LET: Leveraging Error Type Information for Grammatical Error Correction

Lingyu Yang*, Hongjia Li*, Lei Li, Chengyin Xu, Shutao Xia, Chun Yuan†
Tsinghua University, Beijing, China
{yly20, lhj20, lei-li18, xucy20}@mails.tsinghua.edu.cn
{xiast, yuanc}@sz.tsinghua.edu.cn

Abstract

Grammatical error correction (GEC) aims to correct errors in given sentences and is significant to many downstream natural language understanding tasks. Recent work introduces the idea of grammatical error detection (GED) to improve the GEC task performance. In contrast, these explicit multi-stage works propagate and amplify the problem of misclassification of the GED module. To introduce more convincing error type information, we propose an end-to-end framework in this paper, which Leverages Error Type (LET) information in the generation process. First, the input text is fed into a classification module to obtain the error type corresponding to each token. Then, we introduce the category information into the decoder's input and cross-attention module in two ways, respectively. Experiments on various datasets show that our proposed method outperforms existing methods by a clear margin.

1 Introduction

The grammatical error correction (GEC) task aims to correct grammatical errors in natural language texts, including spelling, punctuation, grammar, word selection, and more. As shown in Figure 1, a GEC model receives text containing errors and produces its corrected version.

Current GEC algorithms are mainly divided into two categories: detection-based models and end-to-end generative models.

Detection-based models treat GEC as a token classification problem (Omelianchuk et al., 2020). By classifying each token in the sentence, we can make a detailed transformation according to the classification result to obtain the modified sentence. This method has strong interpretability and a transparent error correction process. However, to achieve precise error correction, it is necessary first to identify and classify all possible grammatical errors. The training data is then manually annotated based on the error categories, which is labor-intensive.

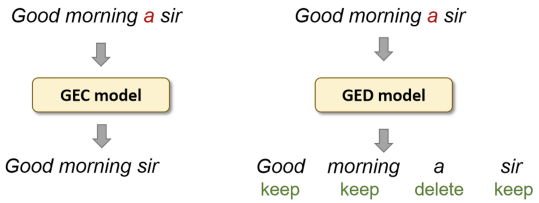

Figure 1: An example of grammatical error correction and detection.

To avoid manually designing wrong categories and labeling data, many works (Yuan and Felice, 2013a; Yuan and Briscoe, 2016) have built end-to-end generative GEC systems from the perspective of Machine Translation (MT), which is also the current mainstream method. In this approach, erroneous sentences correspond to the source language, and error-free sentences correspond to the target language.

Most recent generative models (Raheja and Alikaniotis, 2020; Yuan et al., 2021) are based on the Transformer encoder-decoder architecture (Vaswani et al., 2017). They also achieve competitive results compared to detection-based models. The most significant advantage of the end-to-end generative model is that we do not need to design complex error categories manually or perform labor-intensive labeling work on the data. We can only use parallel corpora to train the model.

Recent works (Wang et al., 2020; Chen et al., 2020) have shown that if the error type results obtained in the grammatical error detect (GED) task are introduced into the generative model in some form, the error correction ability of the model will be further improved. This is because the entire training and inference process can be viewed as a black-box operation in an end-to-end generative model. Furthermore, the model can generate more

*These authors contributed equally to this work.
†Corresponding authors.

Findings of the Association for Computational Linguistics: ACL 2023, pages 5986–5998

accurate results if additional information guides this process (e.g., The classification result of some location is "delete").

Yuan et al. (2021) extends the Transformer encoder-decoder model based on introducing error type information. They classify input tokens into different error types, transform them into representations, and feed them into the decoder's cross-attention module. However, this method suffers from two fundamental limitations:

1) **Error propagation**. Each token is mapped into a one-hot classification vector in the first process. If there is a misclassification in the results, it will be passed on and negatively influence the following parts.

2) **Mismatched cross attention**. In the original transformer decoder block, the input Q and K of the cross-attention module are from the semantic space of tokens. However, these inputs are from the semantic space of the error type information and the original tokens, respectively. This mismatch can lead to a reduction in the representation of the model.

Therefore, to solve the above problems, we propose a simple yet novel generative model to improve the performance of GEC, termed LET (Leveraging Error Type information).

First, we utilize the intermediate representation of the error type classification module as the error type vector. It would not discard the probabilities of other classes, even if their values are small. This operation ensures more convincing guidance of the type vectors to the generated modules.

Second, to discard the mismatch in the cross-attention module, we transfer the input from the previous sub-layer in the decoder to the classification vector. Thus, both parts of the input are in the same semantic space. Therefore, the cross-attention for them is more reasonable.

