Towards an On-device Agent for Text Rewriting

Anonymous ACL submission

Abstract

Large Language Models (LLMs) have demonstrated impressive capabilities for text rewriting. However, creating a smaller yet potent language model for text rewriting presents two formidable challenges: costly data collection and absence of emergent capabilities. In this paper, we present solutions to address the above challenges. We propose an new instruction tuning method to develop a mobile text rewriting model that leverages LLM-generated data and heuristic reinforcement learning, eliminating the need for human data collection. Moreover, to bridge the performance gap from the constraint size, we propose a cascading approach based on the confidence levels which are distilled from the large server model’s critiques. To evaluate the text rewriting tasks for mobile scenarios, we introduce MESSAGEREWRITEEVAL, a human-labeled benchmark that focuses on text rewriting of messages through natural language instructions. Through empirical experiments, we demonstrate that our on-device model surpasses the current state-of-the-art LLMs in text rewriting while maintaining a significantly reduced model size using public benchmark EDITEVAL and our new benchmark. We show that our proposed cascading approach improves model performance further.

1 Introduction

The process of text rewriting can be considered a form of controlled text generation (Zhang et al., 2022), where text inputs are modified based on user specifications. Various text rewriting categories have been extensively explored, including paraphrasing (Siddique et al., 2020; Xu et al., 2012), style transfer (Riley et al., 2020; Zhang et al., 2020; Reif et al., 2021), sentence fusion (Mallinson et al., 2022), and sentence compression (Mallinson et al., 2018; Stahlberg et al., 2022). The advent of Large Language Models (LLMs) (Passos et al., 2023; Brown et al., 2020; Touvron et al., 2023) has ushered in a new era for text rewriting, demonstrating unparalleled quality by harnessing pre-trained models (Shu et al., 2023). With the widespread use of mobile communications and text messaging (Hanson et al., 2010; Pennington et al., 2022), these LLMs are being integrated into text rewriting applications, enabling users to create messages that are “formal”, “concise” etc. (Burke, 2023).

Despite the impressive text rewriting ability enabled by LLMs, their deployment for real-world chat messaging faces practical issues. While deploying large models on users’ devices is impractical due to their size, server-based architectures introduce several drawbacks. They make it harder to preserve user privacy (Li et al., 2021), limit the models’ ability to operate offline (Murshed et al., 2021), and incur higher overall compute costs (Chen et al., 2023a). Developing a compact yet potent language model presents two unique challenges. First, training smaller models requires significantly larger datasets which requires costly data collection (Kang et al., 2023). Second, the emergent capabilities of the LLM only appears after reaching a critical size (Wei et al., 2022).

In this paper, we present a systematic approach for enhancing the rewriting capability of LLMs while adhering to size constraints to ensure reasonable on-device inference speeds. We introduce a benchmark called MESSAGEREWRITEEVAL, compiled from human-donated message texts and rewrites by human with diverse language instructions. Unlike existing benchmarks for text rewriting such as EDITEVAL (Dwivedi-Yu et al., 2022) or OPENREWRITEEVAL (Shu et al., 2023) which are derived from text sourced from paragraphs or long passages, our benchmark is designed to better represent daily conversational exchanges between individuals.

Inspired by InstructGPT (Ouyang et al., 2022), we train our model using a combination of super-
vised fine-tuning (SFT) and reinforcement learning (RL). While InstructGPT relies heavily on human raters for both instruction data and preference data, our approach minimizes human intervention in the data collection process. To elaborate:

(1) For instruction data generation, we develop a novel method based on continued generations from LLMs to generate high quality synthetic data.

(2) Instead of using a reward model (Shu et al., 2023), we propose a heuristic-based reward signal for reinforcement learning that can improve the model without additional labeling. We conduct empirical investigations to assess the model’s performance against the MESSAGE\textsc{RewriteEval} and EDIT\textsc{Eval} benchmarks. Our proposed model outperforms its corresponding foundation model and other instruction-tuned LLMs, which validates the usefulness of the generated training data and the proposed heuristic reinforcement learning.

To further mitigate the size constraints and bridge the gap between the on-device model and the giant server-side LLMs, we propose a cascading approach to chain our on-device model with the more powerful server model. The system follows a simple yet effective principle: the server side will only be used when the on-device language model fails to provide a good response. Instead of relying on an external model to judge the quality of response (Chen et al., 2023a), we propose to add a simple suffix to the on-device model output that indicates how confident the model is in its prediction. The suffix is learned from the larger server-side LLM via distillation. Our findings demonstrate that the proposed cascading approach further enhances performance.

Our main contributions can be summarized as follows:

- We develop a powerful LLM that demonstrates superior performance compared to the state-of-the-art LLMs for text rewriting while being efficient for on-device inference. Importantly, this model’s efficacy does not rely on human-labeled data collection. We devise innovative strategies to generate varied instruction datasets for rewriting, that enhance the editing and rewriting capacities of the model. Additionally, we present a heuristic-based reinforcement learning approach that eliminates the need for training the reward model.

- We design an effective cascading mechanism to connect our on device model to the server-side model. We distill the critiquing ability of the server LLM to the smaller model using discriminative training, which enables efficient inference. Our cascading strategy can further improve the on-device model’s performance, bringing it closer to the capabilities of the server-side model while reducing the number of server calls.

- We introduce a new benchmark, MESSAGE\textsc{RewriteEval}, designed for research on message text rewriting and covering different types of rewrites expressed through natural language instructions: formality, elaboration, shortening, paraphrasing, and proofreading. To the best of our knowledge, no such benchmark is currently available.

2 Related Work

2.1 Text Editing

The text editing (Chuklin et al., 2022) task covers a wide range of sub-tasks such as paraphrasing (May, 2021), style transfer (Tikhonov et al., 2019), spelling and grammatical error correction (Napoles et al., 2017), formalization (Rao and Tetreault, 2018), simplification (Xu et al., 2016) and elaboration (Iv et al., 2022). Recent work has investigated a more diverse set of rewrite options (Faltings et al., 2020; Schick et al., 2022; Shu et al., 2023) by leveraging the diversity of edits in Wikipedia. While our model can take diverse prompts as input, its core strength is on rewriting messages through formalizing, shortening, elaborating, paraphrasing, and proofreading.

