GEC-METRICS: A Unified Library for Grammatical Error Correction Evaluation

Takumi Goto, Yusuke Sakai, Taro Watanabe
Nara Institute of Science and Technology (NAIST)
{goto.takumi.gv7, sakai.yusuke.sr9, taro}@is.naist.jp

Abstract

We introduce GEC-METRICS, a library for using and developing grammatical error correction (GEC) evaluation metrics through a unified interface. Our library enables fair system comparisons by ensuring that everyone conducts evaluations using a consistent implementation. Moreover, it is designed with a strong focus on API usage, making it highly extensible. It also includes meta-evaluation functionalities and provides analysis and visualization scripts, contributing to developing GEC evaluation metrics. Our code is released under the MIT license[1] and is also distributed as an installable package[2]. The video is available on YouTube[3].

1 Introduction

Grammatical error correction (GEC) is a task that aims to automatically correct grammatical and surface-level errors, e.g., spelling, tense, expression, and so on (Bryant et al., 2023). GEC serves as a writing support and is being successfully applied in commercial applications such as Grammarly. Therefore, many GEC methods have been proposed, such as sequence-to-sequence models (Katsumata and Komachi, 2020; Rothe et al., 2021), sequence labeling (Awasthi et al., 2019; Omelianchuk et al., 2020), and language model-based approaches (Kaneko and Okazaki, 2023; Loem et al., 2023). To evaluate their performance, some automatic GEC evaluation methods have been proposed (see Section 2.1). These evaluation methods are expected to exhibit a high correlation with human judgments, and their development has become an NLP task in itself.

Although various automatic GEC evaluation methods have been proposed, there is no common library that includes many of the latest studies, making it difficult to compare their performance. Indeed, this has caused several critical issues, such as unfair evaluation, high reproduction costs, and limited extensibility (see Section 3). In fact, most baseline scores are cited from reported results in previous studies, which makes it difficult to reproduce the original scores and to compare methods on new datasets or settings (Maeda et al., 2022).

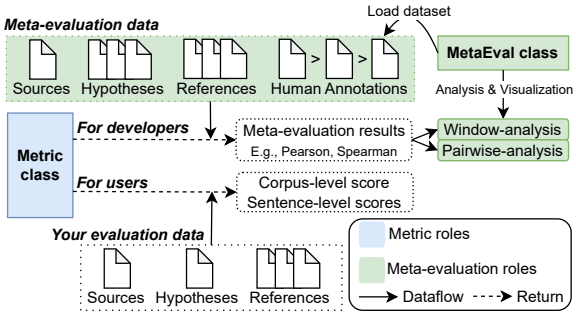


Figure 1: System overview of GEC-METRICS. The *sources* are sentences containing grammatical errors, the *hypotheses* are their corrected version, and the *references* are human-corrected sentences. `Metric` classes support both corpus-level and sentence-level evaluation. The `MetaEval` classes conducts meta-evaluation of metrics, by calculating correlations with human evaluation. These classes also provide analysis and visualize scripts which are useful especially for developers.

While GEC models are being unified through frameworks, UnifiedGEC (Zhao et al., 2025), GEC evaluation metrics remain fragmented and lack a unified implementation, making consistent evaluation difficult. Model development and evaluation are inherently interconnected. For instance, the Hugging Face Transformers (Wolf et al., 2020) has unified various language models into a single framework, while the Hugging Face Evaluate (Von Werra et al., 2022) has similarly consolidated evaluation metrics into a unified library, which has further accelerated and simplified model development. In the same way, a unified framework for the GEC evaluation metric is highly desired.

We introduce GEC-METRICS, a unified frame-

[1] : https://github.com/gotutiyan/gec-metrics
[2] : pip install gec-metrics
[3] : https://youtu.be/cor6dkN6EfI

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations), pages 524–534
July 27 - August 1, 2025

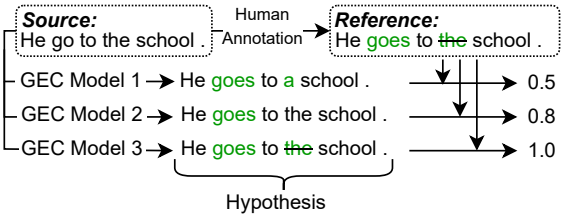


Figure 2: Examples of input/output for GEC evaluation.

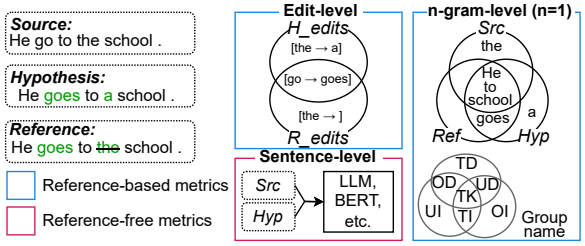


Figure 3: Categories of the current GEC metrics. The edit-level metrics considers the overlap of edits. The n-gram level metrics categorize n-gram into seven groups and use the n-gram count for each group. The sentence-level metrics employ neural models and estimate score without references.

work library that supports a variety of GEC evaluation metrics. It provides a unified interface with many useful features for comparison and developing new evaluation methods. Figure 1 shows the workflow overview of GEC-METRICS. In the figure, each module, i.e., "Metric class" and "MetaEval class", is easily extensible. In addition, we carefully designed GEC-METRICS to ensure transparency and reproducibility. Furthermore, we provide a meta-evaluation interface that simplifies the development of new metrics. Our meta-evaluation experiments using the SEEDA (Kobayashi et al., 2024b) dataset show that GEC-METRICS can efficiently handle various evaluation metrics through a unified interface.