In summary, our contributions can be summarized in the following points:

1) We propose a novel sequence-to-sequence model which realizes the alignment of error type for GEC. This model improves the effect of this task with much more fine-grained error detection.

2) We demonstrate how GED benefits the correction task by introducing the error type information into the input module and the cross-attention module of the decoder in two ways.

3) Experimental results on multiple datasets show that our proposed method achieves state-of-the-art results.

2 Related Work

Much progress in the GEC task can be attributed to transforming the problem into a machine translation task (Brockett et al., 2006) from an ungrammatical source sentence to a grammatical target sentence. Early GEC-MT methods leveraged phrase-based statistical machine translation (PB-SMT) (Yuan and Felice, 2013b). With the rapid development of related work on machine translation, statistical machine translation (SMT) and neural machine translation (NMT) have been successfully applied to various task-specific adaptations of GEC (Felice et al., 2014;Yuan and Briscoe, 2016;Junczys-Dowmunt et al., 2018) With the introduction of transformer architectures, this approach rapidly evolved to powerful Transformer-based seq2seq models (Vaswani et al., 2017). Transformer-based models autoregressively capture the complete dependency among output tokens (Yuan et al., 2019). Grundkiewicz et al. (2019) leveraged a Transformer model pre-trained on synthetic GEC data. Several improvement strategies of BERT were also adopted in the GEC model (Kaneko et al., 2020). With the development of large-scale pre-trained models recently, Rothe et al. (2021) built their system on top of T5(Xue et al., 2021) and reached new state-of-the-art results.

Grammatical Error Detection is usually formulated as a sequence tagging task, where each erroneous token is assigned with an error type, e.g., selection errors and redundant words. Early GED methods mainly used rules to identify specific sentence error types, such as preposition errors (Tetreault and Chodorow, 2008). With the development of neural networks, Rei and Yannakoudakis (2016) presented the first work using a neural approach and framed GED as a binary sequence labeling problem, classifying each token in a sentence as correct or incorrect. Sequence labeling methods are widely used for GED, such as feature-based statistical models (Chang et al., 2012) and neural models (Fu et al., 2018). Due to the effectiveness of BERT (Devlin et al., 2019) in many other NLP applications, recent studies adopt BERT as the basic

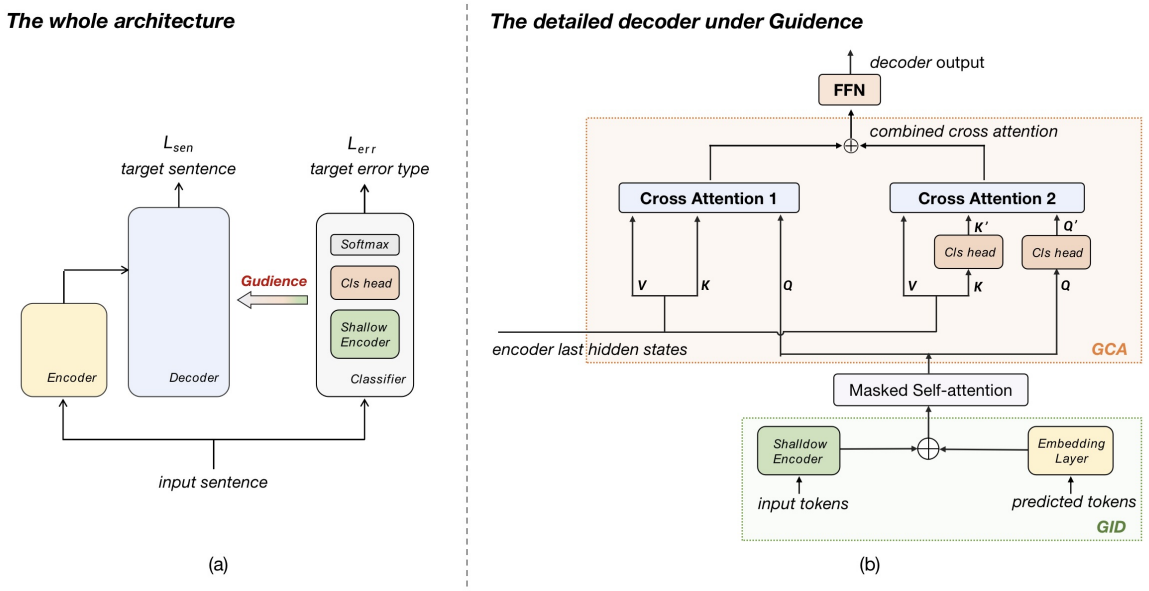

Figure 2: Architecture of LET (Leveraging Error Type information). (a) shows the overview architecture of LET, and (b) describes the detailed components in the cross-attention module.

architecture of GED models(Li and Shi, 2021).