2.2 Instruction Tuning

Instruction tuning has been shown to improve model performance and generalization to unseen tasks (Chung et al., 2022; Sanh et al., 2022). InstructGPT (Ouyang et al., 2022) extends instruction tuning using reinforcement learning with human feedback (RLHF), which heavily relies on human raters to obtain instruction data and rankings of model outputs. The dependency on human preference data could be alleviated by reinforcement learning with AI feedback (RLAIIF) (Bai et al., 2022; Shu et al., 2023), but training a separate reward model is still required. We extend this framework using a heuristic based reinforcement learning (Cheng et al., 2021) for rewriting tasks, which enables reinforcement learning without a reward model.
2.3 Distillation and Data Augmentation

Knowledge distillation (Hinton et al., 2015) has been successfully used to transfer knowledge from larger teacher models into smaller student models (Hinton et al., 2015; Tang et al., 2019; Wang et al., 2021; Smith et al., 2022; Beyer et al., 2022; Peng et al., 2023; Wu et al., 2023). The quality of distillation could be improved in a variety of ways such as using a better design of Chain-of-Thought prompts (Shu et al., 2023), combining the noisy predictions with majority vote (Arora et al., 2022), using a augmented label with reasoning (Hsieh et al., 2023), reweighting the student’s loss (Iliopoulos et al., 2022) etc. Unlike previous work, we use a pre-trained LLM to generate data and also provide critique for generated output, enabling automatic filtering. Furthermore, we extend our distillation technique to perform critiques.

2.4 LLM Cascades

Language model cascades have been investigated in many previous works (Li et al., 2020; Cai et al., 2023; Wu et al., 2022; Dohan et al., 2022). Frugal GPT (Chen et al., 2023a) proposed several strategies for using multiple LLMs to minimize the inference cost. For the cascaded design, the regression score from DistillBert (Sanh et al., 2019) is used for deciding whether or not the model response is adequate. Although our approach achieves a similar goal, it does not require an extra model. We incorporate this capability into the language model in a single pass text generation step by using the suffixes of the generation (Thoppilan et al., 2022).

3 Methods

Our approach follows the “supervised fine-tuning (SFT) + reinforcement learning (RL)” paradigm (Ouyang et al., 2022), but does not require any human labeling or preference data collection. We first discuss our approach to generate synthetic training data for supervised fine-tuning. We then present our heuristic reward and RL process. Finally, we describe our cascading method.

3.1 Supervised Fine-tuning

We follow existing works to leverage the document level edit data from Wikipedia (Schick et al., 2022; Shu et al., 2023). In pilot studies, we observed that using this data alone cannot provide adequate short form, message like data for training our on-device models. To generate in-domain data efficiently, we propose a data generation approach based on continued generation by off-the-shelf LLMs, which can then be filtered using LLMs. The details of the training data are provided in Section A.5.

3.1.1 Synthetic Paired Dataset from Continued Generations

To collect more shorter-form and message-like data, we leverage the few-shot capability of pre-trained LLMs. Figure 1 shows an example of the initial prompts and demonstrates how the LLM is continuing to generate diverse examples from a given query, which is sampled from a small seed query set. The continued generations enable efficient generation of diverse paired data.

3.1.2 An LLM guided data selection

To further improve the quality of our synthetic dataset, we propose to use LLMs to critique the generated data. We leverage the few-shot Chain-of-Thoughts (CoT) reasoning of the off-the-shelf LLM to judge whether the response is following the instruction of the prompt to rewrite the original sentence in a good manner. We provide detailed prompt samples in Table 15. We also leverage the self-consistency (Wang et al., 2022a) approach to improve the accuracy. Specifically, we only keep the data when it is approved by all LLM judges.

3.1.3 Generative Fine-tuning

Given a pre-trained decoder-only language model, we fine-tune it using the collected instruction tuning dataset. The input is formed by concatenating...
the <instruction> and the <source> with a newline, while the output is the <target>.

3.2 Heuristic based Reinforcement Learning

The reinforcement learning part is typically called Reinforcement Learning with Human Feedback (RLHF) (Ziegler et al., 2019) as human labelers are heavily involved in training the reward model. In this section, we introduce a novel approach to improve alignment through heuristics without any human labeling.

3.2.1 Heuristic Reward

The intuition is that a few common heuristics can yield high quality rewrites. We propose to use the following heuristics as reward signals.

Natural Language Inference (NLI) (Bowman et al., 2015) scores over the source-prediction pair. Given a “premise” and a “hypothesis”, NLI scores the probability that the “hypothesis” is correct given the “premise”. In the context of LLMs, NLI score estimates whether the LLM’s output prediction preserves meaning and factuality given the source text. We use the off-the-shelf NLI predictor from (Honovich et al., 2022), denoted as nli.

Reversed NLI. NLI score where the premise and the hypothesis are reversed, denoted rnli.

Length Ratio. The ratio of the number of tokens in the LLM output text to that in the source text, denoted length_ratio.

Edit Distance Ratio (Edit Ratio). Edit distance (Levenshtein, 1966) measures the minimum number of token-level edits (insertions, deletions and substitutions) to convert a source text into a target text. We use the relative edit distance between the prediction and source text, computed as the ratio of the edit distance to the length of the source text. The edit ratio, denoted as edit_ratio, represents the proportion of the source text that has been modified.

N-gram frequency. Text generation can easily get stuck in undesirable sentence-level loops with decoding (Xu et al., 2022). We propose measuring the N-gram frequency to detect potential loops in the generated output – if the frequency of a certain N-gram is too high, we introduce a constant negative reward to penalize it. We denote the output of this algorithm as ngram_reward.

We formulate the final reward as a weighted combination of all the signals above in equation (1). For different rewriting tasks, the coefficient $\sigma_i$ should be designed to reflect the expectation of the rewrites. For instance, the expectation for “shorten” is higher nli value (a larger positive $\sigma_1$) and lower length_ratio (a negative $\sigma_3$). We share the choice of hyper-parameter $\sigma_i$ in Appendix Table 8.

$$
\text{Reward} = \sigma_1nli + \sigma_2rnli + \sigma_3length\_ratio + \sigma_4edit\_ratio + \sigma_5ngram\_reward
$$

3.2.2 Reinforcement Learning

We further refine the fine-tuned model by employing reinforcement learning (Ouyang et al., 2022), guided by the heuristics provided. The prompts for reinforcement learning are collected from the LLM during training data generation. For each prompt in the train set, we first use LLM’s fewshot ability to classify the prompt into the rewrite types. During the reinforcement learning, this rewrite type will be fed to the “heuristic reward” module to generate the reward, which will be finally optimized through PPO (Schulman et al., 2017).

