Rationalize and Align: Enhancing Writing Assistance with Rationale via Self-Training for Improved Alignment

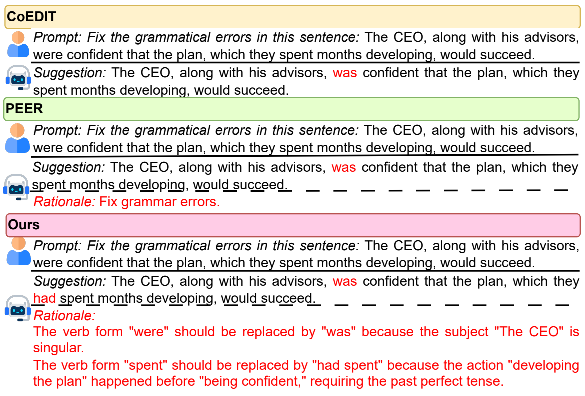
Hannan Cao **Hai Ye** **Hwee Tou Ng**
Department of Computer Science, National University of Singapore
caoh@u.nus.edu, e0684045@u.nus.edu
nght@comp.nus.edu.sg

Abstract

A Writing Assistant (WA) is a system that offers writing suggestions based on user instructions. Existing WAs are typically built by training large language models (LLMs) on domain-specific instruction data through supervised fine-tuning (SFT) only. However, SFT optimizes models to match a single reference, failing to capture the inherent flexibility of text editing, where multiple valid revisions exist. Therefore, solely relying on SFT limits WA performance. To address this limitation, we propose the **Rationalize and Align** framework, which enhances the WA performance with rationale (*i.e.,* linguistic explanations) and alignment. Our framework automatically generates the rationale and preference data for writing tasks via distillation and self-training, eliminating the need for human annotation. These data are then leveraged to refine WA using a novel preference optimization method. Empirical results show that our framework significantly improves WA performance. Our WA outperforms both open-source state-of-the-art WAs and the closed-source GPT-4o by 3.9 and 7.1 points on average, respectively, across eight well-established writing-related test sets.[1]

1 Introduction

Large language models (LLMs) demonstrate strong proficiency in generating fluent and coherent text revisions, assisting users with various writing tasks (Achiam et al., 2023). However, general-purpose LLMs, such as GPT-4, often underperform compared to task-specific models (Cao et al., 2023b; Raheja et al., 2023). Consequently, research has focused on tailoring LLMs for writing-related tasks, such as grammatical error correction (GEC) (Bryant et al., 2023), text simplification (Baez and Saggion, 2023), and style transfer (Luo et al., 2023). Several studies (Raheja et al., 2023; Schick et al., 2023) have proposed writing assistants (WAs) that generate text suggestions based on user instructions.


CoEDIT

Prompt: Fix the grammatical errors in this sentence: The CEO, along with his advisors, were confident that the plan, which they spent months developing, would succeed.

Suggestion: The CEO, along with his advisors, was confident that the plan, which they spent months developing, would succeed.

PEER

Prompt: Fix the grammatical errors in this sentence: The CEO, along with his advisors, were confident that the plan, which they spent months developing, would succeed.

Suggestion: The CEO, along with his advisors, was confident that the plan, which they spent months developing, would succeed.

Rationale: Fix grammar errors.

Ours

Prompt: Fix the grammatical errors in this sentence: The CEO, along with his advisors, were confident that the plan, which they spent months developing, would succeed.

Suggestion: The CEO, along with his advisors, was confident that the plan, which they had spent months developing, would succeed.

Rationale:
The verb form "were" should be replaced by "was" because the subject "The CEO" is singular.
The verb form "spent" should be replaced by "had spent" because the action "developing the plan" happened before "being confident," requiring the past perfect tense.

Figure 1: Comparison of our WA and related systems' output: CoEDIT (Raheja et al., 2023) and PEER (Schick et al., 2023).

State-of-the-art (SOTA) WAs (Raheja et al., 2023; Zhang et al., 2023) are built by directly applying supervised fine-tuning (SFT) on labeled data. However, text editing inherently allows multiple valid revisions for a given input. Since SFT optimizes models to match a single reference revision, it fails to capture this flexibility. In contrast, task-specific evaluation metrics (*e.g.,* SARI, ROUGE) may account for such variation (Paulus et al., 2018). Consequently, relying solely on SFT may result in suboptimal performance. Furthermore, existing open-source WAs cannot provide rationales (*i.e.,* linguistic explanations) for their suggestions (Fig. 1), limiting their applicability.

To bridge this gap, we introduce the **Rationalize and Align** framework, which enhances WA with rationales and aligns WA towards suggestions with higher overall quality (*e.g.,* fluency, coherence). However, the absence of rationale and preference data presents a significant challenge, and acquiring human annotations is both time-consuming and expensive. To address this, we propose to automatically generate these data via *distillation* and *self-training*. Since existing WAs are not able to

[1]Our source code will be released upon paper publication.

provide rationales for their suggestions, we *distill* rationales (Fig. 2(a)) from GPT-4o into the labeled data, creating rationale-enhanced labeled datasets. These datasets are then used to develop a rationale-enhanced WA through SFT.

We then generate preference data to further enhance WA. However, text editing tasks lack appropriate reward models. Moreover, widely used evaluation metrics in writing tasks are reference-based metrics, restricting the generation of preference data to labeled datasets (*i.e.,* containing labeled suggestions for user instructions). However, such labeled data is scarce and costly to obtain. To address this, we propose a *self-training* alignment method that leverages unlabeled data to align WA. This method consists of two phases: (1) *reward modeling* (Fig. 2 (b)), which builds a reward model to assess suggestion quality, and (2) *preference optimization* (Fig. 2 (c)), which creates preference data and aligns WA towards higher-quality suggestions.

During *reward modeling*, we utilize labeled data (*i.e.,* data used for SFT) and evaluation metrics to generate preference data, which is then used to build the reward model. In the subsequent *preference optimization* phase, we leverage the constructed reward model and rationale-enhanced WA to create new preference data from unlabeled data for WA. This preference data, along with its corresponding reward assigned by the reward model, is then used to align WA via a novel margin-based preference optimization loss. This loss extends DPO (Rafailov et al., 2023) by replacing the reference model with the reward margin between preferred and non-preferred data, thereby capturing their quality differences more effectively.

Our experiments demonstrate that the Rationalize & Align framework significantly enhances our WA's performance. Across eight widely-used writing-related tasks, our WA significantly outperforms both open-source SOTA WA and GPT-4o, by 3.9 and 7.1 points on average, respectively. The contributions of our paper are as follows:

- We present the first open-source explainable WA that provides rationales for its suggestions.
- We introduce the Rationalize & Align framework to enhance WA. Our analysis demonstrates that each component contributes effectively.
- To the best of our knowledge, this is the first study to assess the impact of incorporating rationales into WA. Extensive experiments confirm that rationales enhance performance and increase confidence in the generated suggestions.

2 Related Work

Alignment. DPO (Rafailov et al., 2023) serves as a prominent approach in offline preference optimization. However, its reliance on a reference model and the absence of a reward model restrict its ability to sample preference pairs from the optimal policy (Meng et al., 2024). Recent advancements have aimed to eliminate the need for a reference model. CPO (Xu et al., 2024) entirely removes the reference model, whereas SimPO (Meng et al., 2024) replaces it with a length regularization term and integrates a pre-defined margin. Similar to SimPO, ODPO (Amini et al., 2024) integrates an offset into DPO while retaining the reference model. Unlike these approaches, our approach leverages the reward values generated by our reward model and replaces the reference model with a dynamically adjusted reward difference, tailored according to the quality of different preference data.

Enhancing LLMs with Rationale. Incorporating rationales into LLMs effectively enhances their faithfulness and performance across various tasks, including reasoning (Krishna et al., 2023; Wei et al., 2022), question answering (Kassner et al., 2023), and continual relation extraction (Xiong et al., 2023). Unlike previous studies, ours focuses on leveraging writing-related rationales to improve the performance of LLMs in writing tasks.

Writing Assistants. Previous WAs (Raheja et al., 2023; Zhang et al., 2023) provide suggestions based on user instructions yet lack sufficient explanatory support. In contrast, the PEER system (Schick et al., 2023), trained using the edit history of Wikipedia, offers high-level rationales. However, these rationales (Fig. 1) tend to be too abstract to effectively assist users in understanding or validating the suggestions. Our system overcomes these limitations by providing writing suggestions and detailed linguistic explanations for each edit.

3 Rationalize & Align

WAs aim to generate a proper writing suggestion s based on the user's instruction and the input text. Existing WAs (Raheja et al., 2023; Zhang et al., 2023) focus solely on providing writing suggestions, which are trained to learn the mapping from x to s, where x represents the combination of instruction and input text. Here, we want to enhance the WA by providing a rationale (or explanation) e for its s. Therefore, our goal is to learn a mapping

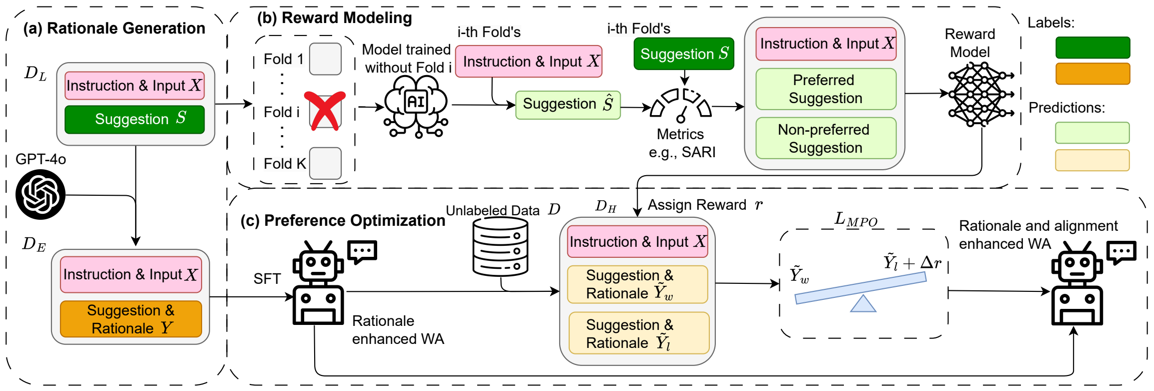


Figure 2: Our WA involves the following steps: (a) generating rationales for labeled data by distilling the output from GPT-4o, (b) constructing the reward model with $\mathcal{D}_L$ and evaluation metrics, and (c) building a rationale and alignment enhanced WA with self-generated preference data through the preference optimization process.

from x to y, where y represents the combination of suggestion s and rationale e. Fig. 2 shows the method to build our WA, comprising three key components: rationale generation, reward modeling, and preference optimization.