Recent work has attempted to explore a different approach to using GED in GEC, which aims to use the detection results of GED to guide GEC generation. Yuan et al. (2019) introduced token-level and sentence-level GED as auxiliary tasks when training for GEC. Zhao et al. (2019) employed multi-task learning to utilize the detection results of GED to guide GEC generation. Similarly, Chen et al. (2020) fine-tuned RoBERTa (Zhuang et al., 2021) for GED and improved the efficiency for GEC by dividing the task into two sub-tasks: Erroneous Span Detection and Erroneous Span Correction. (Yuan et al., 2021) treated GED as a sequence labeling task and GEC as a sequence-to-sequence task and additionally investigated ways to use multi-class GED predictions to inform GEC.

3 Method

In this section, we first describe the problem definition and the basic model, our baseline. Then we describe the LET (Leveraging Error Type information) model, which explicitly applies the classification information (error types) of tokens to guide the generative model to generate better-corrected sentences. The whole architecture of LET is shown in Figure 2.

3.1 Problem Definition

Given a sentence that may contain erroneous tokens $U = \{u_i\}^N$, the target of GEC is to correct the input sentence and output the corrected sentence $C = \{c_i\}^M$. N and M are the input and output sequence length, respectively.

3.2 Backbone

We use BART (Lewis et al., 2020) as the backbone model of our end-to-end GEC system. BART is a denoising autoencoder that maps the noisy text to the correct form. It is implemented as a sequence-to-sequence model with a bidirectional encoder over corrupted text and a left-to-right autoregressive decoder (Vaswani et al., 2017). The word embedding layer is represented as Emb. The encoder and decoder are represented as EC and DC, respectively. The process of encoding and decoding can be formulated as:

$$E_U = EC(U) \quad (1)$$

$$C = DC(E_U) \quad (2)$$

where E_U is the output of the encoder EC.

3.3 Grammatical Error Detection

We aim to obtain the error type classification of each token in the sentence by the sequence labeling task. In practice, we construct this classifier with three parts. First, a two-layer transformer encoder block EC' is designed to encode the input sentence

U and obtain the long error type representation R_U^{long}, of which the embedding dimension is the same as the word embedding, such as 768 or 512. This procedure can be formulated as:

$$R_U^{long} = EC^{'}(U) \tag{3}$$

Then, a two-layer fully-connected network FF aims to transform the long error type representation to short error type representation R_U^{short}:

$$R_U^{short} = FF(R_U^{long}) \tag{4}$$

where the dimension of the short one is the number of error types, such as 4, 25 or 55.

Finally, the error type can be calculated by a Softmax layer SM:

$$Y = SM(R_U^{short}) \tag{5}$$

where $Y = \{y_i\}^N$ is the label sequence of N tokens.

3.4 GID: Guided Input of the Decoder

Naturally, after the generation module autoregressively decodes the tokens at some time step, if there is the error type information of the next time step of the original sentence, the generation module may make the correct decision more easily. For example, considering the error type of the next token in the original sentence is "Delete" (This token is redundant and needs to be deleted), the generation module will delete the next token by greater probability after receiving the information indicating "Delete."

Metaphorically speaking, we can compare the decoder to a little boy and the decoding process to the boy solving a complex math problem. If a reference material is available to guide the problem-solving process, the little boy will undoubtedly find it easier to arrive at the correct answer. This reference material is what we refer to as "additional guiding information" in this context.

Formally, at time step t, we have obtained the output of the last time step, which is represented as p_{t-1}. Therefore, we take two elements as the input of this GID module: 1) Emb_{t-1}: the word embedding of p_{t-1}; 2) R_t^{long}: the corresponding long error type representation of the token u_t. Therefore, we obtain T_i, the output of GID and also the input of the decoder DC, by a direct point-wise add operation:

$$T_i = Emb_{t-1} + R_t^{long} \tag{6}$$

3.5 GCA: Guided Cross Attention module

In addition to the above approach, we also want to introduce error type information in the cross-attention module.

