Focal Training and Tagger Decouple for Grammatical Error Correction

Minghuan Tan[1], **Min Yang**[1*] and **Ruifeng Xu**[2]

[1] Shenzhen Key Laboratory for High Performance Data Mining,
Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences.
[2] Harbin Institute of Technology (Shenzhen).
{mh.tan,min.yang}@siat.ac.cn, xuruifeng@hit.edu.cn

Abstract

In this paper, we investigate how to improve tagging-based Grammatical Error Correction models. We address two issues of current tagging-based approaches, label imbalance issue, and tagging entanglement issue. Then we propose to down-weight the loss of correctly classified labels using Focal Loss and decouple the error detection layer from the label tagging layer through an extra self-attention-based matching module. Experiments on three recent Chinese Grammatical Error Correction datasets show that our proposed methods are effective. We further analyze choices of hyper-parameters for Focal Loss and inference tweaking.

1 Introduction

Grammatical Error Correction (GEC) has been receiving increasing interest from the natural language processing community with the surging popularity of intelligent writing assistants like Grammarly. In the English language, a series of benchmarks (Ng et al., 2013, 2014; Bryant et al., 2019) have been created for the evaluations of different methods. In many other languages, various emerging datasets with language-specific challenges are also attracting plenty of attention (Trinh and Rozovskaya, 2021; Korre and Pavlopoulos, 2022; Náplava et al., 2022; Zhang et al., 2022; Xu et al., 2022; Jiang et al., 2022).

Existing methods for GEC can be categorized into sequence-to-sequence approaches, tagging-based approaches, and hybrid approaches. Sequence-to-sequence approaches require a larger amount of training data and usually rely on synthetic data for pretraining (Rothe et al., 2021; Stahlberg and Kumar, 2021; Kaneko et al., 2020). Tagging-based approaches adopt editing operations between the source text and the target text as training objectives (Malmi et al., 2019; Awasthi et al., 2019). These methods are faster in inference and can achieve competitive performance as sequence-to-sequence approaches. Hybrid models separate the tagging process and the insertion process into two stages, and can easily change word order with an extra pointer network module (Mallinson et al., 2022). In average cases, hybrid approaches can achieve sub-linear inference time.

In this work, we are interested in tagging-based models due to their simplicity and high efficiency and would like to investigate how to further improve their performance. In the literature, there is also work on improving the performance of existing tagging-based models. For example, Tarnavskyi et al. (2022) explored ensembles of recent Transformer encoders in large configurations with various vocabulary sizes. For general-purpose improvements, existing methods range from optimizing training schemes to changing inference techniques. For training, Li et al. (2021) explore how to enhance a model through generating valuable training instances and applying task-specific pretraining strategies. For inference, Sun and Wang (2022) propose Align-and-Predict Decoding (APD) to offer more flexibility for the precision-recall trade-off. From the perspective of system combination, Qorib et al. (2022) propose a simple logistic regression algorithm to combine GEC models effectively.

Different from the methods discussed above, we focus on improving tagging-based models from the perspective of model designing and learning. Currently, GECToR (Omelianchuk et al., 2020) is one of the representative tagging-based models. GECToR contains a pretrained transformer-based encoder with two linear classification layers as the tagger. One of the linear layers is used for label tagging, and the other for error detection. However, we identify the following issues with current tagging-based models: (1) Label imbalance issue. The training labels contain a large portion of easy-to-learn labels and the distribution is highly

* Corresponding author.

Findings of the Association for Computational Linguistics: ACL 2023, pages 5978–5985
July 9-14, 2023

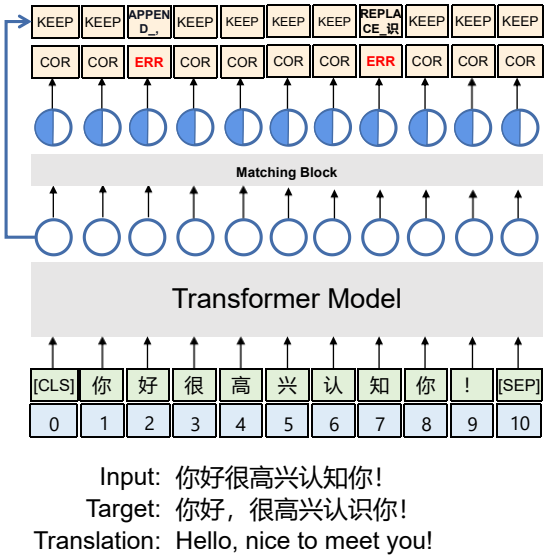

Figure 1: Model structure with error detection decoupled from label tagging.

skewed. Existing learning methods use cross entropy with label smoothing as the loss function which is deemed as sub-optimal for this scenario. (2) In current tagging-based models, sequence labeling and error detection are two linear classification layers over the same hidden representation. This entanglement may also hurt the performance of models.