3.1 Rationale Generation

To enhance a WA with rationales in writing tasks, it is essential to provide linguistic explanations for its suggestions. However, current open-source WAs lack this capability, and no existing datasets have rationale annotations. To address this, we propose distilling rationales from the advanced closed-source model GPT-4o[2] into open-source models. Song et al. (2024) demonstrated that GPT-4 generates high-quality rationales (e) (92.2% accuracy in English) when given the input text (in), suggestion (s), and necessary edits ($edits$) that transform in into s, formulated as $e = \text{GPT-4}(in, s, edits)$.

Following their approach, we use GPT-4o to generate rationales using these three components. Specifically, we leverage the MaxMatch toolkit (Dahlmeier and Ng, 2012) to identify the edits required to transform the input into the suggestion. These *edits*, along with the *input* and *suggestion*, are incorporated into prompts (Fig. 3) that instruct GPT-4o to generate the *rationale*. This process yields a rationale-enhanced dataset $\mathcal{D}_E = \{x, s, e\}$ from the labeled dataset $\mathcal{D}_L = \{x, s\}$.

3.2 Self-Training Alignment

Aligning LLMs requires preference data, which is unavailable for writing-related tasks. To address this, we propose a self-training alignment framework that utilizes widely available unlabeled data to perform alignment. The framework comprises two key steps: *reward modeling* and *preference optimization*. In the *reward modeling* step, we use labeled data ($\mathcal{D}_L$) and task-specific evaluation metrics to generate preference data for training the reward model. During *preference optimization*, we first construct an initial WA on $\mathcal{D}_E$. We then utilize this initial WA and the reward model to generate high-quality preference data, which are subsequently used to further refine the WA.

You are given a pair of English sentences along with a list of atomic edits. For each edit, the first word identifies content in the source sentence that is less appropriate, while the second word suggests a better phrase in the target sentence. `[Task Instruction]` Please generate a succinct explanation for each edit using the following template:

The word X should be deleted/inserted/replaced by Y because ...

###Source sentence:
`[Input Text]`

###Target sentence:
`[Suggestion]`

###Edits:
`[Edit Content]`

###Explanation:

Figure 3: Prompt to generate rationale. `[Task Instruction]` for each task is shown in Table 12.

3.2.1 Reward Modeling

Our objective of reward modeling is to develop a robust reward model, r_ϕ, that can accurately evaluate the quality of any suggestion s given input x without relying on human-labeled suggestions. We achieve this by generating multiple pseudo-suggestions through sampling and then using automatic metrics to construct preference pairs, which are then used to build the reward model.

Specifically, we first partition the entire $\mathcal{D}_L$ into

[2] https://chat.chatbotapp.ai/?model=gpt-4o

$$\mathcal{L}_{DPO} = -\mathbb{E}_{(x,\tilde{y}_w,\tilde{y}_l)\sim D_H}\left[\log\sigma\left(\beta\log\frac{\theta_{\mathrm{W}}(\tilde{y}_w \mid x)}{\theta_{\mathrm{SFT}}(\tilde{y}_w \mid x)} - \beta\log\frac{\theta_{\mathrm{W}}(\tilde{y}_l \mid x)}{\theta_{\mathrm{SFT}}(\tilde{y}_l \mid x)}\right)\right] \tag{1a}$$

$$= -\mathbb{E}_{(x,\tilde{y}_w,\tilde{y}_l)\sim D_H}\{\log\sigma[\beta(\log\theta_{\mathrm{W}}(\tilde{y}_w \mid x) - \log\theta_{\mathrm{W}}(\tilde{y}_l \mid x) - \underbrace{(\log\theta_{\mathrm{SFT}}(\tilde{y}_w \mid x) - \log\theta_{\mathrm{SFT}}(\tilde{y}_l \mid x))}_{\text{margin}})]\} \tag{1b}$$

$$\mathcal{L}_M = -\mathbb{E}_{(x,\tilde{y}_w,\tilde{y}_l)\sim D_H}\{\log\sigma[\beta(\log\theta_{\mathrm{W}}(\tilde{y}_w \mid x) - \log\theta_{\mathrm{W}}(\tilde{y}_l \mid x) - \gamma\underbrace{(r_w - r_l)}_{\text{margin}})]\} \tag{1c}$$

K folds. For each fold, a model θ_k is trained on $K-1$ folds to learn the mapping from x to s. Subsequently, we use θ_k to generate N responses $\hat{s}$ for the reserved i-th fold through sampling. We then create preference pair $(\hat{s}_p, \hat{s}_n)$ for each x by randomly selecting two predictions from these responses and evaluating them against the true label s using automatic metrics. The response with the higher score is labeled as the preferred output $\hat{s}_p$, and the one with the lower score as the non-preferred output $\hat{s}_n$. Following Wang et al. (2024), we build the reward model with the objective function: $\mathcal{L}_r = -\log(\sigma(r_\phi(\hat{s}_p) - r_\phi(\hat{s}_n)))$, where σ is the sigmoid function.

3.2.2 Preference Optimization

Labeled data is often scarce, while unlabeled data is abundant. Therefore, we propose a preference optimization method that generates preference data from unlabeled data. The preference data is generated through sampling, utilizing both a reward model and an initial WA model. We build this initial WA model, θ_{W}, by fine-tuning LLM on $\mathcal{D}_E$ through SFT. To differentiate this initial WA model from our final WA model, we refer to this initial model as the SFT model and the final model as the WA model. This SFT model is optimized to generate the suggestion s followed by its rationale e, with the cross-entropy loss: $\mathcal{L}_{CE}(y) = -\log\theta_{\mathrm{W}}(y|x)$.

Preference Data Generation. Utilizing the SFT model, we sample multiple responses and employ the reward model to identify the preferred and non-preferred responses, constructing preference pairs from unlabeled data. Specifically, for each input x, we sample M responses with θ_{W}, where each response contains a writing suggestion $\tilde{s}$ and its corresponding rationale. The reward model then assigns a score to each $\tilde{s}$, designating the highest-scoring suggestion as the winning suggestion $\tilde{s}_w$ and the lowest-scoring as the losing suggestion $\tilde{s}_l$. Each suggestion is paired with its rationale to form the winning response $\tilde{y}_w$ and the losing response $\tilde{y}_l$, resulting in a preference pair $(\tilde{y}_w, \tilde{y}_l)$ for each x. The reward for $\tilde{s}_w$ and $\tilde{s}_l$ are denoted as r_w and r_l, respectively.

However, suggestions generated through sampling exhibit varying quality (Beigi et al., 2024). To ensure that θ_W learns only from high-quality suggestions, we filter out lower-quality ones based on the reward r_w. Specifically, we rank the preference pairs by r_w and retain only the top 20% of the preference pairs with the highest reward. We denote this subset of high-quality preference pairs and their corresponding inputs as $\mathcal{D}_H = \{x, \tilde{y}_w, \tilde{y}_l\}$. Since no established methods exist for evaluating rationale quality, we rank suggestions and rationales together based on the reward value of the suggestions, hypothesizing that higher-quality suggestions correlate with better rationales.

Optimization. Eq. (1a) and Eq. (1b) present the DPO (Rafailov et al., 2023) loss. As shown in Eq. (1b), DPO ensures that the log probability of the winning response, $\tilde{y}_w$, exceeds that of the losing response, $\tilde{y}_l$, by a margin equal to their log probability difference under the SFT model, θ_{SFT}. Within our framework, since the reward model assesses suggestion quality, the quality gap between the winning and losing responses can be captured by the reward difference, $r_w - r_l$. With an effective r_ϕ (Fig.4), we hypothesize that the reward difference more accurately captures this margin[3], motivating us to replace the log probability difference computed by θ_{SFT} with the reward difference. Consequently, we define our margin loss, $\mathcal{L}_M$, in Eq. (1c), where γ controls the weight assigned to the reward difference. To better align WA with the distribution of winning responses (Xu et al., 2024), we integrate $\mathcal{L}_{CE}$, computed for $\tilde{y}_w$, with $\mathcal{L}_M$ and propose the margin-based preference optimization loss, $\mathcal{L}_{MPO}$. We then optimize θ_{W} with $\mathcal{L}_{MPO}$:

$$\mathcal{L}_{MPO} = \lambda\mathcal{L}_M + \mathcal{L}_{CE}(\tilde{y}_w) \tag{2}$$

where λ is a hyper-parameter, controlling the weight assigned to $\mathcal{L}_M$. SimPO (Meng et al., 2024) also incorporates a margin, but our approach differs in motivation. Unlike SimPO, which uses a

[3]Fig. 5 demonstrates the importance of this margin.