Cross Attention 1 In the original transformer, the cross attention module in the decoder layer performs attention weighting calculations on the token embedding output by the encoder and the output of the previous self-attention module. The calculation formula is expressed as:

$$E^{CA1} = softmax(\frac{QK^T}{\sqrt{d_K}})V \tag{7}$$

where E^{CA1} is the output of the Cross Attention 1 module. Here, Q represents the representation vector output by the input tokens of the current decoder after passing through the previous self-attention module. K represents the representation vector output by all the input tokens after passing through the stacked encoder. V is a copy of K. In practice, $Q/K/V$ are firstly mapped to different representation spaces by matrices $W_q/W_k/W_v$, respectively.

In Equation 7, by performing the scaled dot-product operation on Q and K, the weight parameter for weighted summation of V is obtained. Previous work (Lee et al., 2018; Li et al., 2020) has shown that such an operation is to align the tokens input by the encoder and decoder at the semantic level, so that the decoder is able to generate accurate and reasonable results.

Cross Attention 2 We describe alignment at the semantic level in the last subsection. However, more than this alignment is needed. What about alignment at the error type level? That is, we use the existing detection module to classify the original Q and K to error types and then use the obtained results to replace Q and K in Equation 7, which realizes the alignment at the error type level.

Specifically, as shown in Figure 2, we utilize classification head FF to classify Q and K, and obtain their short error type representation vectors Q' and K' respectively:

$$Q^{'} = FF(Q) \tag{8}$$

$$K^{'} = FF(K) \tag{9}$$

where the dimension of Q' and K' depends on the classification category of the detection task. These

Model	with GED	BEA-test			CoNLL-2014		
		Precision	Recall	$F_{0.5}$	Precision	Recall	$F_{0.5}$
Constrained Data							
Lewis et al. (2020)	✘	48.4	41.7	47.2	50.6	26.3	43.1
Raheja and Alikaniotis (2020)	✘	53.8	36.5	49.1	64.7	22.6	47.1
Kaneko et al. (2020)	✔	58.1	44.8	54.8	63.6	33.0	53.6
Yuan et al. (2021)	✔	60.8	50.8	58.5	60.4	39.0	54.4
LET (ours)	✔	61.8	52.1	**59.5**	61.2	40.9	**55.6**
Unconstrained Data							
Ji et al. (2017)	✘	-	-	-	-	-	45.2
Ge et al. (2018)	✘	-	-	-	61.2	37.9	54.5
Kiyono et al. (2019)	✘	65.5	59.4	64.2	67.9	44.1	61.3
Lichtarge et al. (2020)	✘	67.6	62.5	66.5	69.4	43.9	62.1
Wan et al. (2020)	✘	66.9	60.6	65.5	69.5	47.3	63.5
Stahlberg and Kumar (2021)	✘	72.1	64.4	70.4	72.8	49.5	66.6
Yuan and Bryant (2021)	✘	-	-	-	74.3	39.0	62.9
Zhao et al. (2019)	✔	-	-	-	67.7	40.6	59.8
Yuan et al. (2019)	✔	70.5	55.1	66.8	-	-	-
Kaneko et al. (2020)	✔	67.1	60.1	65.6	69.2	45.6	62.6
Chen et al. (2020)	✔	70.4	55.9	66.9	72.6	27.2	61.0
Wang et al. (2020)	✔	-	-	-	65.0	33.5	54.6
Yuan et al. (2021)	✔	73.3	61.5	70.6	71.3	44.3	63.5
LET (ours)	✔	74.6	62.9	**71.9**	71.7	45.6	**64.3**
Omelianchuk et al. (2020)	✘	79.2	53.9	72.4	77.5	40.1	65.3

Table 1: Evaluation results using ERRANT on BEA-test and M^2 (Dahlmeier and Ng, 2012) on CoNLL-2014. Methods with Grammatical Error Detection (GED) module are marked with a check mark. On the contrary, pure sequence-to-sequence models and sequence labelling systems (only Omelianchuk et al. (2020)) are labeled with a cross mark. Only public BEA-2019 data is used in the training process of all constrained systems, while unconstrained systems are variously trained on private and/or artificial data.

representation vectors can be viewed as representations of error types. Therefore, applying cross attention to them realizes the alignment of the tokens input by the encoder and the decoder at the error-type level. The modified self-attention equation can be formulated as follows:

$$E^{CA2} = softmax(\frac{Q^{'}K^{'T}}{\sqrt{d_{K^{'}}}})V \quad (10)$$

where $d_{K^{'}}$ is the dimension of $K^{'}$, E^{CA2} is the output of this Cross Attention 2 module.

Combination of CA1 & CA2 Then, we combine the output of two cross-attention modules at pointwise. Nevertheless, before this, we need to define the weight of each one. Therefore, we calculate the dynamic Weighting factor λ:

$$\lambda = \sigma(W[E^{CA1}; E^{CA2}] + b) \quad (11)$$

where σ is the logistic Sigmoid function and W and b are learnable parameters.

