System Report for CCL23-Eval Task 7: Chinese Grammatical Error Diagnosis Based on Model Fusion

Yanmei Ma, Laiqi Wang, Zhenghua Chen, Yanran Zhou, Ya Han and Jie Zhang
Beijing Funcun-wuyou Technology Co., Ltd.
{mayanmei, wanglaiqi, chenzhenghua, zhouyanran, hanya, zhangjie}@ifuncun.cn

Abstract

The purpose of the Chinese Grammatical Error Diagnosis task is to identify the positions and types of grammar errors in Chinese texts. In Track 2 of CCL2023-CLTC, Chinese grammar errors are classified into four categories: Redundant Words, Missing Words, Word Selection, and Word Ordering Errors. We conducted data filtering, model research, and model fine-tuning in sequence. Then, we performed weighted fusion of models based on perplexity calculations and introduced various post-processing strategies. As a result, the performance of the model on the test set, measured by COM, reached 49.12.

1 Introduction

The purpose of the Chinese grammatical error diagnosis (CGED) task is to detect the location and type of each grammatical error in the Chinese text. The types of grammatical errors are divided into four categories: Redundant Words (R), Missing Words (M), Word Selection (S), and Word Ordering Errors (W). In recent years, the task of Chinese grammar error correction has attracted more and more attention, and some applications with potential commercial value have also appeared. This technology has a broad application space in education, news, official documents and other fields. The mainstream methods to solve this task are Seq2Seq and Seq2Edits. The Seq2Seq method regards the grammatical error correction task as the process of translating an erroneous sentence into a correct sentence, and uses an advanced neural translation model to solve it; the Seq2Edits method is to design editing actions (such as insertion, deletion, replacement, etc.), the grammar diagnosis task is regarded as a sequence labeling task to solve. In the CCL2023-CLTC track 2: Chinese grammar error detection task, we use the multi-model fusion method and post-processing strategy to realize the text grammar error correction function. Finally, on the CCL2023-CLTC track 2 Chinese grammar error diagnosis task, the result of COM is 49.12.

2 Model

We did a lot of research on models and papers when we were doing the task of Chinese grammar error detection in Track 2. The mainstream methods to solve this task are Seq2Seq and Seq2Edits. The benchmark models we choose are the current mainstream BART (Bidirectional and Auto-Regressive Transformers) (Lewis et al., 2020), GECToR (Grammatical Error Correction: Tag, Not Rewrite) and T5 (Text-to-Text Transfer Transformer) that have achieved SOTA performance on the CGEC (Chinese Grammatical Error Correction) dataset. The following is a detailed introduction to the models we use in this task.

2.1 BART

The BART model (Lewis et al., 2020) uses the Transformer structure (Vaswani et al., 2017). The overall architecture consists of two parts: an encoder and a decoder. The encoder is responsible for converting the input sequence into a high-dimensional representation, and the decoder generates an output sequence based on the representation.

Proceedings of the 22nd China National Conference on Computational Linguistics, pages 250-261, Harbin, China, August 3 - 5, 2023.
(c) Technical Committee on Computational Linguistics, Chinese Information Processing Society of China

The encoder of the BART model is stacked by multi-layer encoders. Each encoder consists of a multi-head self-attention mechanism and a feed-forward neural network. This structure enables the encoder to model different positions of the input sequence and capture the dependencies between global and local. Although the decoder of the BART model also uses the Transformer structure, it is different from the traditional Transformer decoder in that it uses an autoregressive generation method. In the decoding stage, the BART model gradually generates output sequences through autoregressive methods, and the generation of each step depends on the previously generated parts.

The basic architecture of the BART model based on the Transformer neural network is shown in Figure 1.