2 Background

2.1 Preliminaries for GEC Evaluation Metrics

Figure 2 shows the overview of the GEC task and its evaluation. The source S is a sentence containing grammatical errors, and hypothesis H is its corrected version made by a GEC model: $H = \text{GECModel}(S)$. Basically, we also have one or more references R, which is a human-corrected sentence, for the evaluation. The goal of the GEC evaluation is to assess the quality of the hypothesis. The evaluation metrics are broadly categorized into reference-based and reference-free metrics, depending on whether they require references R.

$$\text{Score} = \begin{cases} \text{Metric}(H|S,R) & \text{(Ref.-based)} \\ \text{Metric}(H|S) & \text{(Ref.-free)} \end{cases} \quad (1)$$

Edit-level Metrics The reference-based metrics is often conducted by an edit-level evaluation. The GEC field often handles sentence rewriting by decomposing into the granular level of editing. By using automatic edit extraction method such as ERRANT (Felice et al., 2016; Bryant et al., 2017), we extract two edit sets: hypothesis edit set H_{edit} by comparing S and H, and reference edit set R_{edit} from S and R. In Figure 3, you can see there are two edits in each of H_{edit} and R_{edit}. Then, we set the weight w_e for each edit e, and calculate weighted scores: precision, recall, and F_β score [4] by considering the intersection between H_{edit} and R_{edit}: $I = (H_{edit} \cap R_{edit})$ in Equation (2). For instance, a single edit [go → goes] is in both H_{edit} and R_{edit}, thus $I = \{[\text{go} \rightarrow \text{goes}]\}$ in Figure 3.

$$\text{Precision} = \frac{\sum_{e \in I} w_e}{\sum_{e \in H_{edit}} w_e}, \text{Recall} = \frac{\sum_{e \in I} w_e}{\sum_{e \in R_{edit}} w_e} \quad (2)$$

ERRANT (Felice et al., 2016; Bryant et al., 2017) sets $w_e = 1.0$ for all of edits, and **PT-ERRANT** (Gong et al., 2022) computes a weight by BERTScore (Zhang et al., 2020) or BARTScore (Yuan et al., 2021). **GoToScorer** (Gotou et al., 2020) uses the error correction difficulty, which is based on the correction success ratio of the predefined systems, as a weight[5].

n-gram level Metrics The n-gram level metrics have also been employed for the reference-based evaluation. Koyama et al. (2024) provided a generic interpretation by an n-gram Venn diagram. Figure 3 shows an example for $n = 1$. Each group in the Venn diagram is named as True Keep (TK), True Delete (TD), True Insert (TI), Over Delete (OD), Over Insert (OD), Under Delete (UD), Under Insert (UI). In Figure 3, you can see that *He, to, school* are TK, *the* is TD, *a* is OI, and *goes* is TI. Similar to edit-based metrics, n-gram level metrics calculates precision or F_β score from n-gram intersection. **GLEU** (Napoles et al., 2015, 2016) is a precision-based metric and **GREEN** (Koyama et al., 2024) uses F_β score. Further detailed explanations are described in Appendix A.

[4] $F_\beta = \frac{(1+\beta^2)\text{Precision} \times \text{Recall}}{\beta^2 \text{Precision} + \text{Recall}}$

[5] Precisely, the GoToScore additionally considers the non-corrected spans.

Sentence-level Metrics Reference-free metrics are primarily designed as sentence-level metrics and are built using pretrained language models. **SOME** (Yoshimura et al., 2020) focuses on grammaticality, fluency, and meaning preservation; they fine-tuned BERT (Devlin et al., 2019) with regression head respectively optimize to human evaluation directly. **Scribendi** (Islam and Magnani, 2021) evaluates corrected sentences based on perplexity computed by a pretrained language model, and surface-level similarity. **IMPARA** (Maeda et al., 2022) combines similarity scores between S and H with an quality estimation score for H. The quality estimation score is predicted using a BERT-based regression model trained to distinguish different levels of text quality. **LLM-S** (Kobayashi et al., 2024a) performs 5-stage evaluation using a large language model. **LLM-E** (Kobayashi et al., 2024a) inputs edit sequences instead of corrected sentences.

2.2 Meta-Evaluation of GEC Metrics

The quality of GEC evaluation metrics is meta-evaluated by calculating the agreement between human evaluation results and metric-based evaluation results. Meta-evaluation is conducted from two perspectives: *sentence-level* and *system-level*.

In sentence-level evaluation, GEC evaluation methods score the hypothesis of multiple GEC systems associated with each source sentence. Pairwise comparisons of hypotheses are performed for each source, and agreement between the human and metric evaluation results is accumulated over the entire data set. The reported scores are Accuracy (Acc.) and Kendall rank correlation coefficient (τ).

In system-level evaluation, the focus is on comparing the overall relative quality of systems. System-level rankings are generally computed by averaging or accumulating sentence-level results. The metrics for system-level evaluation are Pearson (r) and Spearman (ρ) correlation coefficients.

To facilitate this, some meta-evaluation datasets have been proposed, such as GJG15 (Grundkiewicz et al., 2015) and SEEDA (Kobayashi et al., 2024b), which are derived from CoNLL-2014 shared task submissions (Ng et al., 2014). Nonetheless, the number of available meta-evaluation datasets remains limited. One contributing factor is the lack of a unified framework for GEC evaluation metrics, which hinders consistent and comprehensive validation and increases the cost of implementing baselines when constructing meta-evaluation datasets.