Then we obtain the combined output E^{GCA} as follows:

$$E^{GCA} = \lambda E^{CA1} + (1 - \lambda)E^{CA2} \quad (12)$$

After this sub-module, E^{GCA} is used as the input to the next sub-layer. Ultimately, the forward computation and back-propagation of the entire model are trained like the regular encoder-decoder model.

3.6 Loss function

The total loss contains two parts:

1) L_{err}: The cross-entropy of the predicted error types and the ground truth of token-level labels.

2) L_{sen}: The cross-entropy of the output corrected sentences and corresponding target sentences.

Total loss is defined as:

$$L = \alpha L_{err} + (1 - \alpha) L_{sen} \quad (13)$$

where $\alpha \in [0, 1]$ is a hyper-parameter.

4 Experiments

To test the performance of the LET system, we conduct evaluation experiments on two mainstream GEC benchmarks: BEA-test (Bryant et al., 2019) and CoNLL-2014 (Ng et al., 2014) and compare with previous state-of-the-art approaches.

4.1 Datasets

Following previous work, we use five datasets:

- Lang-8 Corpus (Mizumoto et al., 2011)
- Cambridge Learner Corpus (CLC) (Nicholls, 2003)
- First Certificate in English (FCE) corpus (Yannakoudakis et al., 2011)
- National University of Singapore Corpus of Learner English (NUCLE) (Dahlmeier et al., 2013)
- Cambridge English Write & Improve + LOCNESS (W&I) corpus (Bryant et al., 2019)

Following the training process of previous work (Kiyono et al., 2019; Lichtarge et al., 2020; Yuan et al., 2021), we pre-train two LET systems on public Lang-8 Corpus (under the constrained setting) and the CLC dataset (under the unconstrained setting), then fine-tune them on the same three datasets, including W&I, FCE, and NUCLE. Finally, we train all modules in LET simultaneously, jointly optimizing L_{err} and L_{sen} based on Equation 13.

4.2 Error Type Annotations

We obtain error type annotations in these corpora by the ERRANT (Bryant, 2019) annotation toolkit, which can pre-process sentences and standardize tokens into error type annotations. The kind of error types can be binary classes, 4-classes consisting of basic operations, 25-classes consisting of word types and 55-classes combining the above tags. Table 2 shows the error type annotations in different numbers of classes.

N.C	Error Type Annotations
2	right and wrong
4	insert, delete, replace and keep
25	insert(noun), insert(verb tense), insert(prep) replace(noun), replace(verb tense), keep, delete, etc.
55	insert(M:DET), insert(U:PREP), insert(R:VERB:TENSE), replace(R:VERB:TENSE), replace(M:DET) , keep, delete, etc.

Table 2: Error Type Annotations. N.C: Number of Classes

4.3 Experiment Setup

The LET model, which is implemented with Transformers* (Wolf et al., 2020), consists of 6 encoder layers, 6 decoder layers, and a shared classification head.

The dimension of embedding is set to 768, and the batch size is set to 32. The maximum sequence length is 1024, and we pad sequences with the longest length in the batch. We train the model with Adam optimizer, and the learning rate is set to 2e-5. The weight factor α in Equation 13 is set to 0.2. The evaluation metric of text generation contains precision, recall, and $F_{0.5}$ score. We train the model on 4 Nvidia V100 GPUs. It takes about 4 hours to train the model in one epoch.

4.4 Results analysis

We report the experimental results of various methods in Table 1. The experimental results demonstrate the effectiveness of LET.

Overall performance As shown in Table 1, the proposed LET network outperforms most previous state-of-the-art methods on two mainstream GEC benchmarks: BEA-test and CoNLL-2014 under constrained and unconstrained settings.

Constrained setting Compared with the previous SOTA seq2seq model Yuan et al. (2021), LET improve $F_{0.5}$ score with 1% and 1.2% on BEA-test and CoNLL-2014, respectively. Compared with other seq2seq models, our work has achieved more obvious improvements based on the same experimental data.