	System	CoN	ITR-F	ITR-L	ITR-O	STS	WNC	AST	GYAFC		ALL
									EM	FR	
a)	Llama-3.3-70B-Instruct	55.6	46.5	31.4	31.0	34.6	31.8	46.4	59.5 / 98.1	56.2 / 98.4	43.7
	ChatGPT	53.3	50.9	31.5	31.0	39.9	36.3	47.0	57.7 / 99.6	60.4 / 99.5	45.3
	GPT-4	59.9	51.6	32.6	32.3	42.2	40.8	46.3	60.2 / 99.6	62.4 / 99.5	47.6
	GPT-4o	59.4	51.1	32.4	32.4	42.4	41.1	47.4	62.8 / 99.2	63.7 / 99.1	48.1
b)	Llama-3.3-70B-Instruct (R)	58.4	49.4	35.0	31.8	37.7	41.2	44.7	63.6 / 97.8	65.3 / 98.2	47.5
	ChatGPT (R)	56.1	51.1	30.3	28.7	40.6	36.6	45.0	63.1 / 98.7	63.5 / 98.9	46.1
	GPT-4 (R)	60.4	50.1	33.3	32.8	41.2	40.7	**47.6**	63.2 / 98.8	63.5 / 99.1	48.1
	GPT-4o (R)	60.8	51.4	32.6	32.2	43.3	40.9	46.1	64.4 / 98.4	64.6 / 98.6	48.5
c)	PEER-EDIT-11B	N.A.	52.1	32.5	32.7	28.2	54.5	29.5	N.A.	N.A.	N.A.
	Writing-Alpaca (7B)	55.9	**52.8**	**39.4**	37.1	44.6	64.4	44.7	N.A.	N.A.	N.A.
	CoEDIT-xxl (11B)	57.1*	51.6	31.8	31.5	42.9*	**71.0**	41.7	66.0 / 98.7*	68.7 / 97.9*	51.7
	Ours based on Flan-T5-xxl (11B) (RealEdit-11B)										
d)	SFT model	58.3	50.9	33.6	32.2	43.0	70.8	41.4	69.2 / 97.3	70.5 / 97.1	52.1
	+ Self-Training Alignment	61.4	49.3	32.8	34.7	47.0	68.9	41.1	75.3 / 96.8	78.0 / 96.3	54.3
e)	SFT model (R)	61.8	51.3	30.2	36.1	46.6	69.0	43.1	73.4 / 97.4	76.2 / 97.1	54.1
	+ Self-Training Alignment (R)	62.1	52.5	33.5	38.6	44.7	70.2	42.8	75.6 / 97.1	77.4 / 96.9	55.2
	Ours based on Llama 3.1 8B (RealEdit-8B)										
f)	SFT model	61.7	50.5	31.6	35.6	43.3	66.4	42.0	75.3 / 97.9	75.5 / 96.8	53.5
	+ Self-Training Alignment	62.5	48.7	31.9	**40.3**	**47.1**	64.6	45.2	**78.0** / 96.8	78.0 / 95.9	55.1
g)	SFT model (R)	62.7	51.1	34.0	37.5	45.1	65.8	41.3	75.9 / 98.4	76.4 / 97.5	54.4
	+ Self-Training Alignment (R)	**65.5**	48.7	35.4	37.6	46.5	65.9	45.2	77.3 / 97.9	**78.2** / 97.3	**55.6**

Table 1: Performance on writing-related tasks. All results are shown in %. *: Results reproduced using the official checkpoint and scripts released by Raheja et al. (2023), due to different evaluation metrics or test sets not previously evaluated. For the GYAFC test sets, the first score is BLEU and the second is accuracy. Following Raheja et al. (2023); Zhang et al. (2023), we show the averaged result under the ALL column, and we only consider the BLEU score for the GYFAC test sets when taking the average. **a)**: zero-shot performance of LLMs. **b)**: zero-shot performance of LLMs when also prompted to generate rationales (or explanations) for their writing suggestions. **c)**: SOTA WAs. **d) & f)**: RealEdit trained without rationale. **e) & g)**: RealEdit trained with rationale.

fixed margin as a hyper-parameter, our margin is determined by the reward difference, allowing it to adapt to varying input qualities. Furthermore, SimPO is not as effective as ours (Fig. 5).

4 Experimental Results

4.1 Data and Model Configuration

Data. Following previous works (Raheja et al., 2023; Zhang et al., 2023), we evaluate our system's performance in generating suggestions across eight writing-related tasks. These tasks closely align with EDITEVAL (Dwivedi-Yu et al., 2022): GEC, fluency, clarity, coherence, paraphrasing, neutralization, simplification, and formality style transfer (FST). For GEC, we use the CoNLL-2014 (CoN) test set (Ng et al., 2014). Fluency, clarity, and coherence are assessed using the ITERATER test sets (Du et al., 2022), corresponding to ITR-F, ITR-L, and ITR-O, respectively. Paraphrasing is evaluated with the STSB (STS) test set (Dwivedi-Yu et al., 2022), while neutralization is assessed using the WNC test set (Pryzant et al., 2020). Simplification is evaluated using the ASSET (AST) test set (Alva-Manchego et al., 2020). FST is evaluated on the GYAFC test set (Rao and Tetreault, 2018), covering both the Entertainment & Music (EM) and Family & Relationships (FR) domains.

Metrics. Following prior research (Raheja et al., 2023; Schick et al., 2023; Zhang et al., 2023), we primarily utilize the SARI metric (Xu et al., 2016) for evaluation, except for the GEC and FST tasks. For GEC, we use the MaxMatch scorer (Dahlmeier and Ng, 2012), as done in previous studies (Cao et al., 2023a). For FST, we follow the evaluation setting from previous work (Rao and Tetreault, 2018; Tang et al., 2022), using both the BLEU metric (Papineni et al., 2002) and a formality classification model to assess style transfer accuracy.

Implementation Details. Following Raheja et al. (2023), we randomly select 80k sentence pairs (source and target sentences) from the training sets of eight tasks to build the SFT and the reward model. During alignment, we randomly sample another 400k sentences from the entire training set, using only the sentences from the source side. The 80k sentence pairs used for building the SFT model were excluded from this sample. These 400k sentences are not parallel and do not include their human-annotated target sentences. We name the WA built with our **R**ationaliz**e** & **Al**ign framework as **RealEdit**. Both the reward model and WA are based on Llama-3.1-8B (Dubey et al., 2024), fine-tuned using the LoRA (Hu et al., 2022). We report the average scores of three runs and use a one-tailed sign test with bootstrap resampling for statistical significance tests (Cao et al., 2021). More information is provided in Appendix B.

Baselines. We compare RealEdit against both open-source and closed-source SOTA LLMs. Specifically, we compare against previous SOTA LLMs in the writing domain, such as CoEDIT-xxl, PEER-EDIT-11B, and Writing-Alpaca (Zhang et al., 2023). We also compare against strong open-sourced and close-sourced LLMs in zero-shot settings, including Llama-3.3-70B-Instruct, ChatGPT, GPT-4, and GPT-4o.

4.2 Main Results

Table 1 shows the performance of RealEdit-8B, and SOTA LLMs across eight writing-related tasks. RealEdit-8B (Ours) significantly outperforms both the open-source SOTA WA, CoEDIT-xxl, and the closed-source LLM, GPT-4o. Specifically, RealEdit-8B surpasses CoEDIT-xxl by an average of 3.4 points and GPT-4o by an average of 6.6 points when trained without rationales. Incorporating rationales further enhances performance, with an average increase of 3.9 points over CoEDIT-xxl and 7.1 points over GPT-4o.

Our statistical significance tests confirm that our self-training alignment method significantly enhances the performance of RealEdit-8B, both with and without rationales. Specifically, alignment increases RealEdit-8B's average performance by 1.6 points without rationales and by 1.2 points with rationales. Moreover, adding rationales to RealEdit-8B results in a substantial improvement: with GPT-4o's rationales, RealEdit-8B's performance improves by an average of 0.9 points before alignment and by 0.5 points after alignment.

To ensure a fair comparison with CoEDIT-xxl, we construct a new WA model, RealEdit-11B, by replacing Llama-3.1-8B with the older Flan-T5-xxl model (Chung et al., 2022) for both the WA and the reward model. Despite using an older architecture, RealEdit-11B outperforms CoEDIT-xxl by an average of 3.5 points, demonstrating the effectiveness of our Rationalize & Align framework. More performance are shown in Appendix D.

5 Analysis

5.1 Self-Training

Reward Model. Following previous work (Tunstall et al., 2024; Cai et al., 2024), we evaluate our reward model on the downstream task. In our framework, the reward model is responsible for selecting high-quality suggestions. Prior research (Dahlmeier and Ng, 2012; Xu et al., 2016) has shown that evaluation metrics align well with human preferences on their respective tasks. Thus, we assess the quality of $\tilde{s}_w$ and $\tilde{s}_l$ using these metrics.

Fig. 4 (Left) shows that 88% of $\tilde{s}_w$ outperform $\tilde{s}_l$ on task-specific metrics, demonstrating our reward model is effective in selecting high-quality suggestions. Additionally, Fig. 4 (Right) reveals a strong correlation between our model's scores and these metrics, with a Pearson correlation coefficient of 0.63 (Cohen et al., 2009). Since these metrics align with human preferences, this suggests our reward model does as well. Appendix C.1 shows more detail about the experimental setup and metric descriptions.

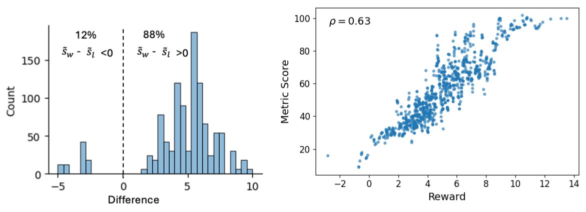


Figure 4: **Left**: The distribution of $metric(\tilde{s}_w) - metric(\tilde{s}_l)$, where $metric$ represents the task-specific evaluation metric. **Right**: Pearson correlation between our reward model and automatic evaluation metric.

Quality of the Winning Response. As described in §4.1, RealEdit-8B was developed using 80k labeled examples to build the SFT model, which was subsequently fine-tuned with 78k preference data, $\mathcal{D}_H$. To assess the effectiveness of the winning suggestion, we compared fine-tuning the SFT model using the winning suggestion ($\tilde{s}_w$) against using the labeled data (s). Notably, this experiment was conducted without incorporating any rationale into RealEdit-8B.

The results presented in Fig. 6 demonstrate that utilizing $\tilde{s}_w$ leads to a greater improvement in performance. Statistical tests confirm that using $\tilde{s}_w$ consistently outperforms using s across all test sets. This suggests that using the winning response is more helpful for the WA to achieve better alignment. The improvement likely stems from the reward model's selection of the winning suggestions, as it is optimized to favor suggestions with higher overall quality (*e.g.,* fluency and coherence) as measured by task-specific metrics.