To solve the problems discussed above, we propose the following modifications to tagging-based models: (1) We use Focal Loss (Lin et al., 2017) to counteract class imbalance and down-weight the loss assigned to correctly classified labels. (2) We decouple error detection from label tagging using extra attention matching module (Wang and Jiang, 2017).

We then verify the effectiveness of the proposed method over three recent Chinese grammatical error correction datasets. Through our experiments, we find that both focal loss and matching mechanisms contribute to performance gain.

2 Method

Suppose we have an edit operation set O for the manipulation of text at token level. Given a piece of source context denoted as $\mathbf{x} = (x_1, x_2, \ldots, x_N)$ and its corrected target sequence $\mathbf{w} = (w_1, w_2, \ldots, w_M)$, to construct a mapping from the $(\mathbf{x}, \mathbf{w})$ to O, we use a tagging scheme $\mathcal{T}$ to first compute alignments between the two sequences. Then we assign each token in the sequence with candidate operations. The tagged label sequence is denoted as $\mathbf{y}_l = \{(y_1, y_2, \ldots, y_N) | y_i \in O\} = \mathcal{T}(\mathbf{x}, \mathbf{w})$. Its corresponding error detection target is named as $\mathbf{y}_d$.

Tagging Scheme. We use a tagging scheme from GECToR (Omelianchuk et al., 2020) with adapted vocabularies and operations by MuCGEC (Zhang et al., 2022). Specifically, the scheme computes an optimal token-level alignment between $\mathbf{x}$ and $\mathbf{w}$. Then for each aligned pair of tokens, there will be four choices for tagging labels: (1) `KEEP` for identical tokens, (2) `DELETE` if the token comes from $\mathbf{x}$ only, (3) `REPLACE_w` if the token from $\mathbf{x}$ is replaced by the one from $\mathbf{w}$, (4) `APPEND_w` if the token comes from $\mathbf{w}$ only. For example, `APPEND_,` and `REPLACE_识` in Figure 1. Notice that if multiple insertions appear within one alignment, only the first one is used for training. The error detection labels are constructed from the tagging labels. It will be a correct label `COR` if tagged as `KEEP` else error label `ERR`.

Encoder. We use a transformer-based encoder to process the tokenized input text. We enclose the context with special tokens `[CLS]` and `[SEP]` and pass them into the BERT model. We use the last layer of BERT as the encoded hidden representation for the context. Considering our tagging system is consistent with GECToR, we also use a mismatched encoder to get hidden representations of the original word, denoted as $\mathbf{H} = (\mathbf{h}_0, \ldots, \mathbf{h}_N)$.

Tagger. Our tagger contains two separate linear classification heads. The label tagging head is conducted over the encoder's hidden representation $\mathbf{H}$ directly:

$$\mathbf{p}_l = Linear(Dropout(\mathbf{H})) \quad (1)$$

The error detection head is decoupled from the label tagging head using an input constructed from $\mathbf{H}$ with a matching mechanism over its self-attended representation:

$$\alpha = softmax(\mathbf{H}\mathbf{H}^\top) \quad (2)$$

$$\mathbf{A} = \alpha^\top \mathbf{H} \quad (3)$$

$$\mathbf{M} = Linear([\mathbf{H}, \mathbf{A}]) \quad (4)$$

$$\mathbf{p}_d = Linear(LayerNorm(\mathbf{M})) \quad (5)$$

Training Objective. In this paper, we choose Focal Loss to down-weight correctly classified labels:

$$FL(p, y) = -\sum_t (1 - p_t)^\gamma \log p_t \quad (6)$$

where γ is a hyper-parameter to control the loss assigned to these labels. Both label tagging and error detection contribute to the final loss. The final loss is a linear combination of label tagging loss and error detection loss:

$$\mathcal{L} = FL(\mathbf{p}_l, \mathbf{y}_l) + \lambda FL(\mathbf{p}_d, \mathbf{y}_d) \quad (7)$$

where λ is a positive hyper-parameter.