Unconstrained setting On BEA-test, compared with Yuan et al. (2021), LET is at least 1.3% better

*https://github.com/huggingface/transformers

N.C	Algorithm	P	R	$F_{0.5}$
-	Baseline	55.1	44.3	52.5
2	+ GID	55.8	44.6	53.1
	+ GCA	57.4	42.7	53.7
	+ GID & GCA	58.1	44.7	54.8
4	+ GID	56.1	44.6	53.4
	+ GCA	58.2	41.6	54.0
	+ GID & GCA	58.8	45.0	54.2
25	+ GID	55.8	44.3	53.0
	+ GCA	58.2	42.9	54.3
	+ GID & GCA	58.7	45.1	55.4
55	+ GID	55.2	44.0	52.5
	+ GCA	58.7	42.9	54.7
	+ GID & GCA	**58.9**	**45.2**	**55.5**

Table 3: Results based on ablated modules and different number of error types. N.C: Number of Classes

Variations	Prec	Rec	$F_{0.5}$
Baseline	55.1	44.3	52.5
Static weights	55.8	**44.6**	53.1
Dynamic weights	**58.2**	41.6	**54.0**

Table 4: Results of two weighting ways of combining CA1 and CA2.

than the state-of-the-art model on three key metrics. On CoNLL-2014, LET also achieves significant improvements. Notably, compared to precision (+1.3%/+0.4%), our method improves the recall score more (+1.4%/+1.3%). Under the Constrained setting, there is also a similar data distribution. It shows that the model is better at recalling the correct editing operations under the combined effect of our multiple innovations.

Compared with models without a GED module, our LET is less than Stahlberg and Kumar (2021). The possible reason is that they used more data to train the model.

As shown in the last line of Table 1, Omelianchuk et al. (2020) outperforms all systems above. Due to more data, fine-grained labels, and multiple ensemble strategies, this sequence-tagging model has taken the first place in GEC for a long time.

5 Discussion

We discuss many details of the model in more depth in this section. Unless stated otherwise, all experiments in this section are tested on the BEA-dev dataset under the constrained data setting.

5.1 Ablation study

We explore the effect of each component in the whole LET (Leveraging Error Type) system. We compare the Bart-base as the Baseline (6 encoder-layers, 6 decoder-layers, 768 hidden-states, 16 attention-heads, and 139M parameters). As shown in Table 3, GID and GCA achieve higher values than the Baseline on three key metrics no matter how many error types exist. Moreover, the combination of them even obtains more improvement, demonstrating the effectiveness of the proposed two modules.

5.2 Results on different error categories

In this section, we explore the performance of LET on different error categories. We use the same pre-training and fine-tuning data splits for the baseline model but with no additional GED input for fine-tuning, which follows the standard encoder-decoder GEC training procedure. As shown in Table 3, the results demonstrate the efficacy of the multi-encoder GEC model: adding GED predictions as auxiliary input yields a consistent statistically significant improvement in performance over the baseline. Our best system uses the 55-class GED predictions, achieving 55.5 $F_{0.5}$. The reason may be that the 55-class system represents the best compromise between label informativeness and model reliability.

Unlike the optimal scheme of LET, GID achieves the best result (53.4 $F_{0.5}$) in the setting of 4-class GED prediction. Notably, GID using the 2-class GED predictions (binary predictions) outperformed the same model using the 55-class GED predictions. This is because 2-class GED predictions are less informative but more reliable. After all, there are only two classes, while 25-class and 55-class predictions tend to be more informative but less reliable because of the increased difficulty in predicting sparser classes. This also shows that the GID model lacking the alignment of the error type is not good at using too subdivided error type guidance information, and it can also be inferred that the GID model does not make full use of the error type information effectively.

Notably, similar to LET, GCA achieves the best result (54.7 $F_{0.5}$) in the setting of 55-class GED prediction. Meanwhile, the experiment shows that with the increase in the number of error type categories, the GCA model's effect gradually improves.

Algorithm	Variation	Precision	Recall	$F_{0.5}$
Baseline	-	55.1	44.3	52.5
+ Guided Cross Attention	-	**58.7**	42.9	**54.7**
	ablation study A	54.9	**44.4**	52.4
	ablation study B	55.2	44.3	54.6

Table 5: Discussion on the number of parameters. Baseline: There is no new cross attention module. ablation study A: We construct a new classifier, which is a two-layer fully connected layer with the hidden size of (768->768->55). The parameters are initialized randomly. ablation study B: We also construct a new classifier with the hidden size of (768->768->768). Follow the rest settings of ablation study A.

5.3 Effects of dynamic weight setting

As described in Section 3.5, the guided cross-attention module contains two sub-modules: Cross Attention 1 (CA1) and Cross Attention 2 (CA2). First, we explore the information fusion method by conducting a controlled experiment. Under the static weights setting:

$$E^{GCA} = \beta E^{CA1} + (1 - \beta) E^{CA2} \tag{14}$$

After grid search, the best β is set to 0.37.