3.3 Critique Distillation and Model Cascade

We apply a simple cascade mechanism whereby the on-device model serves as the first gate to the incoming rewrite request, and the large server side model is invoked only when the on-device rewrite is deemed low quality. Towards this goal, we need to answer two questions. First, how to enable the on-device model to do “self-critique”, which is challenging given its small size and the complexity of the task. Second, how do we make the process more efficient without additional inference steps. We next present our suffix based distillation approach as a solution to the above questions. We leverage the off-the-shelf LLM as a critic and distill its knowledge as an extra “suffix” in the data into the on-device model. The approach is summarized in Figure 2.

![Figure 2: The illustration of distillation for self-critiques. The final sentence with “quality is good” as suffix will be used as training data for discriminative training.](image)

3.3.1 Critique Distillation from LLMs

Similar to reinforcement learning, we prepare unpaired prompt data sampled from continued generation of LLMs. The responses are generated by
our model. Then the (prompt, response) pairs are fed to the off-the-shelf LLM to decide whether they are acceptable or not. We leverage the Chain-of-Thought (CoT) reasoning along with the self-consistency approach. We use the prompts shown in Appendix Table 15.

3.3.2 Discriminative Fine-tuning
Although generative fine-tuning with the larger LLM’s response can make it possible to perform self-critiquing for the small models, “generation” and “self-critique” will be two separate text generation steps, resulting in increased inference times. To fuse the two steps, we transform generative fine-tuning into discriminative fine-tuning (Thoppilan et al., 2022). This is done by concatenate a label (“good”/“bad”) to the response with some predefined delimiter. In this way, we can generate the suffix data distilled from the critique provided by the off-the-shelf LLM. Finally we finetune the on-device model with the suffix data along with original generations.

3.3.3 Cascading
Once the model is finetuned with suffix data, it can use the suffix score, i.e. probability of outputting “good”, to decide whether to cascade. Specifically, after decoding some text we compare the “suffix score” s (which is a probability between 0 and 1) and some pre-defined threshold γ. If s > γ, the on-device model is deemed confident; Otherwise, the model relies on the server side model.

4 Experiment Settings
4.1 Model Training Setting
Our pre-trained checkpoint is PaLM 2-XXS. We leverage pre-trained PALM 2-L as the off-the-shelf LLM for data generation, LLM filtering, and critique distillation. The training hyper-parameters for instruction tuning and reinforcement learning are listed in the Appendix Section A.4.

4.2 Evaluation Datasets
4.2.1 MessageRewriteEval
To evaluate the model performance in the on-device messaging scenario, we introduce MESSAGERewriteEval, a novel evaluation dataset specifically designed for message-level rewrite assessments. All text message pairs are sourced from real-life, human-written daily use cases and evaluated by human raters for data quality. To ensure comprehensive evaluation, these pairs encompass five text rewrite tasks: Formalize, Paraphrase, Shorten, Elaborate, and Proofread. Each text pair in the dataset consists of three components: source, target, and instruction. The task distribution statistics and example instructions are provided in Appendix Section A.1. The data collection guidelines are given in Appendix Section A.2.

4.3 Automatic Evaluation Metrics
We employ various metrics to evaluate the model’s quality:
- NLI (Bowman et al., 2015) and Reversed NLI (Section 3.2.1).
- Edit Distance Ratio (Edit Ratio) (Section 3.2.1).
- SARI (Xu et al., 2016) is an n-gram based metric that measures the similarity of a prediction to both the source and reference texts. The scores of add, retain and delete operations are computed by averaging n-gram scores. The SARI metric is obtained using an arithmetic average of the F1 scores of add and retain operations and the precision of the delete operation.
- BLEU (Papineni et al., 2002) is computed as a geometric mean of n-gram precisions of different orders.
- Update-ROUGE (Updated-R) (Iv et al., 2022) is a modified version of ROUGE (Lin and Hovy, 2003) that specifically computes ROUGE-L on the updated sentences rather than the full text.
- Success Rate We use the LLM to assess whether or not the response follows the instruction (i.e. “good” or not). Although a binary classification might be too coarse grained for evaluating rewrite quality, it is a very intuitive and straightforward
metric to show the merit of cascading. The LLM prompts are provided in Appendix Table 15.

**On-device Inference Ratio** For cascading experiments, A higher ratio means a smaller percent of server calls.

### 4.4 Baselines

Since it is designed for on-device application, our model has a compact size in comparison to other LLMs. In choosing baseline models, we prioritize the ones that are similar in size to ours. We choose the state-of-the-art pre-trained models PaLM 2 (Passos et al., 2023), LLaMA (Touvron et al., 2023) and the instruction tuned models Alpaca (Taori et al., 2023), Vicuna (Chiang et al., 2023), Flan-PaLM 2 (Passos et al., 2023) as our baseline models. We also provide Alpaca-PaLM 2 for comparison. The Alpaca’s instruction dataset is finetuned using a PaLM 2 baseline checkpoint.

For a fair comparison, we leveraged in-context learning with CoT few-shot prompting (we share the details in Appendix Section A.11) to instruct the model to provide reasonable responses for the pre-trained models since they are not instruction tuned. In contrast, for the instruction tuned LLMs including ours, we use zero-shot settings. For cascading, we note that constructing a powerful large language model is not within the the scope of this study. Therefore, our experiments utilize the 175B InsGPT (Ouyang et al., 2022) as the server model.

### 4.5 Human Evaluation

We follow the same human evaluation setup as the RewriteLM paper (Shu et al., 2023). 300 examples are randomly sampled from MESSAGERewriteEval for human evaluation with five language experts. A 3-point Likert scale (0-Bad, 1-Medium, 2-Good) is used for the following features: 1) **Instruction Success**: whether the output text follows the given instruction. 2) **Content Preservation**: whether the essential content of the input text are kept in the output text, independent from style or quality. 3) **Factuality**: whether the output content is accurate and truthful. 4) **Coherency**: whether the output text is non-ambiguous, and logically coherent written, independent from the input text. 5) **Fluency**: whether the output text is written with good clarity, correct grammar, and style. The detailed rating guideline is in Appendix A.8.