Metric	Reported paper	r	ρ
Scribendi	Islam and Magnani (2021)	.951	.940
	Maeda et al. (2022)	.303	.729
	Kobayashi et al. (2024b)	.890	.923
IMPARA	Maeda et al. (2022)	.974	.934
	Kobayashi et al. (2024b)	.961	.965

Table 1: Previously reported meta-evaluation results on GJG15 (Grundkiewicz et al., 2015). The r and ρ are Pearson's correlation and Spearman rank correlation. The results are inconsistent across studies, due to a lack of implementations and an open pre-trained model.

3 Problems of Existing Implementations

Inconsistent interfaces. Although many GEC evaluation metrics have been proposed, their implementations are designed with their own interfaces and lack compatibility, such as input/output formats. This makes cross-metric evaluation difficult and limits multifaceted discussions. For example, recent evaluations of GEC model development heavily rely on ERRANT, while other metrics with high correlation to human evaluation, such as IMPARA, are seldom reported. If the interfaces were unified, the complex experimental procedures caused by inconsistent implementations could be eliminated, which would facilitate better development and evaluation of GEC models.

Lack of official resources. Some metrics do not provide official resources. For example, Scribendi and LLM-{S, E} did not release their implementations, and IMPARA did not provide its fine-tuned weights. Therefore, we must reproduce these metrics, which can lead to discrepancies in reported results, as shown in Table 1. Moreover, some metrics no longer work with their official code, such as GLEU, which is written in Python 2. To avoid the cost of reproduction, most papers cite scores from previous studies, which compromises transparency.

API support. Since most original implementations are developed for specific experiments, they are typically intended to be executed using CLI-based scripts. As a result, they do not support an extensible ecosystem such as APIs, which limits their flexibility and reusability. When evaluation metrics are used as components in other methods, such as a reward function in reinforcement learning (Sakaguchi et al., 2017), a utility function in MBR decoding (Raina and Gales, 2023), or a quality estimation model for ensembling (Qorib and Ng, 2023), APIs facilitate easier integration.

4 GEC-METRICS

Our library, GEC-METRICS, compiles recent GEC evaluation methods into a unified interface. It supports not only the use of GEC metrics by users and GEC system developers but also meta-evaluation for GEC metric developers. GEC-METRICS supports both command-line usage and Python API access, enabling integration into a wide range of applications. It resolves all the limitations of existing implementations highlighted in Section 3. We have verified that the results obtained using GEC-METRICS are consistent with those from official implementations for all publicly available metrics.

4.1 Supported Methods

GEC evaluation metrics. GEC-METRICS supports all of ten metrics described in Section 2.1. For reference-based metrics, it supports **ERRANT**, **PT-ERRANT**, and **GoToScorer** as edit-level metrics, **GLEU** and **GREEN** as n-gram level metrics. For reference-free metrics, it supports **SOME**, **Scribendi**, **IMPARA**, **LLM-S**, and **LLM-E**[6] as sentence-level metrics. We carefully designed the library for extensibility and ease of changing hyperparameters and base models, supporting various use cases such as modifying the value of n in n-gram or switching the language models. Notably, LLM-{S, E} support the OpenAI and Gemini APIs, as well as all causal language models available in Hugging Face Transformers (Wolf et al., 2020), and also provides simplified prompts for applying to any data and scenario, as detailed in Appendix C.

Meta-evaluation. GEC-METRICS also supports all of two meta-evaluation frameworks: **GJG15** and **SEEDA** as introduced in Section 2.2. It accommodates all detailed configurations for each framework, ensuring comprehensive support. Specifically, both datasets contain human Expected Wins (Bojar et al., 2013) rankings and human TrueSkill (Herbrich et al., 2006) rankings. GJG15 adopts Expected Wins as the final human evaluation result, while SEEDA uses TrueSkill. While system-level evaluation scores are typically reported using simple aggregation methods such as averaging, our library also provides the option to follow Goto et al. (2025) by aggregating system-level results using either ***Expected Wins*** or ***TrueSkill***. Furthermore, SEEDA includes two evaluation settings: **SEEDA-S**, where human evaluation is conducted at the sentence level, and **SEEDA-E**, where evaluation is performed at the edit level. It also provides two configurations: **`Base`** and **`+Fluency`**. GEC-METRICS fully supports all of these settings, enabling easy assessment of evaluation performance under diverse conditions.

[6]Notably, our implementations of LLM-{S, E} are the first publicly available resource of Kobayashi et al. (2024a). We contacted the authors, received some codes and prompts, and had several discussions to clarify the implementation details. We are deeply grateful for their support and contributions.

```python
from gec_metrics.metrics import ERRANT
from gec_metrics.meta_eval import MetaEvalSEEDA
metric = ERRANT(ERRANT.Config(beta=0.5))
SRCS = ["He go to the school."] * 100
HYPS = ["He goes to the school."] * 100
REFS = [["He goes to school."] * 100]

# Corpus-level scoring
system_score: float = metric.score_corpus(
  sources=SRCS, hypotheses=HYPS, references=REFS
)  # Output: 0.833
# Sentence-level scoring
sent_score: list[float]  = metric.score_sentence(sources=SRCS, hypotheses=HYPS, references=REFS
)  # Output: [0.833, 0.833, ...]

### Meta-evaluation on SEEDA ###
meta = MetaEvalSEEDA(
  MetaEvalSEEDA.Config(system='base')
)
# System-level meta-evaluation
meta_system = meta.corr_system(metric)
print(f"SEEDA-S: {meta_system.ts_sent}")
# Output: MetaEvalBase.Corr(pearson=0.539, spearman=0.342)
# Sentence-level meta-evaluation
meta_sentence = meta.corr_sentence(metric)
print(f"SEEDA-S: {meta_sentence.sent}")
# Output:  MetaEvalBase.Corr(accuracy=0.594, kendall=0.188)
```

Listing 1: An example of the implementation of evaluation and meta-evaluation using ERRANT as a metric and SEEDA as a meta-evalution framework.