Ablation Study. We analyze the effectiveness of individual components within our loss function $\mathcal{L}_{MPO}$. Fig. 5 (Left) indicates that when the ra-

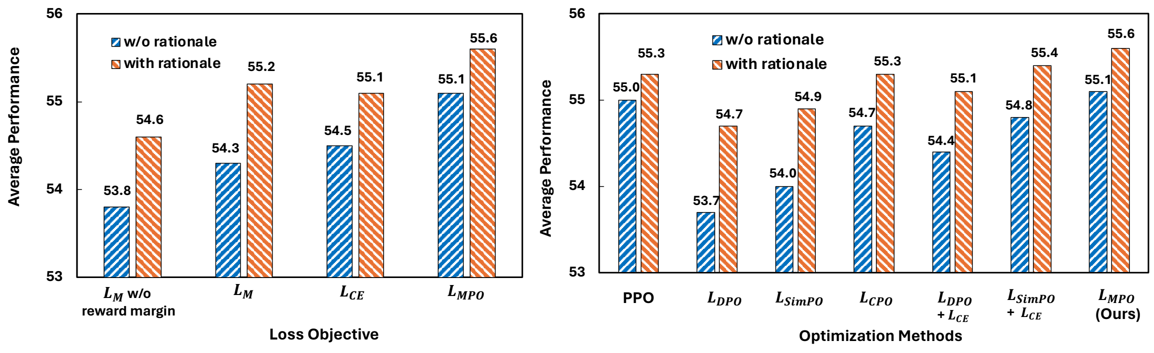


Figure 5: **Left**: Ablation study evaluating the significance of individual components in the loss function (Eq. (2)). The bars labeled 'L_M w/o reward margin' indicate setting γ to 0 in Eq. (1c). **Right**: Performance comparison against other related preference optimization methods.

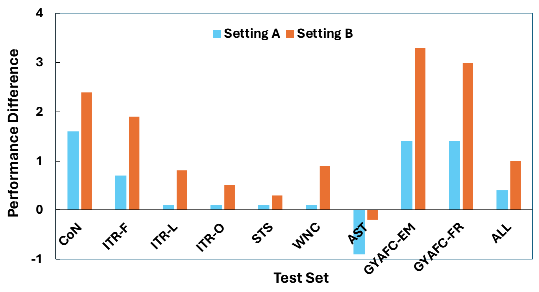


Figure 6: The performance difference (in %) between the WA (obtained under Setting A and B) and the SFT model. **Setting A**: Fine-tune the SFT model with an additional 78k labeled data (s). **Setting B**: Fine-tune the SFT model with an additional 78k $\tilde{s}_w$ from $\mathcal{D}_H$.

tionale is not incorporated, both '$\mathcal{L}_M$ w/o reward margin' and '$\mathcal{L}_M$' enhance the RealEdit-8B's performance. Statistical significance tests confirm that incorporating the reward margin into '$\mathcal{L}_M$' substantially enhances performance, highlighting the critical role of our reward margin. Moreover, employing $\mathcal{L}_M$ or $\mathcal{L}_{CE}$ alone significantly outperforms the SFT model, demonstrating the effectiveness of $\mathcal{L}_M$ and the efficacy of our framework in selecting high-quality samples. Ultimately, $\mathcal{L}_{MPO}$, which combines $\mathcal{L}_M$ and $\mathcal{L}_{CE}$, yields the largest improvement. Similar patterns have been observed when the rationales are incorporated.

Preference Optimization. Our objective function, $\mathcal{L}_{MPO}$, shares similarities with CPO and SimPO. We compare optimizing the SFT model using $\mathcal{L}_{MPO}$ against CPO, SimPO, and PPO (Schulman et al., 2017), where PPO uses our reward model for reward computation. Fig. 5 (Right) shows that $\mathcal{L}_{MPO}$ outperforms CPO, DPO, SimPO, and PPO. Furthermore, incorporating $\mathcal{L}_{CE}$ into DPO and SimPO improves performance, highlighting its importance in reducing deviations from the preferred data distribution. Ours is also more memory-efficient than PPO, requiring only to store θ_{W}, whereas PPO additionally stores θ_{SFT} and r_ϕ. Appendix B.4 shows experimental details.

5.2 Rationale

We use RealEdit-8B without self-training alignment (*i.e.,* only enhanced with rationale) to analyze the impact of rationale on WA.

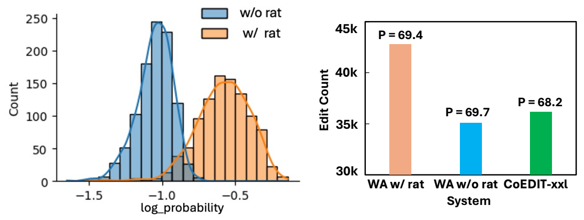


Figure 7: **Left**: The probability to generate the target sentence by WA trained with (w/ rat) and without rationale (w/o rat). **Right**: Number of edits proposed by different systems, with the precision of the edits displayed above each bar (P=*).

Rationale Effect. We compare the probability of WA trained with and without rationales to produce the target (human-annotated) sentence. For this experiment, we randomly selected 1k sentences from the validation set. Fig. 7 (Left) demonstrates that the WA model trained with rationales has a significantly higher probability of generating the target sentence compared to the model trained without rationales. This suggests that the WA model, when trained with rationales, is more confident and proficient in generating accurate writing suggestions. Additionally, Fig. 7 (Right) reveals that the WA model trained with rationales proposes more edits (less conservative) while maintaining relatively high precision. Implementation details are shown in Appendix C.3.

Other Rationales. Fig. 9 (Left) illustrates the

Task		Fluency	Simplification
Instruction		Make the text more fluent and grammatical by initially providing a response, followed by an explanation.	Simplify the text by first providing a response, followed by an explanation.
Input		While I was walking through the park, there was a bird who were singing beautifully, but none of my friends who was with me noticed.	GPT-4o, known for its advanced natural language processing capabilities, is widely for text understanding.
Target		While I was walking through the park, there was a bird that was singing beautifully, but none of my friends who were with me noticed.	GPT-4o is widely used for text understanding.
GPT-4o		While I was walking through the park, there was a bird who was singing beautifully, but none of my friends who was with me noticed.	GPT-4 is widely used for text understanding.
WA w/o rat	suggestion	While I was walking through the park, there was a bird that were singing beautifully, but none of my friends who was with me noticed.	GPT-4o, known for its advanced natural language processing capabilities, is widely used for text understanding.
WA w/ rat	suggestion	While I was walking through the park, there was a bird that was singing beautifully, but none of my friends who were with me noticed.	GPT-4o is widely used for text understanding.
	rationale	The word "who" should be replaced by "that" because "that" is more appropriate for referring to animals. The word "were" should be replaced by "was" because "a bird" is singular. The word "was" should be replaced by "were" because "none of my friends" is plural.	The phrase "known for its advanced natural language processing capabilities" is deleted because it is descriptive but not essential. Removing it simplifies the sentence by focusing on the primary message.

Figure 8: Example output generated by GPT-4o, and the WA trained with (w/ rat) and without rationale (w/o rat). Wrong or unnecessary words in the GPT-4o and WA w/o rat's suggestions are highlighted in yellow.

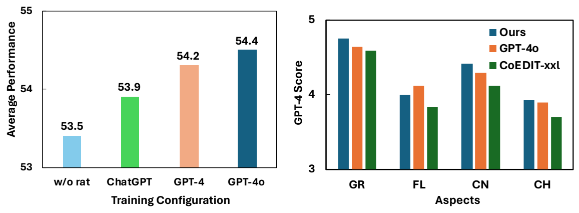


Figure 9: **Left**: WA's performance when trained without rationales (w/o rat) and with rationales provided by ChatGPT, GPT-4, and GPT-4o. **Right**: Evaluation results by GPT-4 on outputs generated by RealEdit-8B (Ours), GPT-4o, and CoEDIT-xxl.

performance of WA under different rationales. Statistical significance tests indicate that rationales generated by ChatGPT significantly enhance performance compared to no rationales. Moreover, rationales from GPT-4 and GPT-4o yield even greater improvements, which are statistically significant compared to those from ChatGPT. Song et al. (2024) demonstrates that GPT-4 generates higher-quality rationales than ChatGPT, indicating that higher-quality rationales can more effectively enhance the suggestion performance.

5.3 LLM Based Evaluations

Text editing is inherently subjective, and traditional automatic metrics may not accurately assess text quality. While human evaluations provide valuable insights, they are susceptible to bias and lack reproducibility. Various researches (Liu et al., 2023; Kim and Kim, 2024) demonstrate that GPT-4 exhibits a strong correlation with human assessments of text quality across multiple dimensions, including grammaticality (GR), fluency (FL), coherence (CH), and consistency (CN). Following Kim and Kim (2024), we employ GPT-4 to score 1k outputs generated by RealEdit-8B, CoEDIT-xxl, and GPT-4o on a 1–5 scale rubric. Fig. 9 (Right) shows that RealEdit-8B outperforms both GPT-4o and CoEDIT-xxl in grammaticality, consistency, and coherence, while slightly underperforming GPT-4o in fluency. Additionally, leveraging the criteria proposed by Song et al. (2024), we assess the accuracy of rationales generated by RealEdit-8B using GPT-4, achieving an accuracy of 87.3%. Appendix C.2 shows the detailed experimental setup.

5.4 Case Studies

Fig. 8 qualitatively compares the suggestions generated by our WA (*i.e.,* RealEdit-8B), GPT-4o, and WA trained without rationale. When trained without rationale, WA often fails to address complex errors or remove superfluous words effectively. Similarly, GPT-4o occasionally mishandles complex errors and removes essential tokens, which significantly alters the meaning of the original sentence. The superior performance of RealEdit comes from two key factors: (1) the incorporation of rationales, which enhances both error identification and confidence in producing high-quality suggestions, and (2) it is optimized toward human preferences in writing, resulting in more refined suggestions. We provide further analysis in Appendix A.