## 5 Results

### 5.1 Performance of the On-device Model

To show that our approach can generally enhance the model’s rewriting ability, we first report performance of our SFT model and RL model on EDIT-EVAL. And then we evaluate the same SFT model and RL model on MESSAGERewriteEval. We present latency and memory metrics for on-device inference in Appendix A.9.

#### 5.1.1 Results on EditEval

Table 1 summarizes the results. The metrics of the baseline models are directly obtained from the EditEval paper (Dwivedi-Yu et al., 2022). We list only those models whose sizes are similar to our on-device models; Nevertheless, our model is substantially smaller than these models. We provide SARI values for each dataset and extra Update-R scores for the two datasets relevant for the paragraph update task.

The results show that our on-device model with size **XSS** outperformed other models on most of the tasks despite being much smaller. For the fluency, coherence, paraphrase, simplification and paragraph update tasks, our model wins by a large margin. Heuristic reinforcement learning generally boosts the model’s performance on all tasks.

#### 5.1.2 Results on MESSAGERewriteEval

The automatic evaluation results for the MESSAGERewriteEval dataset are shown in Table 2. We first examine results of three sets of models: pre-trained LLMs, Instruction-Tuned LLMs and our on-device Instruction-Tuned LLMs.

Edit Ratio measure of token-level different between texts, We empirically observed that a larger Edit Ratio does not always correlate with better rewrite performance, as it often arises from hallucinations. In terms of SARI, BLEU, and Update-R metrics, our on-device size models outperform LLaMA, Alpaca-7B and Vicuna-7B, despite having a much smaller size. We also compare our results to Alpaca-PaLM 2 and Flan-PaLM 2, which share the same base architecture and model size. The fact that our model achieves much better SARI, BLEU, and Update-R scores validates the effectiveness of our approach. Moreover, the gap in performance between the SFT and RL models shows that our heurisic reinforcement learning is very effective. We performed three independent training runs of the RLed model and present the average and standard
The low standard deviations across metrics suggest consistency in the RLed model’s performance. We also study the role of each heuristic by doing ablations. We summarize results in Table 3. As we can see from the table, removing any one of the proposed heuristics will reduce the overall quality of rewrites. Notably the NLI s-t and the NLI t-s play more important roles for securing good rewrite comparing to other rewards.

### 5.2 Performance of Cascading

Our cascading experiments are conducted on the top of the on-device model with RL using MESSAGEWRITEEVAL benchmark. Here we choose it over EditEval for cascading experiments as it is more aligned with the mobile cases. We first evaluate how the critique distillation is impacting the model’s performance. Next we show the end-to-end cascading performance and a detailed analysis and demonstrate that our suffix score is more effective than the baseline LM score.

### 5.2.1 The Effect of Critique Distillation

In Table 2 we show that the model’s overall performance is further improved on SARI, BLEU, and Update-R with little regression on Reversed NLI when we combine the distilled discriminative dataset with the generative dataset. This suggests that with the suffix score from critique distillation, our models achieve best performance compared with all listed Pre-trained LLMs and Instruction-Tuned LLMs, which have either same or larger size then ours. When cascaded with InsGPT, the performance is further improved.

Table 1: Model Performance on EditEval (Dwivedi-Yu et al., 2022). Only models with reasonable sizes are listed. Size XXS is less than half the size of T0/Tk models. Despite their reduced sizes, our models achieve even better performance than most of the other larger models. Relative to similar-sized instruction-tuned models, our models win by a large margin.

<table>
<thead>
<tr>
<th>Model</th>
<th>Size</th>
<th>Edit Ratio</th>
<th>NLI</th>
<th>Reversed NLI</th>
<th>SARI</th>
<th>BLEU</th>
<th>Update-R</th>
</tr>
</thead>
<tbody>
<tr>
<td>InsGPT (Ouyang et al., 2022)</td>
<td>XXS</td>
<td>0.18</td>
<td>0.91</td>
<td>0.88</td>
<td>51.14</td>
<td>35.0</td>
<td>58.91</td>
</tr>
<tr>
<td>LLaMA (Touvron et al., 2023)</td>
<td>XXS</td>
<td>0.98</td>
<td>0.83</td>
<td>0.72</td>
<td>38.92</td>
<td>22.98</td>
<td>37.45</td>
</tr>
<tr>
<td>PalM 2 (Passos et al., 2023)</td>
<td>XXS</td>
<td>1.39</td>
<td>0.76</td>
<td>0.82</td>
<td>31.49</td>
<td>18.81</td>
<td>31.85</td>
</tr>
<tr>
<td>Alpaca (Taori et al., 2023)</td>
<td>XXS</td>
<td>0.26</td>
<td>0.76</td>
<td>0.76</td>
<td>42.21</td>
<td>24.80</td>
<td>45.15</td>
</tr>
<tr>
<td>Vicuna (Chiang et al., 2023)</td>
<td>XXS</td>
<td>0.12</td>
<td>0.86</td>
<td>0.52</td>
<td>38.18</td>
<td>14.30</td>
<td>30.17</td>
</tr>
<tr>
<td>Flan-PalM 2 (Passos et al., 2023)</td>
<td>XXS</td>
<td>0.11</td>
<td>0.94</td>
<td>0.83</td>
<td>29.50</td>
<td>25.89</td>
<td>34.63</td>
</tr>
<tr>
<td>Alpaca-PalM 2 (Passos et al., 2023)</td>
<td>XXS</td>
<td>0.11</td>
<td>0.93</td>
<td>0.80</td>
<td>27.41</td>
<td>17.59</td>
<td>15.43</td>
</tr>
<tr>
<td>SFT + heuristic RL (Ours)</td>
<td>XXS</td>
<td>0.3</td>
<td>0.84</td>
<td>0.78</td>
<td>43.14</td>
<td>25.88</td>
<td>47.76</td>
</tr>
<tr>
<td>Ours + InsGPT (40% server calls)</td>
<td>XXS</td>
<td>0.16</td>
<td>0.93</td>
<td>0.86</td>
<td>49.87</td>
<td>34.59</td>
<td>58.87</td>
</tr>
<tr>
<td>Ours + InsGPT (15% server calls)</td>
<td>XXS</td>
<td>0.16</td>
<td>0.92</td>
<td>0.86</td>
<td>49.03</td>
<td>33.76</td>
<td>57.41</td>
</tr>
</tbody>
</table>

Table 2: Model Performance on MESSAGEWRITEEVAL. Our models achieves best performance compared with all listed Pre-trained LLMs and Instruction-Tuned LLMs, which have either same or larger size then ours. When cascaded with InsGPT, the performance is further improved.

