Grammatical Error Correction with Contrastive Learning in Low Error Density Domains

Hannan Cao **Wenmian Yang** **Hwee Tou Ng**
Department of Computer Science, National University of Singapore
caoh@u.nus.edu, {yangwm, nght}@comp.nus.edu.sg

Abstract

Although grammatical error correction (GEC) has achieved good performance on texts written by learners of English as a second language, performance on low error density domains where texts are written by English speakers of varying levels of proficiency can still be improved. In this paper, we propose a contrastive learning approach to encourage the GEC model to assign a higher probability to a correct sentence while reducing the probability of incorrect sentences that the model tends to generate, so as to improve the accuracy of the model. Experimental results show that our approach significantly improves the performance of GEC models in low error density domains, when evaluated on the benchmark CWEB dataset.

1 Introduction

Grammatical error correction (GEC) is the task of correcting errors in a source sentence and generating a well-written and grammatically correct target sentence. Good results have been achieved by state-of-the-art GEC systems based on the seq2seq transformer architecture (Grundkiewicz et al., 2019; Choe et al., 2019; Omelianchuk et al., 2020). However, most prior approaches in GEC are all targeting English-as-a-second-language (ESL) datasets, where GEC systems are trained to correct errors made by ESL learners. In fact, grammatical and other writing errors are made not only by ESL speakers but also by native speakers. Therefore, correcting grammatical errors made by native speakers should also be considered, which helps to broaden the application of GEC.

Compared to the errors made by ESL learners, native English speakers are less likely to make grammatical errors, so the density of errors in the sentences is much lower. The GEC model may end up over-correcting or failing to correct certain errors unique to native speakers.

To address the problem mentioned above, it is necessary to improve the ability of the model to discriminate grammatical features from ungrammatical features with minor differences. Recently, supervised contrastive learning (CL) was proposed by (Chen et al., 2020), which allows the model to learn discriminative features through pushing the features of positive samples closer together and negative samples further apart. However, since GEC is a text generation task, it is not clear how to generate positive sample sentence pairs. To bridge the gap, we instead incorporate CL by increasing the probability of the model generating the right corrections and further reducing the probability of generating the wrong corrections, thereby improving the ability of the model for error correction in low error density domains.

More specifically, we use the negative log-likelihood (NLL) loss to increase the probability of a model to generate positive samples (the right corrections) and use a margin-based CL loss to increase the gap between the probability of positive samples and the probability of negative samples (the wrong corrections) predicted by the GEC model. In this paper, the negative samples are generated in two ways. The first kind of negative samples consists of those wrong corrections generated with high probability by the GEC model during beam search. The second kind of negative samples consists of erroneous sentences from the dataset that require some correction. Through the above negative sampling method, we make the model avoid over-correcting a correct sentence or neglect to correct an erroneous sentence.

The main contributions of this paper are as follows:[1]

- We propose a new loss function based on CL, which allows the model to achieve higher performance in low error density domains. To the

[1]Our source code is available at https://github.com/nusnlp/geccl.

best of our knowledge, our work is the first to incorporate CL in GEC.

- We design a negative sampling method with two strategies, which makes the GEC model avoid over-correcting a correct sentence or neglect to correct an erroneous sentence.
- Experimental results on the benchmark dataset show that our CL approach can significantly improve the performance of seq2seq GEC models compared to direct fine-tuning in low error density domains.

2 Method

In this section, we will first introduce the background of grammatical error correction in Section 2.1, and then introduce our contrastive learning method in Section 2.2.

2.1 Background of Grammatical Error Correction

Let $s^{(i)}$ be an ungrammatical source sentence and $t^{(i)}$ be the corrected grammatical target sentence. For a grammatical error correction (GEC) model parameterized by θ, the goal is to minimize the NLL for a set of M sentence pairs $\{\langle s^{(i)}, t^{(i)}\rangle\}_{i=1}^{M}$, as follows:

$$L_{NLL}^{(i)}(\theta) = -\log P(t^{(i)}|s^{(i)}, \theta) \tag{1}$$

$$L_{NLL}(\theta) = \frac{1}{M}\sum_{i=1}^{M} L_{NLL}^{(i)}(\theta) \tag{2}$$

Given trained parameters $\hat{\theta}$, the hypothesis sentence $\hat{t}^{(i)}$ is generated using beam search to select the candidate with the highest probability, as follows:

$$\hat{t}^{(i)} = \arg\max_{t^{(i)}}\{P(t^{(i)}|s^{(i)}, \hat{\theta})\} \tag{3}$$