4.2 Interfaces

GEC-METRICS supports three types of interfaces: CLI, Python API, and GUI. While we primarily focus on the Python API, the other interfaces are demonstrated in Appendix D.

Listing 1 shows an example Python code for evaluation using ERRANT and meta-evaluation using SEEDA. Evaluation can be performed simply by passing a list of sentences to the `score_**()` functions in L8 and L12. Similarly, meta-evaluation is supported through a simple interface, where the `corr_**()` functions in L20 and L23 take a metric

Metric	System-level										Sentence-level									
	GJG15		SEEDA-S				SEEDA-E				GJG15		SEEDA-S				SEEDA-E			
			Base		+Fluency		Base		+Fluency				Base		+Fluency		Base		+Fluency	
	r	ρ	r	ρ	r	ρ	r	ρ	r	ρ	Acc.	τ	Acc.	τ	Acc.	τ	Acc.	τ	Acc.	τ
ERRANT	.647	.687	.539	.343	-.592	-.156	.682	.643	-.508	.033	.654	.307	.594	.189	.544	.087	.608	.217	.558	.116
PT-ERRANT	.704	.786	.700	.629	-.548	.077	.788	.874	-.471	.231	.655	.310	.583	.166	.540	.080	.592	.184	.550	.100
GoToScorer	.668	.615	.726	.601	.439	.499	.816	.762	.514	.635	.579	.159	.550	.100	.511	.021	.563	.126	.524	.048
GREEN	.786	.720	**.925**	.881	.185	.569	.932	.965	.252	.618	.660	.319	.600	.199	.552	.105	.574	.148	.537	.073
GLEU	.706	.626	.886	**.902**	.155	.543	.912	.944	.232	.569	.673	.346	.672	.343	.616	.231	.673	.347	.625	.249
SOME	**.957**	**.923**	.892	.867	**.931**	.916	.901	.951	**.943**	.969	**.779**	**.559**	**.778**	**.555**	**.765**	**.531**	**.766**	**.532**	**.754**	**.509**
IMPARA	.956	.885	.916	**.902**	.887	.938	.902	.965	.900	**.978**	.747	.495	.753	.506	.738	.475	.752	.504	.743	.486
Scribendi	.855	.835	.620	.636	.604	.714	.825	.839	.715	.842	.728	.457	.660	.320	.623	.245	.672	.345	.648	.295
GPT-4-E	.383	.357	.085	.027	-.817	-.393	.312	.307	-.764	-.279	.473	-.053	.520	.041	.582	.165	.538	.077	.591	.183
GPT-4-S	-.073	-.181	.848	.748	.322	.613	.923	.958	.390	.714	.674	.348	.607	.214	.582	.165	.603	.206	.591	.183
Gemini-S	-.205	-.318	.776	.622	.461	.714	.891	.902	.521	.802	.628	.257	.597	.195	.577	.154	.600	.200	.575	.150
Qwen2.5-S	-.247	-.274	.920	.839	.788	**.942**	.893	.916	.790	.930	.595	.191	.588	.177	.574	.148	.594	.189	.576	.153
Ensemble (aboves w/o LLM)	.808	.840	.887	.823	.350	.691	**.953**	**.984**	.436	.803	–	–	–	–	–	–	–	–	–	–

Table 2: Meta-evaluation results using our GEC-METRICS library. We use Pearson (r) and Spearman (ρ) for the system-level meta-evaluation, and accuracy (Acc.) and Kendall (τ) for the sentence-level meta-evaluation. **Bold** is the highest value in each column, underline is the second one.

instance as input. In addition, parameters and settings are separated via a `**.Config() dataclass`. If switching to another metric, the process is simple and easy, thanks to the unified API interface.

Extensibility. All classes are implemented by inheriting from an abstract class. The abstract class defines the minimal required methods, such as `score_sentence()`, which must be overridden in the derived classes. This ensures that the interface remains consistent regardless of who implements the metric. Similarly, adding new meta-evaluation also requires only minimal implementation[7].

Reproducibility. CLI supports configuration input in YAML format. This allows users to share the exact settings used for running a metric, e.g., what model is used, contributing to high reproducibility.

4.3 Analyses and Visualizations

Meta-evaluation is not limited to correlation coefficients such as Pearson or Kendall but can also involve more detailed analyses. For example, the *window analysis* (Kobayashi et al., 2024b) enables discussions on evaluation performance by focusing on competitive systems in human evaluation, and the *edit-level attribution* shows which edit operation a metric focuses on in the evaluation (Goto et al., 2024). GEC-METRICS provides tools for such analyses and result visualization.

[7]We provide the documentation, including usage instructions, detailed API references, examples, and quick start guides: https://gec-metrics.readthedocs.io/en/latest/index.html.

Pairwise-analysis. Previous sentence-level meta-evaluations have primarily focused on Accuracy and Kendall's τ, which reflect overall agreement but offer limited interpretability. Therefore, we propose *pairwise analysis*, which focuses on the relationship between differences in human rankings and agreement rates in sentence-level meta-evaluation. The difference between human- and metric-scored rankings for the same source can be calculated for each system pair, allowing agreement to be grouped and analyzed by ranking difference. Intuitively, the greater the difference in rankings assigned by humans, the more accurately a metric is expected to make judgments, reflecting how well it aligns with human evaluation at the sentence level.