### Table 3: Ablation study for the heuristic rewards. Each experiment removes one heuristic and keep the rest.

<table>
<thead>
<tr>
<th>Heuristic</th>
<th>Edit Ratio</th>
<th>NLI</th>
<th>Reversed NLI</th>
<th>SARI</th>
<th>BLEU</th>
<th>Update-R</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.16</td>
<td>0.93</td>
<td>0.85</td>
<td>47.34</td>
<td>30.50</td>
<td>54.97</td>
</tr>
<tr>
<td>Edit Dist</td>
<td>0.13</td>
<td>0.93</td>
<td>0.85</td>
<td>47.30</td>
<td>30.28</td>
<td>54.21</td>
</tr>
<tr>
<td>Len Ratio</td>
<td>0.15</td>
<td>0.93</td>
<td>0.84</td>
<td>47.27</td>
<td>30.24</td>
<td>54.00</td>
</tr>
<tr>
<td>Ngram</td>
<td>0.15</td>
<td>0.92</td>
<td>0.85</td>
<td>47.22</td>
<td>30.32</td>
<td>53.96</td>
</tr>
<tr>
<td>NLI s-t</td>
<td>0.16</td>
<td>0.89</td>
<td>0.84</td>
<td>46.50</td>
<td>28.81</td>
<td>52.86</td>
</tr>
<tr>
<td>NLI t-s</td>
<td>0.15</td>
<td>0.92</td>
<td>0.78</td>
<td>47.11</td>
<td>28.91</td>
<td>52.40</td>
</tr>
</tbody>
</table>
the model tends to pick sample with higher quality.

5.2.2 End-to-end Performance

The on-device model’s reliance on the server model is controlled by the threshold $\gamma$. As shown in Table 2, the performance of the cascaded models lies between the on-device and the server side model. With a higher number of server calls, we obtain higher SARI, BLEU, and Update-R, as expected. With 40% server calls, the overall performance is already quite close to the full server model. We also profiled the latency of it and did more analysis in Appendix A.10.

5.2.3 Suffix Score vs LM Score

We now provide more insight into our cascading approach with suffix score. We vary the threshold $\gamma$ from 0 to 1 to measure Success Rate as a function of the On-device Inference Ratio. The trade-off between the two metrics is shown in Figure 3. To demonstrate the efficacy of the distilled suffix score derived from larger LLM critiques as a reliable indicator of output quality, we compare it with an LM score, representing the likelihood of the generated text. As shown in Figure 3, “suffix score with 1 sample” is outperforming “LM score with 1 sample” by large margin. This indicates that given a text output, suffix score offers higher quality estimates. As a result, when sampling multiple outputs (8 samples), suffix score can accurately select the decoded candidate with the highest quality, which greatly improves performance. In contrast, the LM score stays almost unchanged when increasing the number of samples, showing that it is less helpful.

5.3 Human Evaluation

In Table 4 we show the human evaluation results that align with the auto metric results shown in Table 2. The inter-annotator agreements, quantified using the Fleiss kappa coefficient (Fleiss 1971), demonstrate the reliability of the evaluations. There is a huge gain from SFT after heuristic RL. With 40% server-side calls (GPT 3.5), the model gains another big performance boost very close to the server-side model. Our SFT model’s superior performance compared to Alpaca-PaLM 2 highlights the benefits of our training data over the Alpaca dataset. For coherence and fluency, all models achieve scores over 1.93 with strong ability to generate unambiguous and logic coherent text. The results suggest that the automatic metrics and human evaluation are quite consistent.

6 Conclusion

In this paper we provided an effective approach to build an on-device rewrite model that does not rely on human-labeled data or preference data. We introduced MESSAGE_REWRITE_EVAL, a new human-labeled benchmark that focuses on text rewriting for messages through natural language instructions. We also developed an efficient and effective cascading approach using distillation of critiques. Through experiments, from both automatic metrics and human evaluations, we demonstrated that our on-device model outperforms the current state-of-the-art models in text rewriting despite having a much smaller size. Furthermore, cascading our model with the server side model can further boost its performance.

Table 4: Human Evaluation Results.

<table>
<thead>
<tr>
<th></th>
<th>Instruction</th>
<th>Success</th>
<th>Content</th>
<th>Factuality</th>
<th>Coherence</th>
<th>Fluency</th>
<th>AVG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agreement</td>
<td></td>
<td>0.620</td>
<td>0.748</td>
<td>0.687</td>
<td>0.714</td>
<td>0.710</td>
<td>0.696</td>
</tr>
<tr>
<td>InsGPT</td>
<td>1.780</td>
<td>1.933</td>
<td>1.924</td>
<td>1.979</td>
<td>1.969</td>
<td>1.917</td>
<td>1.917</td>
</tr>
<tr>
<td>Alpaca-PaLM 2</td>
<td>1.193</td>
<td>1.569</td>
<td>1.559</td>
<td>1.937</td>
<td>1.939</td>
<td>1.639</td>
<td>1.917</td>
</tr>
<tr>
<td>SFT</td>
<td>1.492</td>
<td>1.767</td>
<td>1.770</td>
<td>1.967</td>
<td>1.959</td>
<td>1.959</td>
<td>1.971</td>
</tr>
<tr>
<td>SFT + heuristic RL</td>
<td>1.674</td>
<td>1.881</td>
<td>1.853</td>
<td>1.965</td>
<td>1.959</td>
<td>1.867</td>
<td>1.917</td>
</tr>
<tr>
<td>Ours + InsGPT (40% server calls)</td>
<td>1.777</td>
<td>1.932</td>
<td>1.919</td>
<td>1.977</td>
<td>1.970</td>
<td>1.915</td>
<td>1.915</td>
</tr>
</tbody>
</table>

Figure 3: Comparing Suffix Score with LM scores when cascading our model with InsGPT.
7 Limitations

Our paper experiments is based on PALM 2, whose technique details is not open sourced. Thus we can only share a rough and relative size compared to all baselines but can not disclose the exact number of parameters. Besides the authors’ affiliation is not permitted to run LLaMA 2 models due to Meta’s license, thus we can not disclose its metrics as our baselines.