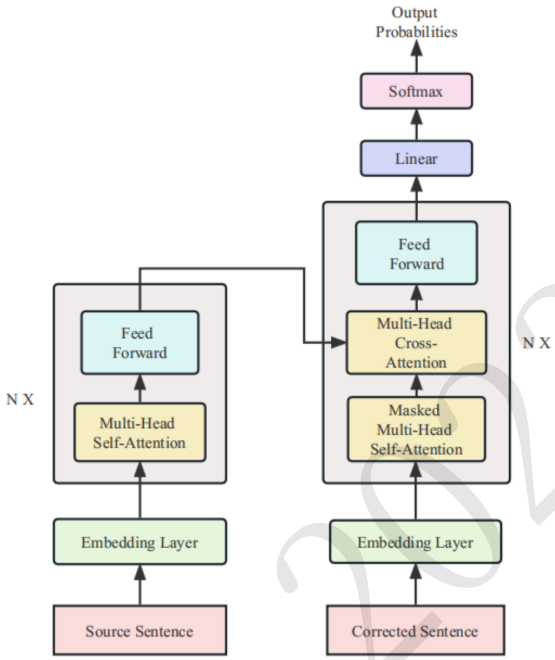

Figure 1: Basic network structure of BART model

2.1.1 Pre-training

Pre-training of the BART model: First, the original text is destroyed by using a variety of noises, and then the original text is reconstructed by the seq2seq model. Therefore, the loss function is the cross entropy of the output of the decoder and the original text. The BART model introduces a total of 5 noise methods that destroy the original text, as shown in the figure 2.

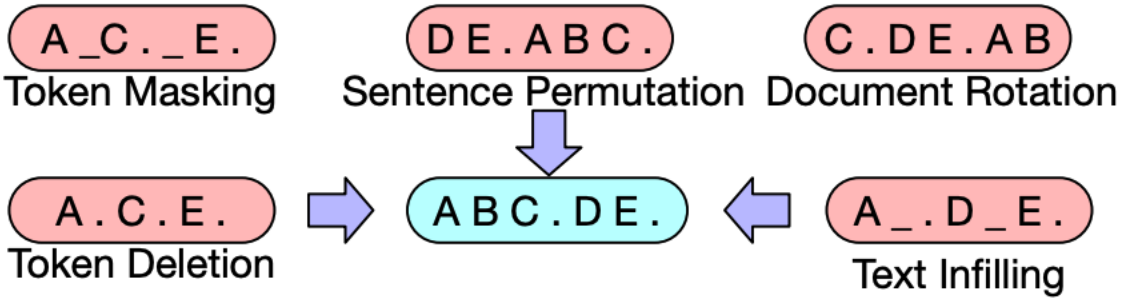

Figure 2: Pre-training strategy for BART model

Proceedings of the 22nd China National Conference on Computational Linguistics, pages 250-261, Harbin, China, August 3 - 5, 2023.

Token Masking: Token mask, which is consistent with the BERT model strategy, randomly extracts tokens and replaces them with [MASK] marks.

Token Deletion: Token deletion, which randomly deletes tokens from the input. Unlike masks, this strategy is for the model to learn which positions lack input information.

Text Infilling: Text filling, randomly select a text segment (the length of the text segment conforms to the Poisson distribution of $\lambda = 3$), and replace it with a [MASK] tag. When the fragment length is 0, it is equivalent to inserting a [MASK] mark at the original position. Different from the SpanBERT model, the SpanBERT model is replaced by the [MASK] mark of the number of fragment lengths.

Sentence Permutation: Sentence sorting, splitting the text according to periods, generating a sequence of sentences, and then randomly shuffling the order between sentences.

Document Rotation: Document rotation, randomly select a token, and then rotate the text, that is, select the token as the beginning of the text. This strategy lets the model learn the beginning of the text.

2.1.2 Fine-tuning

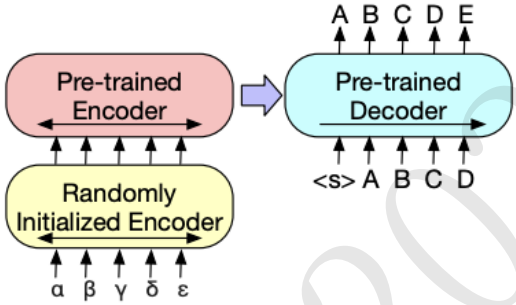

Figure 3: Fine-tuning of BART on translation tasks

Figure 3shows the fine-tuning process of BART in the translation task. Machine Translation: Since the pre-training process is trained in the same language, but machine translation is translated from one language to another, the BART model randomly initializes the Embedding layer of the encoder when performing machine translation tasks, that is, replaces the dictionary. Retrain representations for another language.

In the fine-tuning process, first freeze most of the parameters of the original BART model, and only train the randomly initialized Embedding, the BART model position embedding and the self-attention parameters of the first layer of the BART model encoder connected to the Embedding; then all parameters of the model Do a small amount of training.