5 Experiments

Settings. Using our GEC-METRICS library, we conducted meta-evaluations of GEC evaluation metrics. We employed all metrics listed in Section 4.1, and used GJG15 and SEEDA as meta-evaluation datasets. For system-level evaluation, we used the Expected Wins rankings from GJG15 and the TrueSkill rankings from SEEDA-{S, E}. Appendix B provides the detailed experimental settings, which serve as the default configuration and generally follow those used in the original papers.

Extensive evaluation for LLM-{S, E}. We conducted several variations using different LLMs to provide extensive evaluation for LLM-{S, E} (Kobayashi et al., 2024a). Notably, we re-

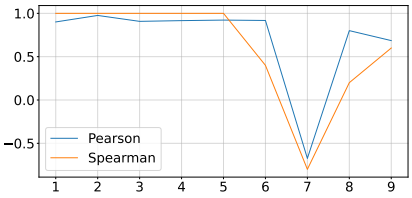
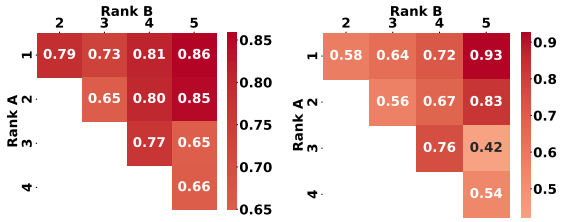


Figure 4: Window-analysis results for IMPARA. The x-axis indicates the start rank in the human-evaluation, and y-axis means Pearson (blue line) or Spearman (orange line) correlation.



(a) IMPARA (b) ERRANT

Figure 5: Results of the pairwise-analysis. (a) shows the agreement rates between IMPARA and SEEDA-S annotation, and (b) shows the rates between ERRANT and SEEDA-E.

port results on GJG15 for the first time, revealing that the trend differs from SEEDA. Detailed motivations and settings are provided in Appendix C. We use `gpt-4o-mini-2024-07-18` (**GPT-4-S, GPT-4-E**) (OpenAI et al., 2024), `gemini-2.0-flash` (**Gemini-S**) (Team et al., 2025), and `Qwen2.5-14B-Instruct` (**Qwen2.5-S**) (Qwen et al., 2025) to emphasize extendability of our library for other language models.

Results. Table 2 shows the experimental results. ERRANT and PT-ERRANT show a higher correlation with SEEDA-E than with SEEDA-S, emphasizing the importance of aligning the evaluation granularity between human and automatic evaluations. Meanwhile, under the +Fluency setting, the correlation becomes negative, indicating the difficulty of evaluating GEC systems that focus on improving fluency. In contrast, SOME and IMPARA achieve high correlations even in the +Fluency setting. These results align with the trends reported in SEEDA (Kobayashi et al., 2024b). On the other hand, for LLM-based metrics, while they achieve relatively high correlations in SEEDA, their performance is lower in GJG15. Our study is the first to apply LLM-based metrics to GJG15, suggesting that the evaluation capability of LLMs does not necessarily generalize and that there is room for improvement. Similarly, GPT-4-E fails to reproduce the results reported by (Kobayashi et al., 2024a), indicating the need for further discussion on the validity of the approach. Figure 4 shows the window-analysis results for IMPARA. We used human TrueSkill rankings of SEEDA-S and used 4 as the window size. An observation is that the correlations suddenly drops at $x = 7$, which is consistent with Kobayashi et al.'s (2024b) observation.

Metric Ensemble. GMEG-Metric (Napoles et al., 2019) proposed an ensemble approach for evaluation metrics and reported robust performance across different domains. Given that new metrics continue to be developed after this work, ensemble techniques are expected to remain important for achieving reliable evaluations. Since ensembling requires results from multiple metrics, using a unified implementation like GEC-METRICS facilitates experimentation. As a simple experiment to explore this, we consider using the negative average ranking across different metrics as the final evaluation score. For instance, if a system is ranked 2nd by a metric and 1st by another metric, its final evaluation score would be -1.5. By ensembling metrics other than LLM-based metrics listed in Table 2, we achieved a Spearman rank correlation of 0.984 on SEEDA-E. This is the highest correlation in our experiment. This short experiment shows that GEC-METRICS facilitates the exploration of novel evaluation metrics.

Analysis for Sentence-level Scores. Figure 5 presents the results of an experiment using human evaluation data from the SEEDA dataset. Rank A and Rank B correspond to the human-assigned rankings of a hypothesis pair. Both of results are showing a trend where agreement increases as the difference in rankings grows (toward the upper right side in each figure). This suggests that current metrics reflect human evaluative tendencies, but there is room for improvement in distinguishing minor differences in quality.

6 Conclusion

In this paper, we proposed a library, GEC-METRICS, to address issues in evaluation caused by inconsistencies in existing metric implementations and the lack of official resources. GEC-METRICS is designed with a strong focus on API usability, making it easier to apply not only for evaluation but also for other purposes. Furthermore, it supports developers in improving evaluation metrics by pro-

viding an interface for meta-evaluation. We hope that our library will lead to further diverse applications and advanced research. We will continue to develop our library, incorporating diverse methods and languages, and contribute to the community.

Ethics Statement and Broader Impact

Contribution for research ethics. Using GEC-METRICS improves the reproducibility and transparency of experiments, which is crucial from a research ethics standpoint. The inclusion of questions about implementation and experimental settings in the ACL Rolling Review checklist[8] highlights the community's emphasis on these aspects. By continuing to maintain and develop metric implementations, GEC-METRICS aims to support and strengthen these efforts.