8 Ethical Discussion

Our work does not collect any user information nor produces any harmful output. We mention it helps improving privacy as on-device model does local inference and thus reduce the chance of privacy leaking.

References


Qinbin Li, Zeyi Wen, Zhaomin Wu, Sixu Hu, Naibo Wang, Yuan Li, Xu Liu, and Bingsheng He. 2021. A survey on federated learning systems: Vision, hype and reality for data privacy and protection. *IEEE Transactions on Knowledge and Data Engineering.*


A Appendix

A.1 MESSAGE REWRITE EVAL Data

Statistics of the MESSAGE REWRITE EVAL are located in Table 5. For every task and the complete dataset, we offer the following details: sample counts; the average word length for instruction (Ins), source (Sou), and target (Tar); the average length ratio (Len Ra) of the target over the source; and the Edit Ratio (Edit Ra) refer to Section Automatic Evaluation Metrics. All these statistical measurements are based on words. Additionally, NLI scores between the source and the golden target are calculated in Table 5. For every task and the complete dataset, we offer the following details: sample counts; the average word length for instruction (Ins), source (Sou), and target (Tar); the average length ratio (Len Ra) of the target over the source; and the Edit Ratio (Edit Ra) refer to Section Automatic Evaluation Metrics. All these statistical measurements are based on words. Additionally, NLI scores between the source and the golden target are provided:

- Content should be preserved in target from source.
- For certain rewrite task, the target should follow the requirement in the instruction.
- Formalize: the target should be more formal compared to source including: (1) formal vocabulary, (2) impersonal expression and (3) standard grammatical forms.
- Shorten: the target is simpler, more concise compared to source preserving the tone and format from the source.
- Elaborate: the target extend the source with more relevant information and ideas but the same tone and format as the source. The relevant information should not be made up facts.
- Paraphrase: the target changes the wording of the source while preserving the content, tone, format and verbosity.
- Proofread: the target fixes the grammar and wording errors in the source text.

A.3 EditEval Dataset

According to EditEval license page\(^3\), it is permitted with the following: Commercial use, Modification, Distribution, and Private use.

The rewrite task and dataset information in EditEval benchmark can be found in Table 7. The two datasets for Updating task are paragraph level, while the rest datasets are all sentence level.

<table>
<thead>
<tr>
<th>Task</th>
<th>Instruction Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Formalize</td>
<td>Make the text formal. Make this sentence more formal. Rewrite this sentence in a more formal way.</td>
</tr>
<tr>
<td>Shorten</td>
<td>Make the text more concise. Rewrite this text in concise language. Make the text shorter. Make this sound more concise</td>
</tr>
<tr>
<td>Elaborate</td>
<td>Make this more verbose. Expand this text. Rephrase this sentence in a more expand style. Make the text more elaborated.</td>
</tr>
<tr>
<td>Paraphrase</td>
<td>Rewrite this sentence. Rephrase the text. Paraphrase the following text. Rewrite, reword and reorganize. way.</td>
</tr>
<tr>
<td>Proofread</td>
<td>Fix the grammar error or spelling error of the following text. Correct the following sentence if there is any spelling or grammar error. Please proofread this sentence.</td>
</tr>
</tbody>
</table>

Table 6: The instruction samples for each task of MESSAGE REWRITE EVAL.

<table>
<thead>
<tr>
<th>Task</th>
<th>Dataset</th>
<th>Abbrev.</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fluency</td>
<td>JFLEG</td>
<td>JFL</td>
<td>747</td>
</tr>
<tr>
<td>Fluency</td>
<td>ITERATOR</td>
<td>ITRefl</td>
<td>203</td>
</tr>
<tr>
<td>Clarity</td>
<td>ASSET</td>
<td>AST</td>
<td>359</td>
</tr>
<tr>
<td>Coherence</td>
<td>ITERATOR</td>
<td>ITRefla</td>
<td>342</td>
</tr>
<tr>
<td>Simplification</td>
<td>TurkCorpus</td>
<td>TRK</td>
<td>359</td>
</tr>
<tr>
<td>Simplification</td>
<td>Neutralization</td>
<td>WNC</td>
<td>1000</td>
</tr>
<tr>
<td>Paraphrasing</td>
<td>WFR</td>
<td>FRU</td>
<td>914</td>
</tr>
<tr>
<td>Updating</td>
<td>WAFER-INSERT</td>
<td>WFI</td>
<td>4565</td>
</tr>
</tbody>
</table>

Table 7: EditEval Dataset Statistics

---

\(^3\)https://github.com/facebookresearch/ EditEval/blob/main/LICENSE
A.4 Hyper-parameter Setting

During supervised finetuning, SFT, we use 8 Tensor Processing Units (TPU) V3 chips for fine-tuning. The batch size is 64, and the maximum training step is 30000. We use the Adafactor optimizer (Shazeer and Stern, 2018) with a learning rate of 0.003. Both the input and output sequence lengths are set to 1024 tokens. The training dropout rate is 0.05. For reinforcement learning, we compute the heuristic reward with parameters in 8. We use same setup as fine-tuning except that the training step is 3000. During inference, the temperature is set to 0.5. Unless specifically noted, we use sampling decoding with sample number 8 for our experiments.

A.5 Training Data Stats

We share the detailed training data stats in Table 9. We splitted the data 8:1:1 as Train:Eval:Text during the training.

A.6 Training Data Samples

In Table 10 We share some samples from our training dataset following the method described in Section 3.1.1.

A.7 MESSAGE/REWRITE/EVAL Samples with Model Outputs

We share some samples from our MESSAGE/REWRITE/EVAL in Table 11. At the same table, we share the outputs from two models, both finetuned on PaLM2 XXS. The first one is finetuned with Alpaca dataset (Alpaca PaLM2) and the second one is finetune with our synthetic dataset, the statistic numbers of these two models can be found in Table 1 and Table 2.

A.8 Human Evaluation Guideline

We follow the same human evaluation guideline as the RewriteLM paper (Shu et al., 2023).