It can be seen from Table 4 that the guided cross-attention module with dynamic weights is significantly better than it with static weights. Therefore, we conjecture that the model needs to adaptively change the information fusion weights of the two attention modules according to the input sentence to satisfy tasks of different difficulty.

5.4 Discussion on the number of parameters

Compared with the baseline method, our method introduces additional parameters, mainly from the newly added cross-attention module GCA. So, does the improvement in model performance benefit from the increase in the number of parameters? In order to explore this question, we conducted a related comparative experiment.

We set up two ablation studies in Table 5. As shown in this table, comparing the results of GCA-ablation studies 1 & 2 shows that the increase in the number of parameters does improve the model effect under the current conditions, but the improvement here is negligible compared to the improvement brought by the GCA module. Experimental results show that our proposed method is necessary to align at the error type level.

6 Conclusion

Grammar error correction is significant for many downstream natural language understanding tasks. In this paper, we propose an end-to-end framework termed LET, which effectively leverages the error type information generated by the GED task to guide the GEC task.

Our work solves two critical problems in the previous work. Firstly, we have alleviated the problem of error propagation caused by hard-coded error types by introducing soft-encoded error types. Secondly, we have introduced the concept of error type alignment, which is more reasonable and adequate. We transfer the original semantic vectors into classification vectors to ensure that the two parts of the input of the proposed cross-attention module are both in the same semantic space. Experiments and ablation studies show that alignment leads to better results.

Overall, LET provides a better sample for research in the GEC field and addresses some potential issues with previous technical solutions.

Limitations

By analyzing the error cases, we find that almost all the existing work (including our LET) cannot handle the disorder problem of words well, primarily when the error occurs far from the correct location. For example, there is a correct sentence: *'On my way to school today, I bought a very tasty apple.'*. If the erroneous form is as follows: *'on my way to school apple today, I bought a very tasty.'*, it is hard for the model to understand that the right thing to do is to put *apple* back at the end of the sentence.

Acknowledgements

This work was supported by the National Key R&D Program of China (2022YFB4701400/4701402), SZSTC Grant (JCYJ20190809172201639, WDZC20200820200655001), Shenzhen Key Laboratory (ZDSYS20210623092001004) and Beijing Key Lab of Networked Multimedia.

References

Chris Brockett, Bill Dolan, and Michael Gamon. 2006. Correcting esl errors using phrasal smt techniques. In *21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, Sydney, Australia*.

Christopher Bryant. 2019. Automatic annotation of error types for grammatical error correction. Technical Report UCAM-CL-TR-938, University of Cambridge, Computer Laboratory.

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

Ru-Yng Chang, Chung-Hsien Wu, and Philips Kokoh Prasetyo. 2012. Error diagnosis of chinese sentences using inductive learning algorithm and decomposition-based testing mechanism. *ACM Transactions on Asian Language Information Processing (TALIP)*, 11(1):1–24.

Mengyun Chen, Tao Ge, Xingxing Zhang, Furu Wei, and Ming Zhou. 2020. Improving the efficiency of grammatical error correction with erroneous span detection and correction. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7162–7169, Online. Association for Computational Linguistics.

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

Daniel Dahlmeier, Hwee Tou Ng, and Siew Mei Wu. 2013. Building a large annotated corpus of learner English: The NUS corpus of learner English. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 22–31, Atlanta, Georgia. Association for Computational Linguistics.

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

Mariano Felice, Zheng Yuan, Øistein E. Andersen, Helen Yannakoudakis, and Ekaterina Kochmar. 2014. Grammatical error correction using hybrid systems and type filtering. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, pages 15–24, Baltimore, Maryland. Association for Computational Linguistics.

Ruiji Fu, Zhengqi Pei, Jiefu Gong, Wei Song, Dechuan Teng, Wanxiang Che, Shijin Wang, Guoping Hu, and Ting Liu. 2018. Chinese grammatical error diagnosis using statistical and prior knowledge driven features with probabilistic ensemble enhancement. In *Proceedings of the 5th Workshop on Natural Language Processing Techniques for Educational Applications*, pages 52–59, Melbourne, Australia. Association for Computational Linguistics.

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

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

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

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

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

Shun Kiyono, Jun Suzuki, Masato Mita, Tomoya Mizumoto, and Kentaro Inui. 2019. An empirical study of incorporating pseudo data into grammatical error correction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*

and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 1236–1242, Hong Kong, China. Association for Computational Linguistics.

Kuang-Huei Lee, Xi Chen, Gang Hua, Houdong Hu, and Xiaodong He. 2018. Stacked cross attention for image-text matching. In *Computer Vision – ECCV 2018*, pages 212–228, Cham. Springer International Publishing.