6 Conclusion

In this work, we introduce a novel framework, Rationalize & Align, to enhance the performance of LLMs in writing-related tasks. Its core idea lies in distilling rationales from GPT-4o into the WA system and further improving it through self-training alignment, which includes reward modeling and preference optimization. Our experiments demonstrate that each component contributes to superior writing suggestions. Our WA surpasses both the SOTA WA system and GPT-4o in suggestion quality. Moreover, we built the first open-source WA capable of generating rationales for its suggestions.

7 Limitations

We conducted extensive experiments on various writing-related tasks (in English), demonstrating the effectiveness of our Rationalize & Align framework. While we believe this approach to be task-agnostic, its application to other domains, such as machine translation, remains unexplored and warrants future investigation. Additionally, our self-training alignment method involves generating multiple sampled responses to create the preference data, which can demand substantial GPU resources when scaling to larger datasets. Future research could explore more efficient methods to generate preference data. We believe our work does not bring any direct harm to individuals or society.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Fernando Alva-Manchego, Louis Martin, Antoine Bordes, Carolina Scarton, Benoît Sagot, and Lucia Specia. 2020. ASSET: A dataset for tuning and evaluation of sentence simplification models with multiple rewriting transformations. In *Proc. of ACL*, pages 4668–4679.

Afra Amini, Tim Vieira, and Ryan Cotterell. 2024. Direct preference optimization with an offset.

Anthony Baez and Horacio Saggion. 2023. LSLlama: Fine-tuned LLaMA for lexical simplification. In *Proceedings of the Second Workshop on Text Simplification, Accessibility and Readability*, pages 102–108.

Mohammad Beigi, Sijia Wang, Ying Shen, Zihao Lin, Adithya Kulkarni, Jianfeng He, Feng Chen, Ming Jin, Jin-Hee Cho, Dawei Zhou, et al. 2024. Rethinking the uncertainty: A critical review and analysis in the era of large language models. *arXiv preprint arXiv:2410.20199*.

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*.

Zheng Cai, Maosong Cao, Haojiong Chen, Kai Chen, Keyu Chen, Xin Chen, Xun Chen, Zehui Chen, Zhi Chen, Pei Chu, et al. 2024. Internlm2 technical report. *arXiv preprint arXiv:2403.17297*.

Hannan Cao, Wenmian Yang, and Hwee Tou Ng. 2021. Grammatical error correction with contrastive learning in low error density domains. In *Findings of EMNLP*, pages 4867–4874.

Hannan Cao, Wenmian Yang, and Hwee Tou Ng. 2023a. Mitigating exposure bias in grammatical error correction with data augmentation and reweighting. In *Proc. of EACL*, pages 2123–2135.

Hannan Cao, Liping Yuan, Yuchen Zhang, and Hwee Tou Ng. 2023b. Unsupervised grammatical error correction rivaling supervised methods. In *Proc. of EMNLP*, pages 3072–3088.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022. Scaling instruction-finetuned language models. *Preprint*, arXiv:2210.11416.

Israel Cohen, Yiteng Huang, Jingdong Chen, Jacob Benesty, Jacob Benesty, Jingdong Chen, Yiteng Huang, and Israel Cohen. 2009. Pearson correlation coefficient. *Noise reduction in speech processing*, pages 1–4.

Daniel Dahlmeier and Hwee Tou Ng. 2012. Better evaluation for grammatical error correction. In *Proc. of NAACL*, pages 568–572.

Wanyu Du, Vipul Raheja, Dhruv Kumar, Zae Myung Kim, Melissa Lopez, and Dongyeop Kang. 2022. Understanding iterative revision from human-written text. In *Proc. of ACL*, pages 3573–3590.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Jane Dwivedi-Yu, Timo Schick, Zhengbao Jiang, Maria Lomeli, Patrick Lewis, Gautier Izacard, Edouard Grave, Sebastian Riedel, and Fabio Petroni. 2022. Editeval: An instruction-based benchmark for text improvements. *arXiv preprint arXiv:2209.13331*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *Proc. of ICLR*.

Nora Kassner, Oyvind Tafjord, Ashish Sabharwal, Kyle Richardson, Hinrich Schuetze, and Peter Clark. 2023. Language models with rationality. In *Proc. of EMNLP*, pages 14190–14201.

Seungyoon Kim and Seungone Kim. 2024. Can language models evaluate human written text? case study on korean student writing for education. *Preprint*, arXiv:2407.17022.

Satyapriya Krishna, Jiaqi Ma, Dylan Z Slack, Asma Ghandeharioun, Sameer Singh, and Himabindu Lakkaraju. 2023. Post hoc explanations of language models can improve language models. In *Proc. of NeurIPS*.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proc. of the ACM SIGOPS*.

Ao Liu, An Wang, and Naoaki Okazaki. 2022. Semi-supervised formality style transfer with consistency training. In *Proc. of ACL*, pages 4689–4701.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. 2023. G-eval: NLG evaluation using gpt-4 with better human alignment. In *Proc. of EMNLP*, pages 2511–2522, Singapore.

Guoqing Luo, Yu Han, Lili Mou, and Mauajama Firdaus. 2023. Prompt-based editing for text style transfer. In *Findings of EMNLP*, pages 5740–5750.

Yu Meng, Mengzhou Xia, and Danqi Chen. 2024. SimPO: Simple preference optimization with a reference-free reward. *arXiv preprint arXiv:2405.14734*.

Masato Mita, Keisuke Sakaguchi, Masato Hagiwara, Tomoya Mizumoto, Jun Suzuki, and Kentaro Inui. 2024. Towards automated document revision: Grammatical error correction, fluency edits, and beyond. In *Proc. of BEA*.

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 *Proc. of CoNLL*, pages 1–14.

Kostiantyn Omelianchuk, Andrii Liubonko, Oleksandr Skurzhanskyi, Artem Chernodub, Oleksandr Korniienko, and Igor Samokhin. 2024. Pillars of grammatical error correction: Comprehensive inspection of contemporary approaches in the era of large language models. In *Proc. of BEA*, pages 17–33.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proc. of ACL*, pages 311–318.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. In *Proc. of ICLR*.

Reid Pryzant, Richard Diehl Martinez, Nathan Dass, Sadao Kurohashi, Dan Jurafsky, and Diyi Yang. 2020. Automatically neutralizing subjective bias in text. *Proc. of AAAI*, pages 480–489.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. In *Proc. of NeurIPS*.

Vipul Raheja, Dhruv Kumar, Ryan Koo, and Dongyeop Kang. 2023. CoEdIT: Text editing by task-specific instruction tuning. In *Findings of EMNLP*, pages 5274–5291.

Sudha Rao and Joel Tetreault. 2018. Dear sir or madam, may I introduce the GYAFC dataset: Corpus, benchmarks and metrics for formality style transfer. In *Proc. of NAACL*, pages 129–140.

Timo Schick, Jane A. Yu, Zhengbao Jiang, Fabio Petroni, Patrick Lewis, Gautier Izacard, Qingfei You, Christoforos Nalmpantis, Edouard Grave, and Sebastian Riedel. 2023. PEER: A collaborative language model. In *Proc. of ICLR*.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Yixiao Song, Kalpesh Krishna, Rajesh Bhatt, Kevin Gimpel, and Mohit Iyyer. 2024. GEE! grammar error explanation with large language models. In *Findings of ACL*, pages 754–781.

Tianyi Tang, Junyi Li, Zhipeng Chen, Yiwen Hu, Zhuohao Yu, Wenxun Dai, Wayne Xin Zhao, Jian-yun Nie, and Ji-rong Wen. 2022. TextBox 2.0: A text generation library with pre-trained language models. In *Proc. of EMNLP*, pages 435–444.

Lewis Tunstall, Edward Emanuel Beeching, Nathan Lambert, Nazneen Rajani, Kashif Rasul, Younes Belkada, Shengyi Huang, Leandro Von Werra, Clémentine Fourrier, Nathan Habib, Nathan Sarrazin, Omar Sanseviero, Alexander M Rush, and Thomas Wolf. 2024. Zephyr: Direct distillation of LM alignment. In *COLM*.

Binghai Wang, Rui Zheng, Lu Chen, Yan Liu, Shihan Dou, Caishuang Huang, Wei Shen, Senjie Jin, Enyu Zhou, Chenyu Shi, Songyang Gao, Nuo Xu, Yuhao Zhou, Xiaoran Fan, Zhiheng Xi, Jun Zhao, Xiao Wang, Tao Ji, Hang Yan, Lixing Shen, Zhan Chen, Tao Gui, Qi Zhang, Xipeng Qiu, Xuanjing Huang, Zuxuan Wu, and Yu-Gang Jiang. 2024. Secrets of rlhf in large language models part ii: Reward modeling. *Preprint*, arXiv:2401.06080.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Proc. of NeurIPS*, pages 24824–24837.

Weimin Xiong, Yifan Song, Peiyi Wang, and Sujian Li. 2023. Rationale-enhanced language models are better continual relation learners. In *Proc. of EMNLP*, pages 15489–15497.

Haoran Xu, Amr Sharaf, Yunmo Chen, Weiting Tan, Lingfeng Shen, Benjamin Van Durme, Kenton Murray, and Young Jin Kim. 2024. Contrastive preference optimization: Pushing the boundaries of LLM performance in machine translation. In *Proc. of ICML*.

Wei Xu, Courtney Napoles, Ellie Pavlick, Quanze Chen, and Chris Callison-Burch. 2016. Optimizing statistical machine translation for text simplification. *TACL*, pages 401–415.

Yue Zhang, Leyang Cui, Deng Cai, Xinting Huang, Tao Fang, and Wei Bi. 2023. Multi-task instruction tuning of llama for specific scenarios: A preliminary study on writing assistance. *arXiv preprint arXiv:2305.13225*.

Appendix

A Further Analysis

A.1 Adaptability

Setting. We investigate whether our WA can be extended to other text editing domains, LaTeX text editing, a common but challenging academic writing scenario. LaTeX editing requires careful handling of keywords, as errors can cause compilation failures. We compare our WA against the best model, CoEDIT-xxl, in Table 1. To ensure a fair comparison, we use our WA based on Flan-T5-xxl (RealEdit-11B). Both LLMs revise the LaTeX text to improve the fluency, and we evaluate them based on the average error rate and fluency for the post-compilation text. The fluency is measured using perplexity (PPL) computed with GPT-2 Large (Radford et al., 2019).