<table>
<thead>
<tr>
<th>Rewrites</th>
<th>NLI $\sigma_1$</th>
<th>Reverse NLI $\sigma_2$</th>
<th>Length Ratio $\sigma_3$</th>
<th>Edit Dist $\sigma_4$</th>
<th>Ngram Freq $\sigma_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Formalize</td>
<td>1.0</td>
<td>1.0</td>
<td>0.0</td>
<td>0.4</td>
<td>1.0</td>
</tr>
<tr>
<td>Shorten</td>
<td>1.0</td>
<td>0.4</td>
<td>-0.2</td>
<td>0.4</td>
<td>1.0</td>
</tr>
<tr>
<td>Elaborate</td>
<td>0.4</td>
<td>1.0</td>
<td>0.5</td>
<td>0.4</td>
<td>1.0</td>
</tr>
<tr>
<td>Paraphrase</td>
<td>1.0</td>
<td>1.0</td>
<td>0.0</td>
<td>0.4</td>
<td>1.0</td>
</tr>
<tr>
<td>Proofread</td>
<td>1.0</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
<td>1.0</td>
</tr>
</tbody>
</table>

Table 8: The choice of $\sigma_i$. For formalize and paraphrase, the length ratio is not considered important while for proofread/grammar correction, we apply the additional logic that the length ratio should be close to 1.

**Instruction Success:** The ability of the model to adhere to the given instruction is evaluated in this criterion. It is:

- Score 2 (Fully/Mostly Followed): if the model output entirely adheres to the provided instructions, demonstrating a clear understanding and implementation of the given task. Or the output mostly adheres to the instructions, with minor deviations or errors.
- Score 1 (Partially Followed): if the model output shows some adherence to the instructions but deviates significantly in certain aspects or fails to completely implement them, leading to partial fulfillment of the task.
- Score 0 (Not Followed/Mostly Ignored): if the model output largely ignores the provided instructions, making it evident that the task has not been understood or implemented properly. Or despite some slight adherence, the output largely deviates from the intended task as per the instructions.

**Content Preservation:** The essential content and meaning of the reference is preserved in the rewrite, independent of its style or the quality of the writing. It is:

- Score 2 (Fully/Mostly Preserved): if the rewrite is an excellent representation of the content in the reference, with no omissions. Or the rewrite mostly matches the content of the reference, but one or two elements of the meaning have been lost.
- Score 1 (Half Preserved): if some of the content is present in the rewrite but approximately the same amount is missing.
- Score 0 (Not Preserved/Mostly Lost): if the rewrite is entirely unrelated to the reference. Or despite some slight similarities, the rewrite is hard to recognize as being based on the reference.

**Factuality:** The rewrite only provides as much information as is present in the reference, without adding anything. It is not misleading and does not make any false statements (unless these were also present in the reference).

- Score 2 (Fully/Mostly faithful): Everything in the rewrite is grounded in the reference. Or the
The meeting will be at 8 p.m.

70 per cent of the total market share.

I can help you, my love.

If someone is an enemy of my enemy, then that person is my friend.

The weather is sunny. The high temperature is near 15°C. Winds SSW at 10 to 15 km/h.

The enemy of my enemy is my friend.

The conference will commence at eight in the evening.

70% of the market.

The weather is sunny. The high temperature is near 15°C. Wind comes from SSW at 10 to 15 km/h.

I can help you, my love.

Formalize: Make this sound more formal

Elaborate: Elaborate the sentences.

Paraphrase: Rephrase the text

Proofread: Please proofread this sentence

Rewrite says something that is not mentioned in the reference or contradicts the reference, but it is not an important addition or it is hard to say whether the statement is true or false.

- Score 1 (Partly faithful): The rewrite adds significant factual statements to the reference. These may be inaccurate or otherwise not based on the reference, but do not entirely undermine the faithfulness of the rewrite as a whole.

- Score 0 (Not/Slightly faithful): The rewrite is mostly wrong, made up, or contradicts what is in the reference text.”

Coherence: The rewrite is coherent if, when read by itself (without checking against the reference), it’s easy to understand, non-ambiguous, and logically coherent. On the other hand, the rewrite is not coherent if it’s difficult to understand what it is trying to say.

- Score 2 (Good): The whole of the rewrite is mostly fluent and easy to read, independent of any reference content. Some specific parts of the rewrite could be more naturally phrased, but overall it is fairly clear and easy to understand.

- Score 1 (Neutral): The rewrite is comprehensible, though not on the first read or only with some effort.

- Score 0 (Bad): The rewrite is very hard to understand, except by checking against the reference.

Fluency: The rewrite is considered fluent if it follows all the rules of its language, including spelling, grammar and punctuation. It reads as though it was written by someone who speaks English as their first language.

- Score 2 (Flawless/Good): The rewrite is grammatically correct, contains no spelling errors, and follows all other linguistic rules. An average English speaker would not see anything that looks “wrong”. Or there are just one or two linguistic errors or non-standard formulations, but nothing serious.

- Score 1 (Flawed): The rewrite contains a number of errors of different types, but these errors, even when taken together, do not make the text significantly harder to understand.

- Score 0 (Poor): The rewrite contains a large number of errors, so that some sections of the text are hard to understand, but other parts are more manageable.

A.9 On-device Inference Metrics

To demonstrate the effectiveness of running our models using limited resources, we obtain benchmark numbers on popular mobile phones to obtain two primary metrics: Inference Latency per Token, measured in milliseconds, and Memory Consumption, quantified in gigabytes during model operation. We introduce an inference engine utilizing OpenCL that harnesses the computational capabilities of on-device GPUs. We adopt similar optimizations reported in (Chen et al., 2023b) and further devise special kernels tailored for our
<table>
<thead>
<tr>
<th>Comment</th>
<th>Source</th>
<th>Alpaca PaLM2</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Formalize</strong></td>
<td>Make this sound more formal.</td>
<td>He was sleepy.</td>
<td>He had been sleeping for a long time.</td>
</tr>
<tr>
<td><strong>Shorten</strong></td>
<td>Rewrite this text more concisely.</td>
<td>I am not in the mood of going to dinner.</td>
<td>I have no desire to eat dinner.</td>
</tr>
<tr>
<td><strong>Elaborate</strong></td>
<td>Make this more verbose.</td>
<td>I’ll be in Lisbon in May. I’ll be in Lisbon during the peak tourist season, which runs from May to August.</td>
<td>I’ll be in Lisbon in May. I’m really looking forward to it!</td>
</tr>
<tr>
<td><strong>Paraphrase</strong></td>
<td>Rewrite the text another way.</td>
<td>No one wants to come with me tonight.</td>
<td>Not a single person is willing to join me tonight.</td>
</tr>
<tr>
<td><strong>Proofread</strong></td>
<td>Please proofread this sentence.</td>
<td>It was all mess...</td>
<td>It was all right.</td>
</tr>
</tbody>
</table>

Table 11: Sample from MESSAGE WRITE EVAL, and the outputs from models trained by Alpaca vs our training dataset.