2.2 Contrastive Learning for GEC

2.2.1 Overview

As we mentioned above, the GEC model may end up over-correcting or fail to correct certain errors unique to native speakers. To tackle this problem, we propose the use of contrastive learning (CL) to expand the loss gap between positive sample pairs (the right corrections) and negative sample pairs (the wrong corrections) so that the model can better distinguish grammatically correct features from grammatically incorrect features.

Algorithm 1 Contrastive Learning for GEC

```
Input: $D_T$, $D_F$
Output: $\hat{\theta}_{CL}$
1: Obtain optimal $\hat{\theta}$ for the GEC model trained on $D_T$ via Eq. 2.
2: Construct $D_{\tilde{F}}$ automatically based on $D_F$ and $\hat{\theta}$ by the negative sampling method.
3: Obtain the optimal parameters for the GEC model fine-tuned on $D_{\tilde{F}}$ via Eq. 5.
```

Our proposed contrastive learning approach is described in Algorithm 1, which consists of three steps. Specifically, in the algorithm, the input D_T and D_F represent the datasets used for training (i.e., the standard GEC dataset) and fine-tuning (i.e., the low error density GEC dataset), respectively. In the first step, the GEC model is trained on D_T via Eq 2 described in Section 2.1. In the second step, the negative sample dataset $D_{\tilde{F}}$ is constructed using the negative sampling method that will be described in Section 2.2.3. In the third step, the model is fine-tuned via Eq 5 to be described in Section 2.2.2.

2.2.2 Loss for Contrastive Learning

The idea of supervised contrastive learning is to make features of samples from the same class close together, and to make features of samples from different classes far apart (Khosla et al., 2020), thereby improving the feature discrimination of the model. However, since the GEC task is a text generation task instead of a classification task, no samples belong to the same class.

To overcome this problem, we instead improve model feature discrimination by increasing the probability of the model generating positive samples (right corrections) and further reducing the probability of the model generating negative samples (wrong corrections).

Specifically, to discourage the model from generating ungrammatical sentences, we design a margin-based contrastive learning loss as follows:

$$L_{CL}^{(i)}(\hat{\theta}) = \frac{1}{N^i}\sum_{n=1}^{N^i} \max(-\log P(t^{(i)}|s^{(i)}, \hat{\theta}) + \log P(\tilde{t}_n^{(i)}|s^{(i)}, \hat{\theta}) + \gamma, 0) \tag{4}$$

where $\langle s^{(i)}, \tilde{t}_n^{(i)}\rangle$ is a negative sample pair. For the i-th positive sample, it is possible to have N^i negative sample pairs, which will be described in Section 2.2.3. We utilize $L_{CL}^{(i)}(\hat{\theta})$ to ensure that the margin of log-likelihood between a positive sample pair and a negative sample pair is higher than γ.

To further encourage the model to generate grammatically correct sentences, we combine $L_{CL}^{(i)}(\hat{\theta})$ with the NLL loss, and obtain the combined loss as:

$$L(\hat{\theta}) = \frac{1}{M}\sum_{i=1}^{M}\left\{L_{NLL}^{(i)}(\hat{\theta}) + L_{CL}^{(i)}(\hat{\theta})\right\} \quad (5)$$

2.2.3 Negative Sampling Method

We choose the wrong corrections that the model tends to generate and the incorrect sentences that the model neglects as the negative samples. In this way, the model can learn more significant grammatically correct features. More formally, given a ground-truth sentence pair $\langle s^{(i)}, t^{(i)}\rangle$ from D_F, a set of negative sample pairs $\langle s^{(i)}, \tilde{t}^{(i)}\rangle$ will automatically be constructed by the following two strategies:

