEG benchmark.

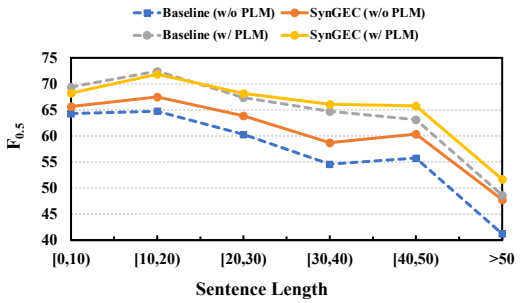


Figure 5: The effect of input sentence length.

C The GED ability of GOPar

We evaluate the binary Grammatical Error Detection (GED) performance of GOPar on two mainstream GED dataset, i.e., BEA-19-dev (Bryant et al., 2019) and FCE-test (Yannakoudakis et al., 2011). We follow Rei and Yannakoudakis (2016) and report token-level P/R/F values for detecting incorrect labels. Table 6 shows the performance of GOPar and other leading GED models. When using the same training data, GOPar has a superior ability to detect grammatical errors. This phenomenon is very interesting and worthy of more in-depth study.

D Experiments on JFLEG

JFLEG (Napoles et al., 2017) is an English GEC evaluation dataset which focuses on fluency and uses the GLUE score (Napoles et al., 2015) as the evaluation metric. We evaluate the baseline and the SynGEC approach in Table 2 on JFLEG. Since JFLEG's scale is relatively small (only 747 sentences), we choose to present the results in the appendix. From Table 8, we can see that the syntactic knowledge still continuously improves the GEC performance over baselines with/without PLMs.

E Performance Regarding the Input Sentence Length

Figure 5 shows performance on CoNLL-14-*test* in terms of input lengths. The baselines heavily degenerate when encountered with long sentences.

After introducing syntactic knowledge, it is clear that larger improvements are obtained for longer sentences in both scenarios with or without the PLM, as syntax helps to solve errors related to remote information, which are common in long sentences.