<table>
<thead>
<tr>
<th></th>
<th>S23</th>
<th>Pixel 7</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>8-bits</td>
<td>4-bits</td>
</tr>
<tr>
<td>P. Parsing (ms)</td>
<td>1.2</td>
<td>1.2</td>
</tr>
<tr>
<td>Decoding (ms)</td>
<td>48.0</td>
<td>35.0</td>
</tr>
<tr>
<td>Memory (Gb)</td>
<td>1.6</td>
<td>0.9</td>
</tr>
</tbody>
</table>

Table 12: Benchmark results of our model. The average latency per token for the prompt parsing and decoding phases are reported in milliseconds. The last row shows the total memory consumption in gigabytes.

on-device Instruct-oriented models. To accommodate models within constrained memory capacities, we employ 8-bit post-training quantization as the standard setting for reporting quality metrics. The latency/memory numbers of both 8- and lower-bit quantized model are presented to compare with commonly adopted configurations. We note that the quality implication of lower-bit quantization and quantization-aware training is beyond the scope of this paper.

Table 12 presents the performance benchmarks of our inference engine on both the Samsung S23 and Pixel 7 Pro. These evaluations were conducted using 1024 input tokens and decoding over 100 tokens. Results for both 8-bit and 4-bit quantized models are provided. It is noteworthy that, on the S23, the mean latency per token during the prompt parsing phase is 1.2ms (equivalent to >800 tokens/second), with the decoding latency being 35ms (29 tokens/second). To the best of our knowledge, the latency of our model on a cell phone is greatly faster than the reported numbers (i.e. 18 - 22 tokens/second) benchmarked on Macbook M1 Pro 32GB Ram for a 7B Llama model with 4-bits quantization (Gerganov, 2023).

A.10 Impact of cascading to the inference latency
We profiled the latency by comparing the cascading method with the InsGPT model in Table 13. As most LLMs are hosted on server, we use the InsGPT as base for evaluation, and we can achieve over 96% performance with 74% less latency or over 97.5% performance with 52% less latency.

<table>
<thead>
<tr>
<th></th>
<th>Sari</th>
<th>Bleu</th>
<th>Update-R</th>
<th>Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>InsGPT</td>
<td>51.14</td>
<td>35</td>
<td>58.91</td>
<td>100%</td>
</tr>
<tr>
<td>Our Model</td>
<td>48.6/95.0%</td>
<td>32.43/92.7%</td>
<td>55.72/94.6%</td>
<td>13%</td>
</tr>
<tr>
<td>15% InsGPT</td>
<td>49.03/95.9%</td>
<td>33.76/96.5%</td>
<td>57.41/97.5%</td>
<td>26%</td>
</tr>
<tr>
<td>40% InsGPT</td>
<td>49.87/97.5%</td>
<td>34.59/98.8%</td>
<td>58.87/99.9%</td>
<td>47.8%</td>
</tr>
</tbody>
</table>

Table 13: Quality vs Latency Tradeoff.

A.11 Few-shot Prompts for Pre-trained LLM Inference
The few-shot prompt in Table 14 is used to facilitate the pre-trained LLM inference for text rewriting tasks. This is only used for evaluation of our baseline Pre-trained LLMs.

A.12 Few-shot CoT Prompts for LLM Critique
The few-shot CoT prompts are used in order to leverage the off-shelf LLMs capability to judge the quality of rewrite. The prompt sample is in Table 15.
Table 14: The few-shot CoT prompts for pre-trained LLM inference.
### Table 15: The few-shot CoT prompt samples for LLM critique. “GOOD” indicates the response is following the instruction of the comment to rewrite the source (context).

<table>
<thead>
<tr>
<th>Comment</th>
<th>Context</th>
<th>Response</th>
<th>Choose (GOOD) or (BAD)</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Make the text more formal.</td>
<td>Do we want to go to movie now? This one should be pretty good.</td>
<td>Want to go to movie? It should be a great one.</td>
<td>BAD</td>
<td>Response is not more formal than Context.</td>
</tr>
<tr>
<td>Simplify the text.</td>
<td>Ric Flair had a match against Mitch of the Spirit Squad. All five members of the Spirit Squad were present, so Flair brought out Rowdy Roddy Piper, Money Inc., and Arn Anderson as his backup. Flair’s allies kept the Squad in check, enabling Flair to win the match.</td>
<td>Ric Flair defeated Mitch of the Spirit Squad with help from Rowdy Roddy Piper, Money Inc., and Arn Anderson.</td>
<td>GOOD</td>
<td>Response is shorter than Context. Response preserves overall meaning.</td>
</tr>
<tr>
<td>Elaborate the following text.</td>
<td>Iuter X Vanguard collaboration T-shirt by Giorgio Di Salvo. Octopus print. All Iuter apparel is Made in Italy.</td>
<td>This T-shirt is part of the collaboration between Iuter and Vanguard. It is designed by Giorgio Di Salvo and features an octopus print. All Iuter apparel is Made in Italy.</td>
<td>GOOD</td>
<td>Response rephrases and elaborates the context with preserved meaning.</td>
</tr>
<tr>
<td>Paraphrase the source text.</td>
<td>He likes the dogs a lot, according to his parents.</td>
<td>He is fond of the dogs.</td>
<td>BAD</td>
<td>Response did not preserve all the meaning of Context. The fact &quot;according to his parents&quot; is missing in Response.</td>
</tr>
<tr>
<td>Fix the grammar and spelling error if there is any.</td>
<td>Native is very fortunate.</td>
<td>Native people are very fortunate.</td>
<td>GOOD</td>
<td>Response fix the grammar errors in the Context.</td>
</tr>
<tr>
<td>{comment}</td>
<td>{input}</td>
<td>{output_best}</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>