TETRA dataset (Mita et al., 2024) comprises 64 research papers written by non-native speakers. Among them, we identified nine papers with available LaTeX source code. Table 3 presents the URLs for these papers. From these papers, we extracted 100 sentences containing at least one LaTeX keyword for evaluation.

Result. Table 2 compares the fluency of revised text across RealEdit-11B and CoEDIT-xxl. RealEdit-11B produces more fluent revisions, while both CoEDIT-xxl and RealEdit-11B exhibit relatively high error rates. This may be due to the lack of fine-tuning on LaTeX-specific data, as LaTeX revision differs significantly from plain text revision, limiting their performance.

	AvgCE.↓	**PPL↓**
RealEdit-11B	**0.54**	**32.4**
CoEDIT-xxl	0.56	35.2
After fine-tuning		
RealEdit-11B	**0.24**	**28.4**
CoEDIT-xxl	0.27	31.3

Table 2: Revised text generated by ReAlEdit-11B and CoEDIT-xxl. **AvgCE.** refers to the average compilation error, while **PPL** indicates the perplexity of the revised text obtained after LaTeX compilation.

ID	URL
1	https://arxiv.org/abs/1805.11267
2	https://arxiv.org/abs/1603.03116
3	https://arxiv.org/abs/1705.00823
4	https://arxiv.org/abs/1704.04859
5	https://arxiv.org/abs/1606.01323
6	https://arxiv.org/abs/1810.05104
7	https://arxiv.org/abs/1804.10959
8	https://arxiv.org/abs/1705.00316
9	https://arxiv.org/abs/1805.07043

Table 3: URLs of papers from which we obtained the LaTeX source.

To mitigate this issue, we extract LaTeX sentences, refine them using GPT-4o for improved fluency, and filter out revisions with compilation errors. We generate 10k data using this process, and we use it to further fine-tune RealEdit-11B and CoEDIT-xxl. After fine-tuning, we observe that the error rates of RealEdit-11B have been significantly reduced, demonstrating that RealEdit-11B can be effectively adapted to other domains with a small amount of in-domain data.

A.2 Relative Position of Suggestion and Rationale

Given a user instruction, our WA (RealEdit-8B) is designed to generate a writing suggestion s first followed by its rationale e. We want to analyze how our WA will perform when it is asked to generate the rationale e first then followed by the suggestion s. Fig. 10 shows that generating the rationale first did not improve performance, whereas generating the suggestion first followed by the rationale yielded significant performance gains.

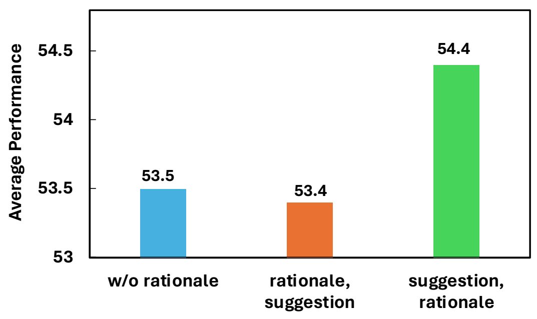


Figure 10: The performance (in %) of WA (RealEdit-8B) trained under different settings: 1) trained without (w/o) rationale (1st bar), 2) trained to generate rationale first followed by the suggestion (2nd bar), 3) trained to generate suggestion first followed by the rationale (3rd bar).

B Experimental Details

B.1 Training Data Configuration.

To balance between each task, we sample 10k labeled sentences per task when building the SFT model, and sample around 40k unlabeled sentences per task when building the final WA model. Table 4 lists the detailed information about the training set and test set.

Task	Train	Test	Metric	Abbrev.	# Sent
GEC	W&I + LOCNESS - Train	CoNLL-2014	M2	CoN	1,312
Fluency	ITERATER-V2-Train	ITERATER-fluency	SARI	ITR-F	88
Clarity	ITERATER-V2-Train	ITERATER-clarity	SARI	ITR-L	185
Coherence	ITERATER-V2-Train	ITERATER-coherence	SARI	ITR-C	35
Paraphrase	Parabank V2	STSB	SARI	STSB	97
Neutralization	WNC - Train	WNC	SARI	WNC	1,000
Simplification	TurkCorpus, NEWSELA, WikiLarge, WikiAuto, Parabank V2	ASSET	SARI	AST	359
FST	GYAFC-EM-Train	GYAFC-EM	BLEU, ACC	GYAFC-EM	1,416
FST	GYAFC-FR-Train	GYAFC-FR	BLEU, ACC	GYAFC-FR	1,332

Table 4: The tasks, training sets, test sets, metrics used, abbreviations used, and numbers of sentences (# Sent) in the various test sets in our evaluation benchmark. ACC represents the accuracy evaluation metric.

B.2 Validation Data Configuration.

Table 5 lists the validation data we have used throughout our experiments. We use the official validation sets for the GEC, fluency, clarity, coherence, paraphrase, neutralization, simplification, and FST tasks. The average performance across these validation sets is used to guide checkpoint and hyper-parameter selection.

B.3 Implementation Details.

We implement our training code using Alignment Handbook Codebase[4]. For inference, we use the vLLM package (Kwon et al., 2023)[5]. When evaluating performance in generating writing suggestions, for vLLM, we set the temperature to 0.0, top_p and top_k to 1, presence_penalty to 0, and best_of to 1. During sampling, we set the temperature parameter to 1.2, top_p to 0.9, top_k to 50, presence_penalty to 1.2, and max_length to 2048. Our reward model code is based on the HuggingFace TRL package[6]. We employed DeepSpeed's Zero-Offload[7] and LoRA techniques for fine-tuning the WA and reward models. Our experiments are conducted on the 4 A100 40GB GPUs, with CUDA version 11.7. The CPU is AMD EPYC 9554P, with 792GB RAM. Training the SFT model required six hours, while additional fine-tuning with our self-training alignment method took 12 hours. For OpenAI models, we set the temperature to 0.7, and use the following model cards: InstructGPT: gpt-3.5-turbo-instruct; ChatGPT: gpt-3.5-turbo-1106; GPT-4: gpt-4-0613, and GPT-4o: gpt-4o-2024-08-06.

Task	Validation Set	# Sent
GEC	W&I+LOCNESS-Dev	4.3k
Fluency	ITERATER-Dev	115
Clarity	ITERATER-Dev	157
Coherence	ITERATER-Dev	41
Paraphrase	STS-Dev	56
Neutralization	WNC-Dev	700
Simplification	ASSET-Dev	2k
STF	GYAFC-EM-Dev	2.8k
	GYAFC-FR-Dev	2.7k

Table 5: The statistics of validation data.

B.4 Training Configuration and Hyper-Parameters.

When building the SFT model, we train for four epochs with a batch size of 128 and a learning rate of 5e-4. The target module for LoRA is query,

[4]https://github.com/huggingface/alignment-handbook
[5]https://github.com/vllm-project/vllm
[6]https://github.com/huggingface/trl
[7]https://github.com/microsoft/DeepSpeed

Method	Objective	Hyperparameter
CPO (Xu et al., 2024)	$-\log \sigma\left(\beta \log \pi_\theta(y_w\|x) - \beta \log \pi_\theta(y_l\|x)\right) - \lambda \log \pi_\theta(y_w\|x)$	$\lambda = 1.0,\ \beta \in [0.01, 0.05, 0.1]$
SimPO (Meng et al., 2024)	$-\log \sigma\left(\frac{\beta}{\|y_w\|} \log \pi_\theta(y_w\|x) - \frac{\beta}{\|y_l\|} \log \pi_\theta(y_l\|x) - \gamma\right)$	$\beta \in [2.0, 2.5]$ $\gamma \in [0.3, 0.5, 1.0, 1.2, 1.4, 1.6]$
MPO (Ours)	$-\lambda \log \sigma\left[\beta\left(\log \pi_\theta(y_w\|x) - \log \pi_\theta(y_l\|x) - \gamma(r_w - r_l)\right)\right] - \log \pi_\theta(y_w\|x)$	$\beta \in [0.01, 0.05, 0.1]$ $\lambda \in [0.2, 0.4, 0.6, 0.8, 1.0]$ $\gamma \in [0.2, 0.4, 0.6, 0.8, 1.0]$

Table 6: Various preference optimization objectives and hyperparameter search range.

key, value, and output projection. Both lora_r and lora_alpha are set to 16, and lora_dropout is set to 0.05. We use the same configuration when building our final WA model, and change the learning rate to 1e-5. Our reward model uses the same setting, but with a learning rate of 3e-4, and we train a task-specific adapter for each individual task.

There are in total six hyper-parameters in our system: (1) K: the number of folds to perform K-fold training. (2) N: the number of samples for creating the preference data for the reward modeling phase. (3) M: the number of samples for creating the preference data for the preference optimization phase. (4) λ: the weight for $\mathcal{L}_M$. (5) β: the scaling constant in Eq. (1c). (6) γ: the weight for the reward margin. We do not tune the values of K, N, and M. We set K to 5, and both N and M to 16. The value of λ is tuned from 0.2 to 1.0 in increments of 0.2, with the best performance on the validation set achieved at 0.8. The value of β is tuned from {0.01, 0.05, 0.1}, with the best performance achieved at 0.01. The value of γ is tuned from 0.2 to 1.0 in increments of 0.2, with the best performance obtained in 1.0.

Table 6 shows the hyper-parameter search ranges for CPO and SimPO. When optimizing the SFT model with PPO, we adopt the default configuration from LLaMA-Factory[8] and tune the learning rate from 1e-5 to 1e-3 following the PPO practices.[9]

C Analysis Setting

C.1 Reward Model Experiments

Setting. When generating the preference pairs for the reward model, we randomly sample 1,000 sentences from these pairs as the validation data. Note that these pairs have not been used for training the reward model. We use both the reward model and automatic metrics to rank these 1,000 sentences and calculate the Pearson correlation (Cohen et al., 2009) between these two rankings. As shown in Table 1, the model's performance varies greatly across different tasks. To better visualize the correlation between our reward model and the automatic metrics, when plotting Fig. 4, for each task, we first scale the metric's score to 0 to 100 through min-max normalization and then combine different tasks' scores together.