Impacts for the community. GEC-METRICS serves as a powerful tool for researchers to easily develop evaluation methods. It also accelerates their application in the GEC field, including bias investigations, integration with learning and inference methods such as reinforcement learning and ensembling, and use as a scorer in shared tasks. In fact, it has already been adopted as a scorer in a shared task competition at a domestic Japanese conference that examined metric vulnerabilities[9]. These cases demonstrate that GEC-METRICS is beginning to contribute to advancing research. At the same time, we recognize the importance of maintenance and management. We are committed to providing long-term support and actively incorporating new methods and pull requests responsibly.

License. We have also confirmed that there are no licensing issues with the code, methods, or data used in our implementation. GEC-METRICS is released under the MIT license.

Acknowledgments

We gratefully appreciate Masamune Kobayashi, the author of SEEDA and LLM-{S, E} (Kobayashi et al., 2024a,b) for generously sharing code and prompts, as well as engaging in extended discussions, which served as a valuable reference during the development of our library. We also thank the anonymous reviewers for their valuable comments and suggestions. The architecture design of GEC-METRICS is inspired by MBRS (Deguchi et al., 2024). This work has been supported by JST SPRING. Grant Number JPMJSP2140.

[8] https://aclrollingreview.org/responsibleNLPresearch/

[9] https://sites.google.com/view/nlp2025ws-langeval/task/gec

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.

Ondřej Bojar, Christian Buck, Chris Callison-Burch, Christian Federmann, Barry Haddow, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. 2013. Findings of the 2013 Workshop on Statistical Machine Translation. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 1–44, Sofia, Bulgaria. Association for Computational Linguistics.

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 (Volume 1: Long Papers)*, pages 793–805, Vancouver, Canada. Association for Computational Linguistics.

Christopher Bryant, Zheng Yuan, Muhammad Reza Qorib, Hannan Cao, Hwee Tou Ng, and Ted Briscoe. 2023. Grammatical error correction: A survey of the state of the art. *Computational Linguistics*, pages 643–701.

Hiroyuki Deguchi, Yusuke Sakai, Hidetaka Kamigaito, and Taro Watanabe. 2024. mbrs: A library for minimum Bayes risk decoding. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 351–362, Miami, Florida, USA. 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, Christopher Bryant, and Ted Briscoe. 2016. Automatic extraction of learner errors in ESL sentences using linguistically enhanced alignments. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 825–835, Osaka, Japan. The COLING 2016 Organizing Committee.

Peiyuan Gong, Xuebo Liu, Heyan Huang, and Min Zhang. 2022. Revisiting grammatical error correction evaluation and beyond. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 6891–6902, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Takumi Goto, Yusuke Sakai, and Taro Watanabe. 2025. Rethinking evaluation metrics for grammatical error correction: Why use a different evaluation process than human? *Preprint*, arXiv:2502.09416.

Takumi Goto, Justin Vasselli, and Taro Watanabe. 2024. Improving explainability of sentence-level metrics via edit-level attribution for grammatical error correction. *Preprint*, arXiv:2412.13110.

Takumi Gotou, Ryo Nagata, Masato Mita, and Kazuaki Hanawa. 2020. Taking the correction difficulty into account in grammatical error correction evaluation. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2085–2095, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Roman Grundkiewicz, Marcin Junczys-Dowmunt, and Edward Gillian. 2015. Human evaluation of grammatical error correction systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 461–470, Lisbon, Portugal. Association for Computational Linguistics.

Ralf Herbrich, Tom Minka, and Thore Graepel. 2006. Trueskill™: A bayesian skill rating system. In *Advances in Neural Information Processing Systems*, volume 19. MIT Press.

Md Asadul Islam and Enrico Magnani. 2021. Is this the end of the gold standard? a straightforward reference-less grammatical error correction metric. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3009–3015, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Masahiro Kaneko and Naoaki Okazaki. 2023. Reducing sequence length by predicting edit spans with large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 10017–10029, Singapore. Association for Computational Linguistics.

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, Suzhou, China. Association for Computational Linguistics.

Masamune Kobayashi, Masato Mita, and Mamoru Komachi. 2024a. Large language models are state-of-the-art evaluator for grammatical error correction. In *Proceedings of the 19th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2024)*, pages 68–77, Mexico City, Mexico. Association for Computational Linguistics.

Masamune Kobayashi, Masato Mita, and Mamoru Komachi. 2024b. Revisiting meta-evaluation for grammatical error correction. *Transactions of the Association for Computational Linguistics*, 12:837–855.

Shota Koyama, Ryo Nagata, Hiroya Takamura, and Naoaki Okazaki. 2024. n-gram F-score for evaluating grammatical error correction. In *Proceedings of the 17th International Natural Language Generation Conference*, pages 303–313, Tokyo, Japan. Association for Computational Linguistics.

Mengsay Loem, Masahiro Kaneko, Sho Takase, and Naoaki Okazaki. 2023. Exploring effectiveness of GPT-3 in grammatical error correction: A study on performance and controllability in prompt-based methods. In *Proceedings of the 18th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2023)*, pages 205–219, Toronto, Canada. Association for Computational Linguistics.

Koki Maeda, Masahiro Kaneko, and Naoaki Okazaki. 2022. IMPARA: Impact-based metric for GEC using parallel data. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 3578–3588, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Courtney Napoles, Maria Nădejde, and Joel Tetreault. 2019. Enabling robust grammatical error correction in new domains: Data sets, metrics, and analyses. *Transactions of the Association for Computational Linguistics*, 7:551–566.

Courtney Napoles, Keisuke Sakaguchi, Matt Post, and Joel Tetreault. 2015. Ground truth for grammatical error correction metrics. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 588–593, Beijing, China. Association for Computational Linguistics.

Courtney Napoles, Keisuke Sakaguchi, Matt Post, and Joel Tetreault. 2016. Gleu without tuning. *Preprint*, arXiv:1605.02592.

