

Training Language Model to Critique for Better Refinement

Tianshu Yu^{1,2*}, Chao Xiang^{1*}, Mingchuan Yang¹, Pei Ke³, Bosi Wen^{2†}, Cunxiang Wang^{4,5},
Jiale Cheng^{2‡}, Li Zhang¹, Xinyu Mu¹, Chuxiong Sun¹, Minlie Huang^{2‡}

¹China Telecom Research Institute

²The Conversational Artificial Intelligence (CoAI) Group, Tsinghua University

³University of Electronic Science and Technology of China

⁴The Knowledge Engineering Group (KEG), Tsinghua University

⁵Zhipu AI

dailyyulun@gmail.com aihuang@tsinghua.edu.cn

Abstract

Large language models (LLMs) have demonstrated remarkable evaluation and critique capabilities, providing insightful feedback and identifying flaws in various tasks. However, limited research has explored which types of critiques are most effective for improving model responses or how to generate such critiques. To address this gap, we introduce **Refinement-oriented Critique Optimization (RCO)**, a novel framework designed to train critic models using refinement signals. RCO uses a feedback loop where critiques, generated by the critic model, guide the actor model in refining its responses. The critique utility (CU) quantifies the effectiveness of these refinements, serving as the reward signal for training the critic model. By focusing on critiques that lead to better refinements, RCO eliminates the need for direct critique preference assessment, ensuring that critiques driving meaningful improvements are rewarded. We evaluate RCO across five tasks, i.e., dialog generation, summarization, question answering, mathematical reasoning, and code generation, and show that it significantly outperforms traditional methods and open-source models in terms of critique quality and refinement outcomes. Our contributions include the introduction of RCO, a novel supervision scheme based on refined response preferences, and comprehensive experimental results that highlight the method’s effectiveness in enhancing LLM critique-refinement loops.¹

1 Introduction

The critique ability, defined as the capacity to identify and refine flaws in responses, is crucial for improving the reliability of automatic evaluation and

* Equal contribution.

† This work was conducted when Bosi Wen and Jiale Cheng were interning at Zhipu AI.

‡ Corresponding author.

¹We release the code and data at <https://github.com/publicstaticvo/critique>.

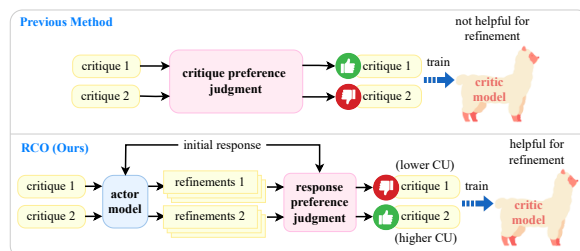


Figure 1: A comparison between previous methods, which train critic models via direct critique preference judgment and are not helpful for refinement, and our method, which train critic models by converting preferences of refined responses into critique reward values and is helpful for refinement.

enabling the self-improvement of large language models (LLMs) (Saunders et al., 2022; Scheurer et al., 2023). Enhancing this ability is challenging, as it requires LLMs to not only detect errors but also engage in a deeper analysis of user queries and responses, extending beyond superficial criticism (Zheng et al., 2023b; Kim et al., 2024). Recent studies have sought to address this challenge of LLMs via human-curated critique datasets and alignment strategies like supervised fine-tuning (SFT) (Cui et al., 2024; Ke et al., 2024) and reinforcement learning (Akyurek et al., 2023; McAleese et al., 2024), yielding promising results in identifying and rectifying shortcomings across various tasks.

However, current methods primarily train models to generate critiques for evaluation purposes, rather than linking critique to refinement (Li et al., 2024b). This disjointed approach fails to connect error identification with output improvement, limiting the practical value of critiques. While the critique-refinement paradigm is explored in LLM alignment (Madaan et al., 2023; Wadhwa et al., 2024) and reasoning (Chen et al., 2024; Wu et al., 2024), existing works lack systematic investigation on how critique enhancements can lead to

meaningful refinements in responses. Without this link, assessing critique quality remains challenging (Sun et al., 2024). Addressing this gap is essential for advancing both LLM evaluation and self-improvement capabilities.

To address these challenges, we introduce Refinement-oriented Critique Optimization (RCO), a novel training paradigm for critic models that uses critique utility (CU) as a reward signal, which is calculated by comparing the refined responses to the initial ones. This approach encourages the generated critiques to drive substantial improvements in the output, creating a more effective critique-refinement loop. Specifically, RCO feeds critiques and initial responses into the actor model and prompts it to generate multiple refined responses. CU is then quantified as the proportion of refinements preferred over the initial response, which serves as the reward signal for training the critic model. By focusing on the refinement outcomes, RCO eliminates the need for direct critique preference assessment and ensures that critiques leading to better refinements are rewarded. We evaluate RCO across five tasks, i.e., dialog generation, summarization, question answering, mathematical reasoning, and code generation, using baseline models, models trained with critique preferences, and advanced open-source models. Experimental results demonstrate that RCO significantly enhances critic model performance, outperforming existing methods across multiple benchmarks.

Our contributions are threefold: (1) We propose RCO, a method that prioritizes critiques that facilitate effective refinement of actor model responses, addressing limitations of previous approaches. (2) We introduce a novel supervision scheme based on refined response preferences, eliminating the need for directly assessing the quality of critiques while rewarding critiques that lead to meaningful improvements. (3) We rigorously evaluate RCO across diverse tasks, showing substantial improvements in critique quality and refinement capabilities, and provide an in-depth analysis of the method’s impact.