2.2 GECToR

GECToR (Grammar Error Correction with Transformer) (Omelianchuk et al., 2020) is also a Transformer-based neural network model, which is specially used for text error correction tasks. Its goal is to automatically detect and correct grammatical and spelling errors in text. The GECToR model as a whole is similar to a sequence labeling task. For the Chinese error correction training task, the input of the model needs to compare two sentences, and use the edit distance operation to represent the label of each character in the original sentence. The total score of the label is There are four forms (KEEP, APPEND, DLETE, REPLAES). The training objective of the model is to minimize the difference between the generated sequence and the reference sequence, using the cross-entropy loss function or other similar objective function as the model loss function. During training, the model learns how to automatically detect and correct grammatical and spelling errors in text.

Proceedings of the 22nd China National Conference on Computational Linguistics, pages 250-261, Harbin, China, August 3 - 5, 2023.

2.3 T5

T5 (Text-to-Text Transfer Transformer) (Raffel et al., 2020) is a powerful language generation model, which is a model architecture or a paradigm for solving NLP tasks. The author borrows the idea of Seq2Seq to unify the tasks of different stages of the model (Pretrain, Fine-tune, Predict) into the task of Text-to-Text (that is, the model input is text, and the output is also text).

T5 retains most of the architecture of the original Transformer, but emphasizes some key aspects. Additionally, some minor changes have been made to vocabulary and functionality. Some main concepts of the T5 mode are listed below:

The encoder and decoder remain in the model. The encoder and decoder layers become blocks, and the sublayers become subcomponents that contain self-attention layers and feed-forward neural networks. The self-attention mechanism is order-independent. Use dot product of matrices instead of recursion. Positional encodings are added to word embeddings before doing the dot product, which explores the relationship between each word and other words in a sequence.

Transformer uses sine and cosine to generate positional encoding, while T5 uses relative positional encoding. In T5, positional encoding relies on the extension of self-attention to compare pairwise relations. The positional encoding of T5 is shared and re-evaluated in all layers of the model simultaneously.

3 Data

Track 2 provides the processed Lang8 dataset and CGED dataset (Rao et al., 2020). The word count statistics of the lang8 dataset and the official test set are shown in Table 1 and Table 2. The statistical results show that: basically all are within 80 characters, of which 97.6% are within 80 characters in the test set, and a few exceed 80 characters. The data sources of CGED are the HSK dynamic composition corpus and the global Chinese interlanguage corpus. CGED-8 includes about 1,400 paragraph units and 3,000 errors. Each unit contains 1-5 sentences, and each sentence is marked with the position, type and modification result of the grammatical error. 5,000 entries were randomly selected from the Lang8 and CGED datasets as the in-group test set.

word count	0-30	30-50	50-80	80-100	100-150	150-200	200 or more
quantity	104.5w	14.5w	2.1w	1395	286	3	1

Table 1: Word count analysis of lang8 dataset

word count	0-30	30-50	50-80	80-100	100-150	150-200	200 or more
quantity	2465	890	322	53	31	5	1

Table 2: Official test set word count analysis

4 Experiment and Results

We use the above data sets to conduct fine-tuning training and comparative experiments on several models, and adopt model fusion and various post-processing strategies to achieve the final submitted result COM: 49.12. Below we will introduce in detail several of the important experiments that have obvious improvement effects.

4.1 Experiment 1: BART

4.1.1 Model training

During the experiment, we used the fairseq tool library to load the BART pre-training model, trained the model with the Lang8 and CGED datasets announced by the competition organizer, and optimized the model parameters through backpropagation and gradient descent algorithms. In each training step, the source language sequence is input into the encoder, and then the decoder is used to generate the

Proceedings of the 22nd China National Conference on Computational Linguistics, pages 250-261, Harbin, China, August 3 - 5, 2023.

target language sequence, and the loss function is calculated, and the decoder parameters are updated according to the gradient of the loss function to gradually improve the performance of the model. The hyperparameter settings during model training are shown in Table 3.

Configurations	**Values**
Pretrained Language model	Chinese-BART-Large
Learning rate	3×10^{-5}
Max epochs	100
Learning rate scheduler	Polynomial
Batch size per GPU	4096 tokens
Loss function	Label smoothed cross entropy (label-smoothing=0.1)
Optimizer	Adam($\beta_1 = 0.9, \beta_2 = 0.99, \epsilon = 1 \times 10^{-8}$)
Dropout	0.3
Max tokens	4096
Patience	5

Table 3: BART model hyperparameter settings

4.1.2 Experimental results

After the model is trained, use the trained BART model to perform reasoning on grammar error correction tasks. Feed the test data as the source language into the encoder, and use the decoder to generate the target language sequences.