3 Experiments

We evaluate our method on three recent Chinese Grammar Error Correction datasets.

3.1 Datasets

We use three recent grammar error correction datasets from the Chinese language, MuCGEC (Zhang et al., 2022), FCGEC (Xu et al., 2022) and MCSCSet (Jiang et al., 2022). Statistics of the three datasets are listed in Table 1. **MuCGEC** is a combination of multiple sources covering diverse types of Chinese grammatical errors. **FCGEC** is a human-annotated corpus with multiple references collected mainly from multiple-choice questions in public school Chinese examinations. **MCSCSet** is a high-quality Chinese Spelling Correction dataset from the medical domain collected from extensive real-world medical queries from Tencent Yidian. The corresponding misspelled sentences are manually annotated by medical specialists.

	Train	Dev	Test
MuCGEC	1,187,605	1,125	5,938
FCGEC	36,340	2,000	3,000
MCSCSet	157,194	19,652	19,650

Table 1: Statistics of used datasets.

The evaluation metric reported in this paper is span level correction $F_{0.5}$ scores evaluated using ChERRANT, a Chinese version of ERRANT[1]. Specifically, ChERRANT computes an optimal sequence of char-level edits with the minimal edit distance given an input sentence and a correction. Then consecutive char-level edits are further merged into span-level, resulting in the following error types: (1) Missing, (2) Redundant, (3) Substitution, (4) Word-Order.

We analyze validation sets of all used datasets using ChERRANT and show the error type distribution in Figure 2. The distribution indicates the difficulty of each dataset which will be discussed in Section 3.3.

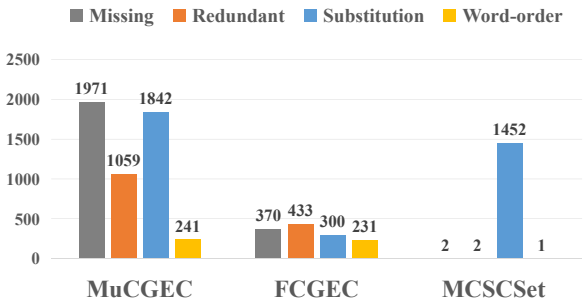

Figure 2: Error type distribution over validation set of used datasets. For MCSCSet, the numbers are divided by 10 to make them fit into the figure.

3.2 Settings

We use a GECToR model released by MuCGEC[2] as our checkpoint. The model uses StructBERT-Large (Wang et al., 2020) as the transformer encoder and it is the best tagging-based model over MuCGEC. It has 7375 labels for token-level operation and 2 labels for error detection. We evaluate our model on the official benchmark websites for MuCGEC[3] and FCGEC[4].

For all our experiments, we use a learning rate of $1e^{-5}$ with batch size 128 and run three epochs for training. The hyper-parameter λ is chosen as 1. The default γ for Focal Loss is chosen as 2. We use the maximum $F_{0.5}$ score over the validation set to choose the best model for evaluation. To be consistent with MuCGEC, we use iterative refinement for five iterations to get the final corrected results. No inference tweaking tricks are used for our main results in Section 3.3. However, we will conduct further analysis on inference tweaking in Section 3.5.

The training costs are computed on a NVIDIA GeForce RTX 3090 GPU. For MuCGEC, the cost is 14 GPU hours . For FCGEC, the cost is 3.5 GPU hours. For MCSCSet, the cost is 3 GPU hours. Our code has been released on Github[5].

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

[2] https://github.com/HillZhang1999/MuCGEC

[3] https://tianchi.aliyun.com/dataset/131328

[4] https://codalab.lisn.upsaclay.fr/competitions/8020

[5] https://github.com/VisualJoyce/TERepo

Model	MuCGEC	FCGEC	MCSCSet
GECToR	39.59	27.45	82.13
+ FL	41.22	29.03	82.47
+ FL + TD	**41.41**	**30.74**	**83.09**

Table 2: Comparison of our proposed methods and the GECToR model over test split.