Metrics' Correlation with Human Preferences This subsection summarizes studies that examine the correlation between evaluation metrics and human preferences across various tasks. Dahlmeier and Ng (2012) demonstrated that the MaxMatch scorer positively correlates with human judgments in the GEC task, as evaluated on the CoNLL-2014 test set. Similarly, Alva-Manchego et al. (2020) found that SARI aligns well with human preferences in the text simplification task (ASSET test set). Furthermore, Rao and Tetreault (2018) reported a positive correlation between BLEU scores and human evaluations in the FST task (GYAFC-EM/FR test set). Finally, Dwivedi-Yu et al. (2022) mentioned that SARI significantly correlates with human judgments on the ITERATER-fluency, ITERATER-clarity, ITERATER-coherence, STSB test set, and the WNC test sets.

C.2 GPT-4 Evaluation

Suggestion Evaluation. Kim and Kim (2024) developed a Likert scale prompt ranging from 1 to 5 to evaluate text quality with GPT-4, focusing on grammaticality, fluency, coherence, and consistency. Their study demonstrated a strong correlation between GPT-4's evaluation results and those of human evaluators. Following their method, we utilized the prompt presented in Table 16 and the scoring rubric detailed in Table 10 to assess text quality across the same dimensions: grammaticality, fluency, coherence, and consistency.

[8] https://github.com/hiyouga/LLaMA-Factory/blob/main/examples/train_lora/llama3_lora_ppo.yaml

[9] https://github.com/llSourcell/Unity_ML_Agents/blob/master/docs/best-practices-ppo.md

System	**CoN**	**ITR-F**	**ITR-L**	**ITR-O**	**STS**	**WNC**	**AST**	**GYAFC**	
								EM	**FR**
SFT model (R)	62.7 ± 0.3	51.1 ± 0.2	34.0 ± 0.3	37.5 ± 0.6	45.1 ± 0.8	65.8 ± 0.4	41.3 ± 0.3	75.9 ± 0.2 / 98.4 ± 0.4	76.4 ± 0.3 / 97.5 ± 0.2
+ Self-Training Alignment (R)	65.5 ± 0.6	48.7 ± 0.4	35.4 ± 0.3	37.6 ± 0.6	46.5 ± 0.6	65.9 ± 0.2	45.2 ± 0.5	77.3 ± 0.2 / 97.9 ± 0.3	78.2 ± 0.3 / 97.3 ± 0.4

Table 7: Means and variances for ExalWrite-8B on writing-related tasks. All the results are shown in %.

Comparsion	CoN	ITR-F	ITR-L	ITR-O	STS	WNC	AST	GYAFC -EM	GYAFC -FR
w/ rat vs. w/o rat	0.01	0.02	0.005	0.001	0.003	N.A.	N.A.	0.01	0.02
w/ self-align vs. w/o self-align	0.003	N.A.	0.01	0.04	0.005	0.04	0.003	0.0008	0.0009

Table 8: P-value for significance tests for ExalWrite-8B. 'w/rat vs. w/o rat' represents comparing 'SFT model' with 'SFT model (R)', and 'w/ self-align vs. w/o self-align' represents comparing 'SFT model (R)' with 'SFT model (R) + Self-Training Alignment (R)'. A p-value < 0.05 indicates statistical significance.

```
[Task Instruction]

### Input:
[Input Text]

### Hypothesis:
[Output Text]
```

Table 9: Instruction template for the reward model.

Rationale Evaluation. Song et al. (2024) provided detailed evaluation criteria for assessing the accuracy of rationales generated by LLMs.[10] We formalize this evaluation guideline into prompts shown in Table 17 which allow GPT-4 to evaluate the accuracy of the rationales. Following their guidelines, we formalized these criteria into prompts, as shown in Table 17, to enable GPT-4 to evaluate the accuracy of the rationale. We randomly selected 1,000 source-target sentence pairs from the GEC training data and extracted rationales generated by GPT-4 using the method outlined in Song et al. (2024). Using the prompts in Table 17, we achieved an accuracy of 91.8%, closely aligning with the 92.2% reported by Song et al. (2024). This result demonstrates that GPT-4, when guided by these prompts, serves as a reliable rationale evaluator.

C.3 Precision Computation

To extract edits, we use the ERRANT toolkit[11]. The precision of these edits is determined by comparing the system's proposed edits with all annotated edits in the validation dataset. Specifically, an edit is considered a true positive (TP) if it matches any of the annotated edits.

D Detailed results

Full Results. Table 15 extends the performance presented in Table 1 by including additional models. Specifically, we additionally included the zero-shot performance of Llama-2 models, Llama-3.1 models, and InstructGPT. Furthermore, we examine the experimental results using Llama-2-13B as the base model for both the WA and reward models. Additionally, Table 15 summarizes the performance of state-of-the-art (SOTA) systems across various tasks (Prev. SOTA). It is important to note that these SOTA systems are mostly task-specific (not multi-task) transformer models that leverage a range of task-specific features. Consequently, their results (Prev. SOTA) are not directly comparable to those of other LLMs.

Mean and Variance. We show the mean and variance of three runs of RealEdit-8B for each task in Table 7.

Statistical Significance Test. As mentioned in §4.1, we use the one-tailed sign test with boostrap resampling to carry out the statistical significance test. The p-value for the best performing model (RealEdit-11B) is shown in Table 8.

E Instructions

Detailed task instructions for Fig. 3 are provided in Table 12. Table 11 presents our template used for

[10] https://docs.google.com/presentation/d/1cYuvdAyd8xb2mZnlRXsOUSjcl6shX_cqqoYlb9KQZGI/edit?pli=1#slide=id.g290c5060bc5_0_5

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

Criteria Description	**Grammaticality**: Does the text demonstrate proper grammatical usage?
Score 1 Description	The text contains frequent grammatical errors, making it difficult to understand.
Score 2 Description	The text shows occasional grammatical errors, which disrupt the flow and clarity of the text.
Score 3 Description	The text generally adheres to grammatical rules, though minor errors are present.
Score 4 Description	The text demonstrates good grammaticality with rare errors that do not affect comprehension.
Score 5 Description	The text excels in grammatical usage, with clear and correct grammar throughout.
Criteria Description	**Fluency**: Is the text fluent and easy to read?
Score 1 Description	The text is disjointed and lacks fluency, making it hard to follow.
Score 2 Description	The text has limited fluency with frequent awkward phrasing.
Score 3 Description	The text is moderately fluent, with some awkward phrasing but generally easy to follow.
Score 4 Description	The text is fluent with smooth transitions and rare awkward phrases.
Score 5 Description	The text is highly fluent, with natural and smooth expression throughout.
Criteria Description	**Consistency**: Is the text consistent in terms of style, tone, and tense?
Score 1 Description	The text is inconsistent in style, tone, and tense, leading to confusion.
Score 2 Description	The text shows occasional inconsistencies in style, tone, and tense.
Score 3 Description	The text is mostly consistent in style, tone, and tense, with minor lapses.
Score 4 Description	The text is consistent in style, tone, and tense, with rare inconsistencies.
Score 5 Description	The text is highly consistent in style, tone, and tense throughout.
Criteria Description	**Coherence**: Is the text coherent and logically organized?
Score 1 Description	The text is incoherent and lacks logical organization, making it difficult to understand.
Score 2 Description	The text shows some coherence but contains several disjointed ideas and poor organization.
Score 3 Description	The text is generally coherent with a logical flow, though minor lapses in organization may occur.
Score 4 Description	The text is coherent and well-organized with clear connections between ideas.
Score 5 Description	The text is highly coherent, with a strong logical structure and seamless organization.

Table 10: Scoring rubrics on a 1-5 scale for the GPT-4.

instruction tuning, while Table 14 presents the task-specific instructions employed during both training and testing. Table 9 shows our template used for instruction tuning in the reward modeling process to build the reward model, and Table 13 shows our task-specific instruction used during training and testing in the reward modeling process.

Below is an instruction that describes a task, paired with an input that provides further context. Write a response that appropriately completes the request.

###Instruction:
`[Task Prompt]`

###Input:
`[Input Text]`

###Response:
`[Output Text]`

Table 11: Instruction template used during training and testing both the SFT model and WA model.

GEC	Specifically, the target sentence corrects the grammar mistakes in the source sentence.
Fluency	Specifically, the target sentence makes the source sentence more fluent.
Clarity	Specifically, the target sentence provides a clarification of the source sentence.
Coherence	Specifically, the target sentence provides a cohesive representation of the source sentence.
Paraphrase	Specifically, the target sentence is a well-phrased paraphrase of the source sentence.
Neutralization	Specifically, the target sentence provides a neutralization of the source sentence.
Simplification	Specifically, the target sentence provides a simplification of the source sentence.
FST	Specifically, the target sentence transforms the source sentence into a formal style.

Table 12: `[Task Instruction]`s for each task.

GEC	The hypothesis corrects all grammar mistakes in the input.
Fluency	The hypothesis makes the input more fluent.
Clarity	The hypothesis is a good clarification of the input.
Coherence	The hypothesis is a good cohesive representation of the input.
Paraphrase	The hypothesis is a well-phrased paraphrase of the input.
Neutralization	The hypothesis is a good neutralization of the input.
Simplification	The hypothesis is a good simplification of the input.
FST	The hypothesis correctly transforms the input into a formal style.

Table 13: `[Task Instruction]`s for each task.