In the process of using the model for inference, I tried to average the weights of the model, multiple rounds of error correction, correct UNK characters, optimize decoding parameters, etc. The specific optimization process of the experimental results is introduced as follows:

Model weight averaging: This strategy refers to averaging the parameters of the model saved at different time points during the training process to obtain a model with smoother and better generalization performance. In the experiment, we selected 5 models with better effects for parameter averaging operation. This strategy will increase the comprehensive score on the test set by 0.15 compared with the baseline.

Multiple rounds of error correction: This strategy is to iterate the reasoning process of the model for multiple rounds, and obtain the best number of multiple rounds of iterations by comparing the experimental results. By comparing the experimental results, it is found that when the number of iterations N=2, the comprehensive score on the test set is the highest. Using this strategy further improves the composite score by 1.11 on the test set. The flow chart of using the model for multiple iterations of inference is shown in Figure 4.

Correct UNK characters: By analyzing the results, it is found that the model decodes some English characters such as BAHAYKUBO, SOGO, MOS into UNK characters during the inference process, then compares the result with the original sentence and replaces the UNK characters in the result by using the content in the original sentence. This strategy will further improve the composite score on the test set by 0.13.

Optimize decoding parameters: In the process of model inference, we use Beam Search, a decoding algorithm that explores potential high-probability sequences by retaining a certain number of candidates at each time step. Among the parameters of the Beam Search algorithm, the beam size is an important parameter, which controls the number of candidates retained at each time step. It is verified by experiments: when the beam size is 12, the comprehensive score on the test set is the best .

By using the above various strategies, the comprehensive score COM obtained by a single model on the validation set is 47.89. The specific experimental results of each item are shown in Table 4.

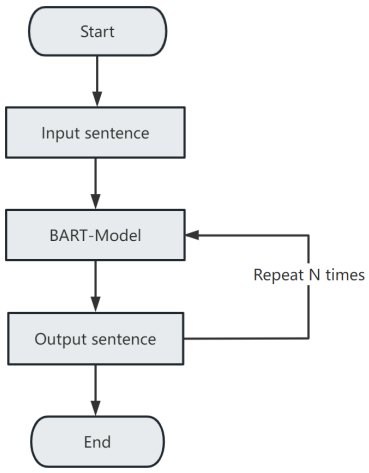

Figure 4: BART model reasoning process

COM	FPR	DET	IDE	POS	COR
47.89	21.79	83.18	59.64	40.93	29.81

Table 4: Experimental results of BART single model on the official test set

4.2 Experiment 2: BART-based seq2seq model

4.2.1 Model training

1. Use lang8 to train the seq2seq model, train 10 epochs, and store the model parameters of each epoch separately.

2. Set max_length to 100 and 256 respectively, and then train with lang8 data. The model parameters are shown in Table 5.

Configurations	Values
Pretrained Language model	Chinese-BART-Large
Learning rate	3×10^{-6}
Max epochs	10
Learning rate scheduler	Polynomial
Batch size per GPU	32

Table 5: seq2seq model hyperparameter settings

4.2.2 Experimental results

1. Use the model inference test set saved in each epoch to compare the results.

2. The models trained with different max_lengths reasoned about the test set separately and compared the results. seq2seq model's experimental results on the in-group test set are shown in Table 6, the value of epoch is inversely proportional to the model's effect under the condition that max_length is the same and epoch is not zero. Under the condition of the same epoch, the model effect of max_length=256 is better than that of max_length=100. So the model with epoch=1 and max_length=256 was selected for the synthesis.

Proceedings of the 22nd China National Conference on Computational Linguistics, pages 250-261, Harbin, China, August 3 - 5, 2023.

model	COM	COR	DET	FPR	IDE	POS
epoch=0,max_length=100	29.46	15.88	70.12	33.83	41.60	24.07
epoch=1,max_length=100	37.76	22.01	83.13	39.09	54.19	30.84
epoch=2,max_length=100	36.82	26.10	72.82	38.20	51.30	35.24
epoch=10,max_length=100	30.24	24.70	84.57	75.81	55.37	32.12
epoch=0,max_length=256	29.55	16.08	70.28	33.78	41.45	24.17
epoch=1,max_length=256	38.55	20.86	78.54	26.25	51.41	29.64
epoch=2,max_length=256	36.91	19.13	71.11	16.52	47.12	26.78
epoch=10,max_length=256	31.08	18.13	79.32	51.77	50.02	28.61