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

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

Xiujun Li, Xi Yin, Chunyuan Li, Xiaowei Hu, Pengchuan Zhang, Lei Zhang, Lijuan Wang, Houdong Hu, Li Dong, Furu Wei, Yejin Choi, and Jianfeng Gao. 2020. Oscar: Object-semantics aligned pre-training for vision-language tasks. In *European Conference on Computer Vision*.

Jared Lichtarge, Chris Alberti, and Shankar Kumar. 2020. Data weighted training strategies for grammatical error correction. *Transactions of the Association for Computational Linguistics*, 8:634–646.

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

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, Baltimore, Maryland. Association for Computational Linguistics.

Diane Nicholls. 2003. The cambridge learner corpus: Error coding and analysis for lexicography and elt. In *Proceedings of the Corpus Linguistics 2003 conference*, volume 16, pages 572–581. Cambridge University Press Cambridge.

Kostiantyn Omelianchuk, Vitaliy Atrasevych, Artem Chernodub, and Oleksandr Skurzhanskyi. 2020. Gector–grammatical error correction: tag, not rewrite. *arXiv preprint arXiv:2005.12592*.

Vipul Raheja and Dimitris Alikaniotis. 2020. Adversarial Grammatical Error Correction. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3075–3087, Online. Association for Computational Linguistics.

Marek Rei and Helen Yannakoudakis. 2016. Compositional sequence labeling models for error detection in learner writing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1181–1191, Berlin, Germany. Association for Computational Linguistics.

Sascha Rothe, Jonathan Mallinson, Eric Malmi, Sebastian Krause, and Aliaksei Severyn. 2021. A simple recipe for multilingual grammatical error correction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 702–707, Online. Association for Computational Linguistics.

Felix Stahlberg and Shankar Kumar. 2021. Synthetic data generation for grammatical error correction with tagged corruption models. In *Proceedings of the 16th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 37–47, Online. Association for Computational Linguistics.

Joel R. Tetreault and Martin Chodorow. 2008. The ups and downs of preposition error detection in ESL writing. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 865–872, Manchester, UK. Coling 2008 Organizing Committee.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

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

Bo Wang, Kaoru Hirota, Chang Liu, Yaping Dai, and Zhiyang Jia. 2020. An approach to nmt re-ranking using sequence-labeling for grammatical error correction. *Journal of Advanced Computational Intelligence and Intelligent Informatics*, 24(4):557–567.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi 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 Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. mT5: A massively multilingual pre-trained text-to-text transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–498, Online. Association for Computational Linguistics.

Helen Yannakoudakis, Ted Briscoe, and Ben Medlock. 2011. A new dataset and method for automatically grading ESOL texts. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 180–189, Portland, Oregon, USA. Association for Computational Linguistics.

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

Zheng Yuan and Christopher Bryant. 2021. Document-level grammatical error correction. In *Proceedings of the 16th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 75–84, Online. Association for Computational Linguistics.

Zheng Yuan and Mariano Felice. 2013a. Constrained grammatical error correction using statistical machine translation. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*, pages 52–61, Sofia, Bulgaria. Association for Computational Linguistics.

Zheng Yuan and Mariano Felice. 2013b. Constrained grammatical error correction using statistical machine translation. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*, pages 52–61.

Zheng Yuan, Felix Stahlberg, Marek Rei, Bill Byrne, and Helen Yannakoudakis. 2019. Neural and FST-based approaches to grammatical error correction. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 228–239, Florence, Italy. Association for Computational Linguistics.

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 the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8722–8736.

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

Liu Zhuang, Lin Wayne, Shi Ya, and Zhao Jun. 2021. A robustly optimized BERT pre-training approach with post-training. In *Proceedings of the 20th Chinese National Conference on Computational Linguistics*, pages 1218–1227, Huhhot, China. Chinese Information Processing Society of China.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
Limitations

☐ A2. Did you discuss any potential risks of your work?
Not applicable. Left blank.

☑ A3. Do the abstract and introduction summarize the paper's main claims?
1 introduction

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☒ Did you use or create scientific artifacts?

Left blank.

☐ B1. Did you cite the creators of artifacts you used?
No response.

☐ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
No response.

☐ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
No response.

☐ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
No response.

☐ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
No response.

☐ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
No response.

C ☑ Did you run computational experiments?

4 Experiments

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
4 Experiments

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
4.3 Experiment Setup

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
4.4 Results analysis

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
4 Experiments

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
No response.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
No response.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
No response.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
No response.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
No response.