GEC	Correct grammatical errors in the text by first providing a response, followed by an explanation. Please use this template for the explanation: "The word X should be deleted/inserted/replaced by Y because ..."
Fluency	Make the text more fluent and grammatical by initially providing a response, followed by an explanation. Please use this template for the explanation: "The word X should be deleted/inserted/replaced by Y because ..."
Clarity	Enhance the readability of the text by initially providing a response, followed by an explanation. Please use this template for the explanation: "The word X should be deleted/inserted/replaced by Y because ..."
Coherence	Enhance the cohesion of the text by initially providing a response, followed by an explanation. Please use this template for the explanation: "The word X should be deleted/inserted/replaced by Y because ..."
Paraphrase	Paraphrase the text by initially providing a response, followed by an explanation. Please use this template for the explanation: "The word X should be deleted/inserted/replaced by Y because ..."
Neutralization	Neutralize the text by initially providing a response, followed by an explanation. Please use this template for the explanation: "The word X should be deleted/inserted/replaced by Y because ..."
Simplification	Simplify the text by first providing a response, followed by an explanation. Please use this template for the explanation: "The word X should be deleted/inserted/replaced by Y because ..."
FST	Convert the following informal text into formal style by initially providing a response, followed by an explanation. Please use this template for the explanation: "The word X should be deleted/inserted/replaced by Y because ..."

Table 14: `[Task Prompt]`s for each task.

	System	CoN	ITR-F	ITR-L	ITR-O	STS	WNC	AST	GYAFC		ALL
									EM	FR	
a)	Llama-2-7B	40.6	34.4	28.4	22.1	33.7	19.8	30.8	26.3 / 67.9	22.4 / 66.4	28.7
	Llama-2-13B	51.7	42.4	33.7	29.8	35.8	28.3	35.7	47.3 / 70.2	50.2 / 70.7	39.4
	Llama-2-70B	55.6	49.1	35.8	35.6	37.3	34.6	38.0	57.8 / 78.2	58.4 / 75.9	44.8
	Llama-3.1-8B	52.2	41.5	31.2	28.9	35.3	29.4	36.1	48.2 / 71.3	51.2 /70.6	39.3
	Llama-3.1-70B	56.5	38.7	35.3	37.7	41.2	38.5	40.8	58.2 / 79.3	61.4 / 78.5	45.3
	Llama-3.3-70B-Instruct	55.6	46.5	31.4	31.0	34.6	31.8	46.4	59.5 / 98.1	56.2 / 98.4	43.7
	InstructGPT	51.5	48.8	35.1	35.9	42.5	35.4	38.0	58.7 / 95.2	59.5 / 95.0	45.0
	ChatGPT	53.3	50.9	31.5	31.0	39.9	36.3	47.0	57.7 / 99.6	60.4 / 99.5	45.3
	GPT-4	59.9	51.6	32.6	32.3	42.2	40.8	46.3	60.2 / 99.6	62.4 / 99.5	47.6
	GPT-4o	59.4	51.1	32.4	32.4	42.4	41.1	47.4	62.8 / 99.2	63.7 / 99.1	48.1
b)	Llama-2-7B (R)	42.3	36.2	34.2	29.2	34.2	19.2	31.7	40.1 / 68.1	34.3 / 67.8	33.4
	Llama-2-13B (R)	52.4	41.5	33.8	29.5	29.6	29.2	36.1	64.8 / 72.7	69.6 / 73.1	42.9
	Llama-2-70B (R)	56.2	50.9	30.5	35.8	36.7	38.7	38.5	65.3 / 74.1	68.4 / 73.8	46.8
	Llama-3.1-8B (R)	52.4	51.4	33.9	35.6	34.9	30.8	35.0	49.0 / 71.6	54.3 / 70.8	41.9
	Llama-3.1-70B (R)	56.0	50.7	31.8	36.3	37.9	39.3	39.4	68.9 / 80.3	71.9 / 79.4	48.0
	Llama-3.3-70B-Instruct (R)	58.4	49.4	35.0	31.8	37.7	41.2	44.7	63.6 / 97.8	65.3 / 98.2	47.5
	InstructGPT (R)	53.4	49.2	34.2	34.3	43.2	35.5	39.1	63.3 / 95.2	63.2 / 95.0	46.1
	ChatGPT (R)	56.1	51.1	30.3	28.7	40.6	36.6	45.0	63.1 / 98.7	63.5 / 98.9	46.1
	GPT-4 (R)	60.4	50.1	33.3	32.8	41.2	40.7	**47.6**	63.2 / 98.8	63.5 / 99.1	48.1
	GPT-4o (R)	60.8	51.4	32.6	32.2	43.3	40.9	46.1	64.4 / 98.4	64.6 / 98.6	48.5
c)	PEER-EDIT-11B	N.A.	52.1	32.5	32.7	28.2	54.5	29.5	N.A.	N.A.	N.A.
	Writing-Alpaca (7B)	55.9	52.8	**39.4**	37.1	44.6	64.4	44.7	N.A.	N.A.	N.A.
	CoEDIT-xxl (11B)	57.1*	51.6	31.8	31.5	42.9*	**71.0**	41.7	66.0 / 98.7*	68.7 / 97.9*	51.7
	Prev. SOTA	72.8	52.8	46.2	38.3	43.3	71.0	44.6	78.8 / 94.6	81.4 / 86.4	N.A.
	Ours based on Flan-T5-xxl (11B) (RealEdit-11B)										
d)	SFT model	58.3	50.9	33.6	32.2	43.0	70.8	41.4	69.2 / 97.3	70.5 / 97.1	52.1
	+ Self-Training Alignment	61.4	49.3	32.8	34.7	47.0	68.9	41.1	75.3 / 96.8	78.0 / 96.3	54.3
e)	SFT model (R)	61.8	51.3	30.2	36.1	46.6	69.0	43.1	73.4 / 97.4	76.2 / 97.1	54.1
	+ Self-Training Alignment (R)	62.1	52.5	33.5	38.6	44.7	70.2	42.8	75.6 / 97.1	77.4 / 96.9	55.2
	Ours based on Llama-2-13B (RealEdit-13B)										
f)	SFT model	58.1	49.5	32.5	38.9	41.3	66.5	41.1	72.8 / 97.8	74.0 / 96.7	52.7
	+ Self-Training Alignment	60.8	**52.9**	31.4	**40.8**	44.2	66.8	41.4	75.6 / 96.2	77.0 / 95.5	54.5
g)	SFT model (R)	60.9	53.2	32.1	35.0	45.0	67.3	41.5	76.7 / 98.2	77.8 / 97.2	54.3
	+ Self-Training Alignment (R)	62.6	51.8	35.9	37.8	45.3	66.7	43.3	77.5 / 97.8	78.4 / 97.1	55.5
	Ours based on Llama-3.1-8B (RealEdit-8B)										
h)	SFT model	61.7	50.5	31.6	35.6	43.3	66.4	42.0	75.3 / 97.9	75.5 / 96.8	53.5
	+ Self-Training Alignment	62.5	48.7	31.9	40.3	**47.1**	64.6	45.2	**78.0** / 96.8	78.0 / 95.9	55.1
i)	SFT model (R)	62.7	51.1	34.0	37.5	45.1	65.8	41.3	75.9 / 98.4	76.4 / 97.5	54.4
	+ Self-Training Alignment (R)	**65.5**	48.7	35.4	37.6	46.5	65.9	45.2	77.3 / 97.9	**78.2** / 97.3	**55.6**

Table 15: Performance on writing-related tasks. All results are shown in %. *: Results reproduced using the official checkpoint and scripts released by Raheja et al. (2023), due to different evaluation metrics or test sets not previously evaluated. For the GYAFC test sets, the first score is BLEU and the second is accuracy. Following (Raheja et al., 2023; Zhang et al., 2023), we show the averaged result under the ALL column, and we only consider the BLEU score for the GYFAC test sets when taking the average. **a)**: zero-shot performance of LLMs. **b)**: zero-shot performance of LLMs when also prompted to generate rationales (or explanations) for their writing suggestions. **c)**: SOTA WAs. **d) & f) & h)**: RealEdit trained without rationale. **e) & g) & i)**: RealEdit trained with rationale. The previous SOTA (**Prev. SOTA**) from left to right are from Omelianchuk et al. (2024) (CoN), Zhang et al. (2023) (ITR-F, STS, AST), Du et al. (2022) (ITR-L, ITR-O), Raheja et al. (2023) (WNC), Liu et al. (2022) (GYAFC-EM/FR).

```
### Task Description:
An instruction (might include an Input inside it), a response to evaluate, and a score rubric representing a evaluation criteria are given.
1. Write a detailed feedback that assess the quality of the response strictly based on the given score rubric, not evaluating in general.
2. After writing a feedback, write a score that is an integer between 1 and 5. You should refer to the score rubric.
3. The output format should look as follows: “(write a feedback for criteria) [RESULT] (an integer number between 1 and 5)”
4. Please do not generate any other opening, closing, and explanations.

### The instruction to evaluate:
[Instruction]

### Response to evaluate:
[Response]

### Score Rubrics:
[Rubric]

### Feedback:
```

Table 16: Instruction template use for GPT-4 to score the output. `[Instruction]` represents the instruction and the input (x), and the `[Response]` represents the WA's suggestion (s), and `[Rubric]` is shown in Table 10.

```
### Task Description:
A source sentence, a target sentence, its corresponding explanation, and an evaluation criteria are given. You should evaluate the explanation according to the evaluation criteria
1. Write a detailed feedback that assess the quality of the response strictly based on the given evaluation criteria, not evaluating in general.
2. After writing a feedback, write a response that is an string either ‘Wrong’ or ‘Correct’. You should refer to the evaluation criteria.
3. The output format should look as follows: “(write a feedback for criteria) [RESULT] (‘Wrong’ or ‘Correct’)”
4. Please do not generate any other opening, closing, and explanations.

### EVALUATION CRITERIA:
### Read the source and target sentences and the explanation:
1. If the error in an explanation is not an actual error in the source sentence, return the result as ‘Wrong’.
2. If there is an error in the source sentence that is not explained, return the result as ‘Wrong’.
3. If an error is not hallucinated but its description or explanation is wrong, return the result as ‘Wrong’.
4. If the assigned error type is wrong, return the result as ‘Wrong’.
5. For the rest of the case, return the result as ‘Correct’.

### SOURCE:
[Source]

### TARGET:
[Target]

### EXPLANATION:
[Rationale]

### Feedback:
```

Table 17: Instruction template use for GPT-4 to score the accuracy of the rationale. `[Source]` represents the input (in), and the `[Target]` represents the WA’s suggestion (s), and `[Rationale]` represents the WA’s generated rationale (e).