- For a positive sample pair $\langle s^{(i)}, t^{(i)}\rangle$, we feed $s^{(i)}$ to the model parameterized by $\hat{\theta}$, and choose the top k output sentences with the highest probability generated by beam search. Each sentence among these top k output sentences that is not identical to the target $t^{(i)}$ is selected as a negative sentence $\tilde{t}^{(i)}$ and forms a negative sample pair $\langle s^{(i)}, \tilde{t}^{(i)}\rangle$.
- If $s^{(i)}$ is not identical to $t^{(i)}$ in the positive sample pair $\langle s^{(i)}, t^{(i)}\rangle$ (i.e., some edits are made to $s^{(i)}$ to generate the corrected sentence $t^{(i)}$), we further form a new negative sample pair $\langle s^{(i)}, s^{(i)}\rangle$.

3 Experiments

In this section, we demonstrate the effectiveness of our CL approach.

3.1 Datasets

To evaluate the efficiency of our CL approach in low error density domains, we conduct experiments on the public dataset CWEB (Flachs et al., 2020)[2], which is currently the only public dataset for native English speakers. Following most existing work, we use BEA 2019 (Bryant et al., 2019)[3] as the training set, consisting of NUCLE (Dahlmeier et al., 2013), FCE (Yannakoudakis et al., 2011), Lang-8 (Tajiri et al., 2012), and W&I (Bryant et al., 2019). Detailed statistics of the datasets are shown in Table 1.

Usage	Dataset	#sent	err%	e/s
Train	BEA-train	1,162,256	51.6	2.6
Fine-tune	CWEB-train	4,729	21.9	1.5
Dev	CWEB-dev	2,000	22.3	1.5
Test	CWEB-S-test	2,864	24.5	1.5
	CWEB-G-test	3,981	25.6	1.9

Table 1: Dataset statistics. e/s is the number of edits per sentence calculated on erroneous sentences. err% is the percentage of erroneous sentences in the entire dataset.

CWEB is a low error density dataset consisting of two domains, S and G. Compared to G, S focuses more on professional writings and contains fewer errors. CWEB-dev is the combination of the first 1,000 sentences from the CWEB-S development set (2,862 sentences in total) and the first 1,000 sentences from the CWEB-G development set (3,867 sentences in total), and the remaining 4,729 sentences are regarded as CWEB-train, similar to the setting used in (Flachs et al., 2020). When testing on CWEB-S/G-test, we use BEA-train for training and CWEB-train for fine-tuning.

3.2 GEC Systems

In this paper, we employ our CL approach on two state-of-the-art seq2seq GEC systems, i.e., GEC-PD (Kiyono et al., 2019), and GEC-BART (Katsumata and Komachi, 2020) to verify its effectiveness. The detailed description of these two systems follows.

GEC-PD uses a Transformer-based framework (Vaswani et al., 2017) with the Transformer-big setting. This system is first pre-trained on 70 million parallel synthetic sentences. Then, it is further trained on the BEA-train erroneous portion by only choosing sentence pairs whose source sentence is not identical to the target sentence, consisting of 561,525 sentence pairs. During fine-tuning, we use the training setting in (Kiyono et al., 2019) but change its optimizer to Adam. Note that we do not apply any of the post-processing steps used in GEC-PD, because our goal is to compare our CL approach against direct fine-tuning.

GEC-BART builds the GEC system based on the BART-large (Lewis et al., 2020) pre-trained model. GEC-BART is also further trained on the

[2]https://github.com/SimonHFL/CWEB
[3]https://www.cl.cam.ac.uk/research/nl/bea2019st/

Method	GEC-PD						GEC-BART					
	S			G			S			G		
	P	R	$F_{0.5}$	P	R	$F_{0.5}$	P	R	$F_{0.5}$	P	R	$F_{0.5}$
DI	17.27	15.75	16.91	21.34	23.00	21.58	18.90	9.94	16.00	26.44	17.72	23.97
NLL	35.91	12.96	26.46	**42.09**	16.56	32.01	**40.52**	11.43	26.79	41.38	12.49	28.14
CL^-	35.32	18.69	29.93	36.54	20.45	31.38	39.24	13.87	28.67	41.99	15.76	31.35
CL	**36.30**	**20.40**	**31.34***†	37.21	**23.15**	**33.03***†	40.16	**15.08**	**30.08***†	**43.68**	**16.81**	**32.92***†