3.3 Main Results

We use the **GECToR** model as the baseline for comparison. For our proposed methods, we show incremental results of adding Focal Loss (FL) and Tagger Decouple (TD), denoted as **GECToR + FL** and **GECToR + FL + TD**.

We report evaluation scores over the test split of each dataset in Table 2. The baseline scores for MuCGEC and FCGEC are quoted directly from their original papers. The baseline score for MCSCSet is offered by us. We then list the scores of our proposed methods.

The table shows that using Focal Loss for training can improve performance for all datasets. If we further decouple error detection from label tagging, extra gains can be achieved consistently.

It is worth noting that error type distribution reflects the complexity of a specific dataset. For example, MCSCSet is easier than the other two even if it comes from a different domain since the error types are mostly Substitution.

3.4 Analysis over choices of γ

It remains to be answered whether we should choose a larger γ to make the model more aggressive about harder labels. We conduct experiments over MuCGEC using different γ. To evaluate the generalizability of the trained models, we further adopt a zero-shot setting using the test split of FCGEC. We don't use MCSCSet for zero-shot evaluation due to its low domain similarity with MuCGEC and low error type diversity.

In Table 3, we list results for GECToR and GECToR + FL using different γs. On MuCGEC, using larger gamma helps the model to do better in evaluation. However, if we take the zero-shot setting into consideration, the performance over FCGEC is not consistent with the increase of γ. This indicates that larger γ tends to make the model overfit the training data.

Model	MuCGEC	FCGEC
GECToR	39.59	18.06
GECToR + FL(γ=1)	40.53	**22.83**
GECToR + FL(γ=2)	41.22	20.67
GECToR + FL(γ=5)	**41.60**	18.65

Table 3: Comparison of our proposed methods with different γ values over MuCGEC and FCGEC.

3.5 Analysis over inference tweaking

Inference tweaking has been used as a post-processing technique to further improve the performance of tagging-based models. The method searches two hyper-parameters (δ, β) over the validation set. δ is a threshold for sentence-level minimum error probability. β is a positive confidence bias for keeping the source token.

Considering inference tweaking promotes $F_{0.5}$ scores through trading off precision and recall, we conduct experiments to compare how our proposed methods perform against it. We use validation and test split of MuCGEC for the illustration. In Table 4, we list the best scores achieved after applying inference tweaking and place the difference value in the bracket. All scores over the validation split increase by roughly 0.5 points. However, on the test split, the tweaked results are not rising consistently. Although inference tweaking is effective, it's not guaranteed the (δ, β) searched over the validation set works for each specific model.

Model	(δ, β)	Dev	Test
GECToR	(0.40, 0)	35.63 (+0.45)	39.87 (+0.28)
+ FL	(0.35, 0)	38.80 (+0.51)	41.21 (–0.01)
+ FL + TD	(0.35, 0)	39.18 (+0.56)	41.79 (+0.38)

Table 4: Performance differences over validation split and test split after applying inference tweaking.

4 Limitations

In this work, we have been focusing on improving the performance of tagging-based Grammatical Error Correction. Our work has the following limitations: (1) We work on three recent Chinese Grammatical Error Correction datasets. But there are many emerging datasets from various languages. We will add support for these languages on our GitHub repository and make all resources publicly accessible. (2) We point out a limitation of inference tweaking, but it remains to be explored how to

explain the phenomenon and derive better tweaking methods.

5 Conclusion

In conclusion, focal training and tagger decoupling are effective in improving current tagging-based Grammatical Error Correction models. However, it is also important to choose a suitable γ for Focal Loss considering the generalizability of the model. For the widely adopted post-processing technique inference tweaking, it depends on the model whether there will be significant performance gain.

Acknowledgements

This work was partially supported by the National Key Research and Development Program of China (2022YFF0902100), Shenzhen Science and Technology Innovation Program (KQTD20190929172835662), Shenzhen Basic Research Foundation (JCYJ20210324115614039 and JCYJ20200109113441941), and NSFC (no. 92270122).

References

Abhijeet Awasthi, Sunita Sarawagi, Rasna Goyal, Sabyasachi Ghosh, and Vihari Piratla. 2019. Parallel iterative edit models for local sequence transduction. 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 4260–4270, Hong Kong, China. Association for Computational Linguistics.

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.