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.

OpenAI, :, Aaron Hurst, Adam Lerer, Adam P. Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, Aleksander Mądry, Alex Baker-Whitcomb, Alex Beutel, Alex Borzunov, Alex Carney, Alex Chow, Alex Kirillov, and 401 others. 2024. Gpt-4o system card. *Preprint*, arXiv:2410.21276.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Muhammad Reza Qorib and Hwee Tou Ng. 2023. System combination via quality estimation for grammatical error correction. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12746–12759, Singapore. Association for Computational Linguistics.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, and 25 others. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, and 1 others. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Vyas Raina and Mark Gales. 2023. Minimum Bayes' risk decoding for system combination of grammatical error correction systems. In *Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 105–112, Nusa Dua, Bali. 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.

Keisuke Sakaguchi, Matt Post, and Benjamin Van Durme. 2017. Grammatical error correction with neural reinforcement learning. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 366–372, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M. Dai, Anja Hauth, Katie Millican, David Silver, Melvin Johnson, Ioannis Antonoglou, Julian Schrittwieser, Amelia Glaese, Jilin Chen, Emily Pitler, Timothy Lillicrap, Angeliki Lazaridou, and 1332 others. 2025. Gemini: A family of highly capable multimodal models. *Preprint*, arXiv:2312.11805.

Leandro Von Werra, Lewis Tunstall, Abhishek Thakur, Sasha Luccioni, Tristan Thrush, Aleksandra Piktus, Felix Marty, Nazneen Rajani, Victor Mustar, and Helen Ngo. 2022. Evaluate & evaluation on the hub: Better best practices for data and model measurements. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 128–136, Abu Dhabi, UAE. Association for Computational Linguistics.

Brandon T. Willard and Rémi Louf. 2023. Efficient guided generation for large language models. *Preprint*, arXiv:2307.09702.

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, and 3 others. 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.

Ryoma Yoshimura, Masahiro Kaneko, Tomoyuki Kajiwara, and Mamoru Komachi. 2020. SOME: Reference-less sub-metrics optimized for manual evaluations of grammatical error correction. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6516–6522, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Weizhe Yuan, Graham Neubig, and Pengfei Liu. 2021. Bartscore: Evaluating generated text as text generation. In *Advances in Neural Information Processing Systems*, volume 34, pages 27263–27277. Curran Associates, Inc.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.

Yike Zhao, Xiaoman Wang, Yunshi Lan, and Weining Qian. 2025. UnifiedGEC: Integrating grammatical

error correction approaches for multi-languages with a unified framework. In *Proceedings of the 31st International Conference on Computational Linguistics: System Demonstrations*, pages 37–45, Abu Dhabi, UAE. Association for Computational Linguistics.

A Details for $n$gram level metrics.

GLEU is a precision-based metric. By using the Venn diagram in the Figure 3, it is formulated by:

$$p_n = \frac{\mathrm{TI}_n + \mathrm{TK}_n - \mathrm{UD}_n}{\mathrm{TI}_n + \mathrm{TK}_n + \mathrm{OI}_n + \mathrm{UD}_n}. \quad (3)$$

Note that $\mathrm{TI}_n, \mathrm{TK}_n \ldots$ represents the n-gram count of each group. The p_n is a precision for n-gram and is usually computed for each n from 1 to 4. Then, the brevity penalty (Papineni et al., 2002) is taken into account after taking the geometric mean. **GREEN** (Koyama et al., 2024) is also an n-gram-level metric, but it computes the precision, recall, and F_β score:

$$\mathrm{Precision}_n = \frac{\mathrm{TI}_n + \mathrm{TD}_n + \mathrm{TK}_n}{\mathrm{TI}_n + \mathrm{TD}_n + \mathrm{TK}_n + \mathrm{OI}_n + \mathrm{OD}_n}, \quad (4)$$

$$\mathrm{Recall}_n = \frac{\mathrm{TI}_n + \mathrm{TD}_n + \mathrm{TK}_n}{\mathrm{TI}_n + \mathrm{TD}_n + \mathrm{TK}_n + \mathrm{UI}_n + \mathrm{UD}_n}, \quad (5)$$

$$F_\beta = \frac{\left(1+\beta^2\right)\mathrm{Precision} \times \mathrm{Recall}}{\beta^2 \mathrm{Precision} + \mathrm{Recall}}. \quad (6)$$

After calculating the geometric mean for each of precision and recall using n from 1 to 4, the F_β score is calculated.

B Details of experimental setup

For the reference-based metrics, we used the official two references of CoNLL-2014 shared task (Ng et al., 2014). The below describes the detail exoerimental settings for each metric.

ERRANT. We use `errant==3.0.0`. Note that the extraction ways of edits have changed slightly between $\geq$v3.0.0 and $<$v3.0.0. We use $F_{0.5}$ as the score. The sentence-level scores are computed by choosing the best reference, which makes the highest $F_{0.5}$ score, for each source sentence.

PT-ERRANT. PT-ERRANT uses F-score of the BERTScore with `bert-base-uncased` for the edit-level weight computation. It rescales the weights by the baseline, but does not use the idf importance weighting. These are the same configurations as the official implementation[10]. After computing edit-level weights, we compute weighed precision, recall, and $F_{0.5}$ score as in ERRANT. The computation method of the sentence-level scores is also the same as that of ERRANT.

GoToScorer. We used the first reference and all system outputs, including input sentences, for calculating the error correction difficulty.

GLEU. We use word-level GLEU and set 500 as the iteration count. The maximum n is 4 for n-gram. The sentence-level scores are defined as the average of each reference.