Table 2: Results of two GEC systems evaluated on CWEB-test (in %). DI (direct inference): The systems are loaded with their trained weights from BEA-train and then tested on CWEB-test. NLL: The systems are fine-tuned by minimizing the NLL loss in Eq 2. For DI and NLL for the GEC-PD system, we use the reported results from (Flachs et al., 2020). For the GEC-BART system, we obtain the DI and NLL results by following the original setting in (Katsumata and Komachi, 2020). CL^- and CL: The systems are fine-tuned with our CL approach by minimizing the loss in Eq. 5 without and with the second strategy of the negative sampling method. For the detailed training setting, please see Appendix A.4. Statistically significant improvements ($p < 0.001$) of the CL approach over the NLL approach and the CL^- system are marked with an asterisk (*) and a dagger (†), respectively.

BEA-train erroneous portion as GEC-PD does. For fine-tuning, we choose the same setting as (Katsumata and Komachi, 2020).

3.3 Settings and Hyper-parameter Selection

In this paper, we implement the GEC systems based on publicly available code[4], and fine-tune the model using NVIDIA V100 GPU. Unless otherwise stated, we use the same hyper-parameters as the original GEC systems. For evaluation, we use the ERRANT scorer (Bryant et al., 2017) for all datasets and carry out statistical significance tests using one-tailed sign test with bootstrap resampling on 100 samples.

There are two hyper-parameters in our CL approach: the number k of top-ranked candidates during beam search in Section 2.2.3, and the margin parameter γ from the loss function in Eq. 4. We select γ in the range (0.1, 1.0) with a step size of 0.05, and k in the range of $\langle 2, 3, 4\rangle$ using grid search. We get the best results on CWEB-dev when $k = 3$, $\gamma = 0.25$ for GEC-PD, and $k = 3$, $\gamma = 0.85$ for GEC-BART, respectively.

3.4 CWEB Results

The results of our CL approach with fine-tuning on CWEB-train are shown in Table 2. Since each CWEB sentence was annotated by two annotators, following the setting in (Flachs et al., 2020), we first calculate the $F_{0.5}$ score based on each individual annotator, and report the average score as the final result.

In the S domain, GEC-PD with CL achieves the best performance with $F_{0.5}$ score of 31.34% and also achieves the most significant improvement of 4.88% compared with NLL. In the G domain, GEC-PD with CL achieves the best performance too, with $F_{0.5}$ score of 33.03%, while GEC-BART achieves the most significant improvement with 4.78% compared with NLL.

Compared with NLL, our CL approach can significantly increase recall and achieve competitive precision in both of the systems, except for GEC-PD in the G domain. This is likely because GEC-PD is pre-trained with a large amount of synthetic data. Although CL fails to increase the precision for GEC-PD in the G domain, the overall $F_{0.5}$ score still increases and the increase is statistically significant.

3.5 Ablation Study

We also carry out an ablation study to show the importance of the second strategy in the negative sampling method in low error density domains. The performance of CL without and with the second strategy of the negative sampling method is shown in Table 2.

The results show that after adding the second strategy of the negative sampling method, both precision and recall increase for both GEC systems. This shows that adding neglected error sentences as negative samples is an effective way for low error density domains.

3.6 Effect on Over Correction and Ignored Correction

We use the Overdone Edit (OE) ratio and Ignored Edit (IE) ratio to measure over-correction and ignored correction, respectively. Specifically, we use the closed interval [start, end] to indicate the range of an edit. Then, an edit in the gold edits is

[4]GEC-PD: https://github.com/butsugiri/gec-pseudodata; GEC-BART:https://github.com/Katsumata420/generic-pretrained-GEC

counted as IE if its range does not intersect with any model-generated edits. Similarly, an edit in the model-generated edits is counted as OE if its range does not intersect with any gold edits. Those model-generated edits that intersect with the gold edits but not correct are counted as wrong edits, not counted as the above two cases.