Table 6: Experimental results of the seq2seq model on the test set within the group

4.3 Experiment 3: GECToR

4.3.1 Model training

In this experiment, the GECToR model was used, and two pre-trained models, chinese-bert-wwm-ext (Cui et al., 2020) and structbert-large-zh (Wang et al., 2019), were used for comparative experiments. During the initial experiments, it was found that the model was more inclined to predict the deletion label—$DELETE label. To solve this problem, split the $DELETE tag in the original task to delete the corresponding Chinese character -$DELETE_char. The training set uses the preprocessed Lang8 data set provided by Track 2 and all the simplified Chinese data sets of CGED, and the test set of CGED2021 is used as the test set for training. In the prediction process of this experiment, multiple predictions are used to obtain the final prediction result, that is, the first prediction result of the model is used as the input of the second prediction of the model, and the process is repeated three times to obtain the final result. The hyperparameter settings of the GECToR model are shown in Table 7.

Configurations	Values
Learning rate	1×10^{-5}
Max epochs	10
Max length	128
Batch size per GPU	64

Table 7: GECToR model hyperparameter settings

4.3.2 Experimental results

The experimental results of the GECToR model on the test set within the group are shown in Table 8.

model	COM	COR	DET	FPR	IDE	POS
base_chinese-bert-wwm-ext	37.70	15.95	67.80	17.40	41.23	23.21
$DELETE_char_chinese-bert-wwmext	37.08	18.98	77.33	22.86	49.38	25.47
$DELETE_char_structbert-large-zh	39.03	26.00	82.66	43.51	55.73	35.24

Table 8: Experimental results of the GECToR model on the test set within the group

4.4 Experiment 4: T5

4.4.1 Model training

The T5 model training uses the T5Corrector base v2 model as the pre-training model, and fine-tunes the model based on it. The training set uses the Lang8 and CGED data sets, and the CGED2021 test set is selected as the verification set, and the model of each epoch is saved. Select the 0th, 10th, 20th, 50th, and 60th epoch models for the official test set test, and analyze the relationship between the model effect and the training time.

Proceedings of the 22nd China National Conference on Computational Linguistics, pages 250-261, Harbin, China, August 3 - 5, 2023.

T5 model hyperparameter settings are shown in Table 9.

Configurations	Values
Pretrained Language model	T5Corrector_base_v2
Learning rate	5×10^{-4}
weight_decay	0.01
Max epochs	60
Learning rate scheduler	Polynomial
Batch size per GPU	64
Optimizer	Adam($\beta_1 = 0.9, \beta_2 = 0.99, \epsilon = 1 \times 10^{-8}$)

Table 9: T5 model hyperparameter settings

4.4.2 Experimental results

model	COM	COR	DET	FPR	IDE	POS
T5	15.05	17.70	41.31	20.66	8.88	7.06
T5(epoch=10)	27.96	20.35	68.10	41.69	13.21	9.18
T5(epoch=20)	35.12	28.32	76.45	49.80	26.43	16.12
T5(epoch=50)	44.25	28.32	82.65	57.08	37.46	28.15
T5(epoch=60)	42.26	30.38	82.92	57.03	33.76	25.74

Table 10: Experimental results of the T5 model on the test set within the group

The experimental results of the T5 model on the test set within the group are shown in Table 10. The results show that when the T5 model is trained and fine-tuned using the lang8 and CGED datasets, the effect is the best at 50 epochs, and the model indicators will decline after more than 50 epochs.

4.5 Experiment 5: Model Fusion

4.5.1 Experimental analysis

A single model may be weak in correcting certain types of grammatical errors. By using multiple models, especially specialized models for different types of errors, the ability to cover a wide range of grammatical errors can be improved. Different models may have different focuses and expertise, so fusion can integrate their strengths to provide a more comprehensive correction capability. Individual models may have problems with missing or false positives when correcting certain types of syntax errors. With model fusion, the output of multiple models can be combined, thereby reducing the number of missed and false positives of errors. For example, GECToR is a non-autoregressive model, so it does not correct errors for multiple consecutive characters correctly, e.g., GECToR will change "This opinion reflects the theory of the average Briton." to "This opinion reflects the theory of the average Briton." This kind of error correction has a high score in ppl and can be filtered by adding ppl to the fusion strategy.