GREEN. We use word-level GREEN and $F_{2.0}$.

Scribendi. We use GPT-2 (Radford et al., 2019) as a language model to compute perplexity. The threshold for the maximum values of Levenshtein-distance ratio and token sort ratio is 0.8.

SOME. We use the official pre-trained weights, which are available from the official repository [11]. The weights for the grammaticality score, fluency score, and meaning preservation score are set to 0.55, 0.43, and 0.02, respectively.

IMPARA. For IMPARA, we reproduce the training experiments because no trained model is publicly available. As follows Maeda et al. (2022), we generated 4,096 instances using CoNLL-2013 (Ng et al., 2013) as the seed corpus, and split them into 8:1:1 for training, development, and evaluation sets. Thus, we used 3,276 instances as training data to fine-tune `bert-base-cased` and made public the pre-trained weights[12]. GEC-METRICS does not contain the training scripts, but we make them public in a separate repository[13]. `bert-base-cased` is used for computing the similarity score with the threshold 0.9.

LLM-S and LLM-E. For GPT-4-S, we use `beta.chat.completions.parse` API for the OpenAI models and use OUTLINES library (Willard and Louf, 2023)[14] for the HuggingFace models, to ensure the output is in JSON structure. While Kobayashi et al. (2024a) uses `gpt-4-1106-preview`, we used `gpt-4o-mini-2024-07-18` model in our experiments to avoid using it due to the high

[10] https://github.com/pygongnlp/PT-M2

[11] https://github.com/kokeman/SOME

[12] https://huggingface.co/gotutiyan/IMPARA-QE

[13] https://github.com/gotutiyan/IMPARA

[14] https://github.com/dottxt-ai/outlines

The goal of this task is to rank the presented targets based on the quality of the sentences.
After reading the source sentence and target sentences, please assign a score from a minimum of 1 point to a maximum of 5 points to each target based on the quality of the sentence (note that you can assign the same score multiple times).
source
[SOURCE]
targets
... <omitted>

Figure 6: Our modified instruction for LLM-S.

experimental cost. We believe that not everyone can afford to use expensive models.

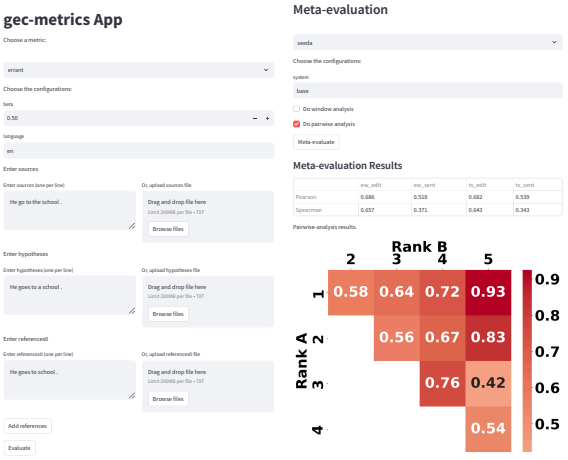
C Our Modifications of the LLM-based Metrics

As described in Section 5, we have made modifications to the LLM-based metric proposed by Kobayashi et al. (2024a). The first modification is the exclusion of contextual information from preceding and following sentences. Some datasets do not include surrounding context, and Kobayashi et al. (2024a) does not specify how to handle such cases. To ensure that evaluation is feasible for any dataset, we employed a prompt that does not incorporate contextual information, which also necessitated changes to the instruction text. We show the instruction text in Figure 6.

The second modification clarifies the sampling method for input correction hypotheses. Their metric accepts up to five hypotheses simultaneously, but when evaluating a large number of systems, the number of different correction hypotheses may exceed five. In such cases, some method of selecting five sentences is required to proceed with evaluation. Kobayashi et al. (2024a) describes only the experimental setup for meta-evaluation using SEEDA, where pre-sampled correction hypotheses are used as input. However, this approach cannot be directly applied when evaluating a different set of systems or when working with a different dataset. Since Kobayashi et al. (2024a) does not define an experimental procedure for such scenarios, we adopted a method that selects five sentences based on their frequency, where frequency is defined as the number of systems that produce the same correction hypothesis. Note that multiple systems may output the same corrected sentence. The selected hypotheses are all unique, and the evaluation score assigned to each hypothesis is expanded across all systems that produced it. By selecting correction hypotheses with higher frequency, we maximize the number of systems that can be evaluated. We use a single RTX3090 for experiments.

```
gecmetrics-eval --src <src> \
    --hyps <hyp1> <hyp2> ... \
    --refs <ref1> <ref2> ... \
    --metric errant \
    --config config.yaml
```

Listing 2: Commandline usage of GEC-METRICS . Each variable within < > indicates a path to a raw text file. You can use another metrics by specifying the - -metric argument e.g. "- -metric impara".



(a) Metric GUI (b) Meta-evaluation GUI

Figure 7: GUI of GEC-METRICS . (a) is for metrics, and (b) is for meta-evaluation, which includes visualization of the analysis. They are actually combined on a single page.

D CLI and GUI Interfaces

Listing 2 provides an example of CLI. It can receive raw text files as inputs, the metric id to `--metric`, and YAML-based configuration input using the `--config` argument.

Figure 7 shows a GUI example, which is developed via STREAMLIT library[15]. You can easily perform the evaluation for any dataset and the meta-evaluation, without coding. Furthermore, it has visualization features for the analysis results of meta-evaluation: window-analysis and pairwise-analysis, such as shown in Figure 5. The code for GUI is provided in a separate repository: https://github.com/gotutiyan/gec-metrics-app.

[15] https://github.com/streamlit/streamlit