The IE ratio is calculated by dividing the number of IEs by the number of gold edits, and the OE ratio is calculated by dividing the number of OEs by the number of model-generated edits. The results of the OE ratio and IE ratio are shown in Table 3.

Category		GEC-PD		GEC-BART	
		NLL	CL	NLL	CL
S	IE ratio	53.99	**38.20**	52.01	**45.23**
	OE ratio	53.54	**52.02**	46.02	**44.35**
G	IE ratio	37.88	**24.33**	41.69	**35.87**
	OE ratio	**43.25**	45.74	40.95	**37.50**

Table 3: IE ratio and OE ratio (in %) of the GEC-PD and GEC-BART systems on the CWEB test set.

We have successfully reduced the IE ratio and the OE ratio for both systems in S and G domain, except for the case of GEC-PD in G domain. This result demonstrates that CL can effectively reduce the over correction problem and ignored correction problem.

4 Related Work

4.1 Grammatical Error Correction

The state-of-the-art approach in GEC uses sequence-to-sequence learning with transformer neural networks (Grundkiewicz et al., 2019; Choe et al., 2019; Omelianchuk et al., 2020). Several task-specific techniques have been proposed for the seq2seq GEC models. (Zhao et al., 2019) incorporated a copy mechanism into transformer networks (Vaswani et al., 2017), since many words in a source sentence are often correct and they should be kept. Diverse ensembles (Chollampatt and Ng, 2018a), rescoring (Chollampatt and Ng, 2018b), and iterative decoding (Omelianchuk et al., 2020; Lichtarge et al., 2019) have also been applied to improve the accuracy of GEC.

4.2 Contrastive Learning

Contrastive learning has been used to learn a good representation by contrasting positive with negative samples. (Chen et al., 2020) demonstrate that contrastive learning could boost the performance of semi-supervised learning and self-supervised learning in computer vision.

In natural language processing, contrastive learning has also been used. In word2vec (Mikolov et al., 2013), a center word and a word in its surrounding context are regarded as a positive sample and their vector representations are pushed together, while a center word and a randomly chosen word are regarded as a negative sample and their vector representations are pushed further apart. Besides word2vec, contrastive learning has also been used in natural language inference (Cui et al., 2020), language modeling (Liza and Grzes, 2018), and knowledge graph embeddings (Bose et al., 2018).

Most of the above methods work at the sample level and have to generate both positive and negative samples. However, since the positive samples are hard to generate in the GEC task, the above methods are not suitable for GEC. Compared to the above methods, our approach does not need to generate extra positive samples. Although (Yang et al., 2019) propose a sentence-level margin loss-based method for machine translation to reduce the word omission errors and do not need positive samples too, their negative samples are generated by word omission at the token level and cannot be used in GEC. In contrast, our approach uses beam search to generate erroneous sentences as negative samples at the sentence level, which effectively prevents the model from making mistakes and thus is more suitable for the GEC task.

5 Conclusion

In this paper, we propose a contrastive learning approach and a corresponding negative sampling method to improve the performance of seq2seq GEC models in low error density domains. By assigning a higher probability to grammatical corrections and reducing the probability of wrong corrections that the model tends to generate, we improve the performance of GEC models in low error density domains.

Acknowledgements

This research is supported by the National Research Foundation, Singapore under its AI Singapore Programme (AISG Award No: AISG-RP-2019-014). The computational work for this article was partially performed on resources of the National Supercomputing Centre, Singapore (https://www.nscc.sg).

References

Avishek Joey Bose, Huan Ling, and Yanshuai Cao. 2018. Adversarial contrastive estimation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 1021–1032.

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.

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

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning*, Proceedings of Machine Learning Research, pages 1597–1607.

Yo Joong Choe, Jiyeon Ham, Kyubyong Park, and Yeoil Yoon. 2019. A neural grammatical error correction system built on better pre-training and sequential transfer learning. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 213–227.

Shamil Chollampatt and Hwee Tou Ng. 2018a. A multilayer convolutional encoder-decoder neural network for grammatical error correction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 5755–5762.

Shamil Chollampatt and Hwee Tou Ng. 2018b. Neural quality estimation of grammatical error correction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2528–2539.