Wangjie Jiang, Zhihao Ye, Zijing Ou, Ruihui Zhao, Jianguang Zheng, Yi Liu, Bang Liu, Siheng Li, Yujiu Yang, and Yefeng Zheng. 2022. Mcscset: A specialist-annotated dataset for medical-domain chinese spelling correction. In *Proceedings of the 31st ACM International Conference on Information; Knowledge Management*, CIKM '22, page 4084–4088, New York, NY, USA. Association for Computing Machinery.

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.

Katerina Korre and John Pavlopoulos. 2022. Enriching grammatical error correction resources for Modern Greek. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 4984–4991, Marseille, France. European Language Resources Association.

Chong Li, Cenyuan Zhang, Xiaoqing Zheng, and Xuanjing Huang. 2021. Exploration and exploitation: Two ways to improve Chinese spelling correction models. 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 441–446, Online. Association for Computational Linguistics.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2017. Focal loss for dense object detection. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2999–3007.

Jonathan Mallinson, Jakub Adamek, Eric Malmi, and Aliaksei Severyn. 2022. Edit5: Semi-autoregressive text-editing with t5 warm-start. *ArXiv*, abs/2205.12209.

Eric Malmi, Sebastian Krause, Sascha Rothe, Daniil Mirylenka, and Aliaksei Severyn. 2019. Encode, tag, realize: High-precision text editing. 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 5054–5065, Hong Kong, China. Association for Computational Linguistics.

Jakub Náplava, Milan Straka, Jana Straková, and Alexandr Rosen. 2022. Czech grammar error correction with a large and diverse corpus. *Transactions of the Association for Computational Linguistics*, 10:452–467.

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.

Hwee Tou Ng, Siew Mei Wu, Yuanbin Wu, Christian Hadiwinoto, and Joel Tetreault. 2013. The CoNLL-2013 shared task on grammatical error correction. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–12, Sofia, Bulgaria. 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. Association for Computational Linguistics.

Muhammad Qorib, Seung-Hoon Na, and Hwee Tou Ng. 2022. Frustratingly easy system combination for grammatical error correction. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1964–1974, Seattle, United States. 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.

Xin Sun and Houfeng Wang. 2022. Adjusting the precision-recall trade-off with align-and-predict decoding for grammatical error correction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 686–693, Dublin, Ireland. Association for Computational Linguistics.

Maksym Tarnavskyi, Artem Chernodub, and Kostiantyn Omelianchuk. 2022. Ensembling and knowledge distilling of large sequence taggers for grammatical error correction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3842–3852, Dublin, Ireland. Association for Computational Linguistics.

Viet Anh Trinh and Alla Rozovskaya. 2021. New dataset and strong baselines for the grammatical error correction of Russian. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4103–4111, Online. Association for Computational Linguistics.

Shuohang Wang and Jing Jiang. 2017. Machine comprehension using match-lstm and answer pointer. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Wei Wang, Bin Bi, Ming Yan, Chen Wu, Jiangnan Xia, Zuyi Bao, Liwei Peng, and Luo Si. 2020. Structbert: Incorporating language structures into pre-training for deep language understanding. In *International Conference on Learning Representations*.

Lvxiaowei Xu, Jianwang Wu, Jiawei Peng, Jiayu Fu, and Ming Cai. 2022. Fcgec: Fine-grained corpus for chinese grammatical error correction. In *Findings of the Association for Computational Linguistics: EMNLP 2022*.

Yue Zhang, Zhenghua Li, Zuyi Bao, Jiacheng Li, Bo Zhang, Chen Li, Fei Huang, and Min Zhang. 2022. MuCGEC: a multi-reference multi-source evaluation dataset for Chinese grammatical error correction. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3118–3130, Seattle, United States. Association for Computational Linguistics.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
4

☒ A2. Did you discuss any potential risks of your work?
The benchmarks are open and the results are reproducible.

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

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

B ☐ Did you use or create scientific artifacts?

Not applicable. Left blank.

☐ B1. Did you cite the creators of artifacts you used?
Not applicable. Left blank.

☐ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
Not applicable. Left blank.

☐ 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)?
Not applicable. Left blank.

☐ 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?
Not applicable. Left blank.

☐ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Not applicable. Left blank.

☐ 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.
Not applicable. Left blank.

C ☑ Did you run computational experiments?

3

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
3.2

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?
3.2

☑ 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?
3.3-3.4

☑ 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.)?
3.2

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.