Through the above analysis for single-model results, we find that by model fusion, the advantages of multiple models can be combined and the shortcomings of a single model can be compensated to improve the performance of Chinese grammar error correction tasks.

4.5.2 Fusion strategy

Based on traditional voting strategy

1. In this evaluation, we integrated the corrected results of multiple models based on the inference results of different models and the perplexity calculated on sentences modified by the models. Proceed as follows: Set the weights of model 1, ..., model n

$$model\_weights = [mw_1, \ldots, mw_n] \quad (1)$$

Proceedings of the 22nd China National Conference on Computational Linguistics, pages 250-261, Harbin, China, August 3 - 5, 2023.

We set the threshold value for error location detection is threshold_detect, and the threshold value for error correction is threshold_correct. Each model has different thresholds, and the exact values can be found in the code.

2. The score for an error detection position in a sentence is:

$$score_{detect} = \sum_{i=1}^{n} mw_i \times detect_i \tag{2}$$

Where $detect_i$ is the error detection result of the i-th model,

$$detect_i = \begin{cases} 1, & \text{if the i-th model detected the error} \\ 0, & \text{if the i-th model didn't detected the error} \end{cases} \tag{3}$$

The score for correcting an error-checked position to a token in a sentence is:

$$score_{token} = \sum_{i=1}^{n} mw_i \times token_i \tag{4}$$

$token_i$represents whether model i corrects this error-checking position to token:

$$token_i = \begin{cases} 1, & \text{if the i-th model corrected this position to this token} \\ 0, & \text{if the i-th model didn't correct this position to this token} \end{cases} \tag{5}$$

ppl-based voting strategy

1. Add confusion strategy, calculate the perplexity for the original sentence and corrected sentences of n models respectively, and analyze the difference of perplexity:

$$diff = \frac{ppl_i - ppl_{src}}{ppl_{src}} \tag{6}$$

ppl_i is perplexity of the i-th model prediction sentence, ppl_{src} is perplexity of the original sentence.

We set perplexity weighting value is in out experiment.

$$weight_{ppl} = \begin{cases} 0.8, & \text{if } diff < 0 \\ 1, & \text{if } 0 \leq diff < 0.2 \\ 1.3, & \text{if } 0.2 \leq diff < 0.4 \\ 1.5, & \text{if } 0.4 \leq diff \leq 1 \end{cases} \tag{7}$$

The formula for each model's score on whether the error-checking position in a sentence requires error correction becomes:

$$score_{detect} = max(weight_{ppl\_1}, ..., weight_{ppl\_n}) \sum_{i=1}^{n} mw_i \times detect_i \tag{8}$$

$weight_{ppl\_i}$ is the perplexity weighted value of i-th model to make error detection judgments for that location.

The score for correcting an error-checking position to this token in a sentence is:

$$score_{token} = max(weight_{ppl\_token\_1}, ..., weight_{ppl\_token\_n}) \sum_{i=1}^{n} mw_i \times detect_i \tag{9}$$

$weight_{ppl\_token\_i}$ indicates the perplexity weighting value when the i-th model modifies this error detection location to this token, and if the error correction result of the i-th model is not a token, then $weight_{ppl\_token\_i}$ is 0.

Proceedings of the 22nd China National Conference on Computational Linguistics, pages 250-261, Harbin, China, August 3 - 5, 2023.

2. Screening Strategy:

Calculate $score_{detect}$ for the error correction results of n models at the position, and $score_{token\_i}$ for each of the multiple error corrections $token_i$ given by n models at the position. Only when:

$$score_{detect} \geq threshold\_detect \tag{10}$$

and:

$$max(score_{token\_1}, ..., score_{token\_i}, ..., score_{token\_n}) \geq threshold\_correct \tag{11}$$

are satisfied, the maximum token is adopted as the correction result for the error-checking position. When either condition is not satisfied, the error is not corrected.

4.5.3 Experimental results of two fusion strategies

We conducted experiments with the above two fusion strategies separately, and the results are shown in Table 11. By comparison, the ppl fusion strategy performs better than the traditional voting strategy.

Fusion strategy	COM	COR	DET	FPR	IDE	POS
Traditional Voting	48.23	29.81	83.86	20.20	59.63	40.24
PPL Voting	48.57	30.19	84.34	19.62	59.15	40.23

Table 11: Results of two fusion strategies

4.5.4 Post-processing strategy

(1) First correct the spelling of the sentence based on pycorrector (Xu, 2021), and then perform grammatical error correction.