Wanyun Cui, Guangyu Zheng, and Wei Wang. 2020. Unsupervised natural language inference via decoupled multimodal contrastive learning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 5511–5520.

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.

Simon Flachs, Ophélie Lacroix, Helen Yannakoudakis, Marek Rei, and Anders Søgaard. 2020. Grammatical error correction in low error density domains: A new benchmark and analyses. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 8467–8478.

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.

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

Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. 2020. Supervised contrastive learning. In *Proceedings of the Advances in Neural Information Processing Systems*, pages 18661–18673.

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*, 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 the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880.

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

Farhana Ferdousi Liza and Marek Grzes. 2018. Improving language modelling with noise contrastive estimation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 5277–5284.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the Advances in Neural Information Processing Systems*, pages 3111–3119.

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.

Toshikazu Tajiri, Mamoru Komachi, and Yuji Matsumoto. 2012. Tense and aspect error correction for ESL learners using global context. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, pages 198–202.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the Advances in Neural Information Processing Systems*, pages 5998–6008.

Zonghan Yang, Yong Cheng, Yang Liu, and Maosong Sun. 2019. Reducing word omission errors in neural machine translation: A contrastive learning approach. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6191–6196.

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.

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*, pages 156–165.

A Appendix

A.1 Experimental Details

In this part, we will introduce the software packages we have used, implementation details and the training time required for each epoch.

Software configurations: All models are implemented based on Fairseq[5] and PyTorch packages. More specifically, we use Python 3.7 and PyTorch 1.7.0 (or above).

Implementation details: Our implementation of the loss function for contrastive learning is based on cross-entropy loss with label smoothing, which is widely used to avoid overfitting for the model.

Note that each CWEB sentence was annotated by two annotators with two possible corrections. We only use the first correction as the target for negative sampling.

A.2 Dev Set Performance

In this part, we will introduce the validation performance for fine-tuning on the two datasets.

CWEB: The validation performance for fine-tuning on the CWEB dataset using our contrastive learning approach is shown in Table 4.

Systems	CWEB - dev		
	P	R	F0.5
GEC-PD	40.49	23.33	35.25
GEC-BART	52.85	22.34	41.51

Table 4: Validation performance (in %) of the 2 systems on the CWEB dataset.

All results are obtained by using their optimal hyper-parameters. The $F_{0.5}$ scores in this table represent the validation scores on CWEB-dev.

A.3 Performance against Both Annotations

The performance of fine-tuning the GEC systems when calculated against both annotations (non-averaged) using the ERRANT toolkit is shown in Table 5.

A.4 Training Configuration

System			P	R	F0.5
GEC-PD	S	NLL	43.65	31.10	40.39
		CL	**46.57**	**46.65**	**46.59**
	G	NLL	**53.88**	34.24	48.33
		CL	49.76	**44.80**	**48.68**
GEC-BART	S	NLL	53.33	28.10	45.21
		CL	**54.07**	**35.70**	**49.02**
	G	NLL	53.07	26.53	44.22
		CL	**58.02**	**34.94**	**51.25**

Table 5: Test set performance (in %) of the 2 systems on the CWEB dataset, fine-tuned using either NLL or our CL approach.

Configuration	Value
Model Architecture	BART ("large" setting)
Optimizer	Adam (β_1=0.9, β_2=0.999, $\epsilon = 1 \times 10^{-8}$)
Max Sequence Length	1400
Learning Rate	3.00×10^{-5}
Learning Rate Scheduler	Polynomial Decay
Number of Epochs	20
Best epoch	18
Dropout	0.3
Beam search	Beam size 1

Table 6: CL fine-tuning setting for the GEC-BART model

Configuration	Value
Model Architecture	Transformer ("large" setting)
Optimizer	Adam (β_1=0.9, β_2=0.98, $\epsilon = 1 \times 10^{-8}$)
Max Sequence Length	4096
Learning Rate	3.00×10^{-5}
Learning Rate Scheduler	Constant
Number of Epochs	10
Best epoch	3
Dropout	0.3
Beam search	Beam size 5

Table 7: CL fine-tuning setting for the GEC-PD model

[5]https://github.com/pytorch/fairseq/tree/9f4256edf60554afbcaadfa114525978c141f2bd