(2) Segmentation of long sentences: Words of more than 80 characters are first segmented according to punctuation marks, and then sequentially spliced according to the rules of no more than 80 characters.

(3) For non-Simplified Chinese conversion strategy: Traditional Chinese is uniformly corrected to Simplified Chinese, Japanese Kanji is uniformly corrected to the corresponding Chinese Kanji, and English is not corrected.

4.5.5 Results

Different strategies on the official test set can yield results as shown in the Table 12.

Model	COM	FPR	DET	IDE	POS	COR
model_E[1]	48.57	19.62	84.34	59.15	40.23	30.19
model_E+CSC[2]	48.73	19.62	84.37	59.37	40.41	30.42
model_E+CSC+cut[3]	48.90	20.35	84.24	60.03	41.09	30.20
model_E+CSC+cut+T2S[4]	49.12	19.47	84.26	60.06	41.16	30.46

[1] The result of the model fusion, as the baseline of the post-processing strategy, is called model_E (model_ensemble)
[2] CSCSpelling Correction
[3] cutLong Sentence Syncopation
[4] T2SNon-simplified conversion strategies

Table 12: Results of different strategies on the official test set

5 Summarize

In this competition, we adopted the method of multi-model fusion, combined with various post-processing strategies, which can effectively improve the performance of the model, and finally obtained the result of COM being 49.12 on the official test set.

Proceedings of the 22nd China National Conference on Computational Linguistics, pages 250-261, Harbin, China, August 3 - 5, 2023.

5.1 Innovation

For this competition, we have the following innovations:

(1) In the process of model reasoning, methods such as averaging model weights and multiple rounds of error correction have been tried.

(2) The fusion technology of multi-model error detection results, including the ppl strategy, maintains the characteristics of each model, complements each other's advantages, first screens the position, and then screens the results.

(3) In the experiment, we tried a variety of post-processing strategies, and compared and selected several strategies that can improve the results.

5.2 Disadvantages

For this competition, we have the following regrets and deficiencies: the models investigated and used in this competition are limited, and they are all character-sized, which cannot cover all the current error correction models. In future work, we can study and use Chinese-specific words information and rich semantic information, further improving the performance of the Chinese grammar error correction model.

Acknowledgements

After two months of data analysis, model research, and comparison experiments, we finally finished this competition.

First of all, we would like to thank our leader, Xueqian Liu, the CTO of Funcun Intelligence, for giving us the opportunity to participate in this competition and for his technical guidance during the competition. Thank him for his support and encouragement. From the selection of the topic to its finalization, he has always given us patient guidance and unremitting support, and he has patiently guided us in model selection, model fusion and other strategies from his professional point of view.

Secondly, we would like to thank the organizers and hosts for providing us with this competition opportunity, which is a valuable experience and enhances the cohesion of our team, which will be a valuable asset for us.

Finally, we sincerely thank the teachers of the working committees of the organizer will attend the evaluation of our evaluation report in their busy schedules.

References

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 the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 657–668, Online, November. Association for Computational Linguistics.

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, July. Association for Computational Linguistics.

Kostiantyn Omelianchuk, Vitaliy Atrasevych, Artem Chernodub, and Oleksandr Skurzhanskyi. 2020. GECToR – grammatical error correction: Tag, not rewrite. In *Proceedings of the Fifteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 163–170, Seattle, WA, USA → Online, July. Association for Computational Linguistics.

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. *The Journal of Machine Learning Research*, 21(1):5485–5551.

Gaoqi Rao, Erhong Yang, and Baolin Zhang. 2020. Overview of NLPTEA-2020 shared task for Chinese grammatical error diagnosis. In *Proceedings of the 6th Workshop on Natural Language Processing Techniques for Educational Applications*, pages 25–35, Suzhou, China, December. Association for Computational Linguistics.

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.

Wei Wang, Bin Bi, Ming Yan, Chen Wu, Zuyi Bao, Jiangnan Xia, Liwei Peng, and Luo Si. 2019. Structbert: Incorporating language structures into pre-training for deep language understanding. *arXiv preprint arXiv:1908.04577*.

Ming Xu. 2021. Pycorrector: Text error correction tool. https://github.com/shibing624/pycorrector.