2 Related Work

Critique Ability of LLMs The rapid advancement of large language models (LLMs) has highlighted the need to enhance their critique capabilities. LLMs like GPT-4 (Achiam et al., 2023) have proven effective as evaluators (Zheng et al., 2023a;

Li et al., 2024a; Cao et al., 2024), but their API-based access limits widespread application. To address cost and stability issues, researchers have fine-tuned open-source models using critique data from these models. Despite these efforts, challenges in critique task complexity remain. Recent work includes Murugadoss et al. (2024), which uses prompt engineering to create critique metrics, and Verga et al. (2024), which proposes using multiple LLMs to mitigate bias. Ke et al. (2024) generates a golden critique dataset from model and reference response pairs, while Lan et al. (2024a) employs a multi-agent framework for preference-based critique data collection. In contrast, our method improves critique quality by automatically obtaining preferences of refined responses to calculate critique utility as reward signals to train critic models.

Preference-Based Reinforcement Learning Reinforcement Learning from Human Feedback (RLHF) (Ziegler et al., 2019) is commonly used to guide LLMs towards human-preferred responses. Scheurer et al. (2023) uses RLHF to train reward models based on human-annotated pairwise comparisons. Recent methods, such as CriticGPT (McAleese et al., 2024), apply RLHF to enhance critique abilities by training critic models on human-identified errors in code generation. Similarly, Wang et al. (2024b) collects preference critique pairs by comparing LLM-generated responses with human-annotated scores. These approaches are limited by the high cost and uncertain quality of human annotations (Sun et al., 2024). In contrast, our approach reduces reliance on human annotations, providing a clear standard for good critiques that effectively guide actor model refinements.

3 Methodology

The overview of RCO is illustrated in Figure 2. Our approach begins with a dataset \mathcal{D} , where each sample $\mathbf{X} = (x, y_0)$ consists of a prompt x and its corresponding initial response y_0 , with y_0 being generated by an actor model $\pi(y_0|x)$. Following this, a base critic model $p(c|y_0, x)$ is employed to generate N distinct critiques, denoted as c_1, c_2, \dots, c_N , for the initial response y_0 . Each critique c_i is then used as input to the actor model π , which produces M distinct refined responses, $y_{i1}, y_{i2}, \dots, y_{iM}$. The distribution used to sample these refined responses is represented as $\pi_{c_i}(y|c_i, y_0, x)$.

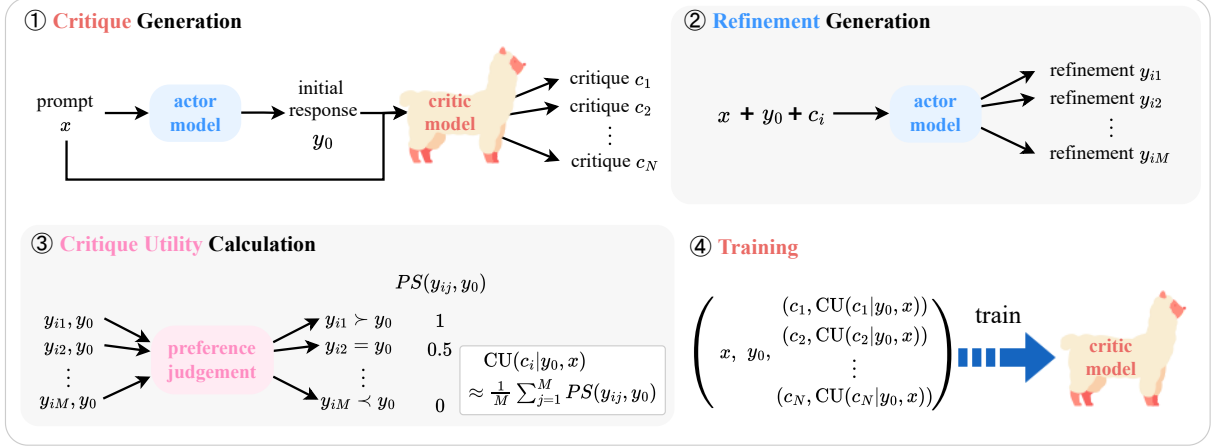


Figure 2: The illustration of our method RCO, describing our data collection and training process. We first generate critiques with the critic model, then sample multiple refined responses with the actor model. Subsequently, we calculate the critique utility of each critique with preference judgment on refined and initial responses, which serves as reward signal of critic model training.

Critique Utility. As highlighted in Section 1, the critique process is increasingly integrated into critique-refinement workflows, requiring supervision from a refinement perspective. Inspired this, we propose to use critique utility (CU) as the reward signal of RCO. The central idea of CU is that a high-quality critique should naturally lead to a better refinement, serving as a catalyst for self-improvement (Sun et al., 2024). Specifically, CU for a critique c_i is defined as the probability where a response y , sampled from $\pi_{c_i}(y|c_i, y_0, x)$, is preferred over the initial response y_0 :

$$\text{CU}(c_i|y_0, x) = P(y \succ y_0 | y \sim \pi_{c_i}) \quad (1)$$

To estimate $\text{CU}(c_i|y_0, x)$, we approximate it using the following sampling-based approach, by defining a preference score (PS) for the refined response y_{ij} and the initial response y_0 :

$$\text{CU}(c_i|y_0, x) \approx \frac{1}{M} \sum_{j=1}^M \text{PS}(y_{ij}, y_0) \quad (2)$$

In this equation, the preference score $\text{PS}(y_{ij}, y_0)$ is determined by a judge model based on the following criteria: $\text{PS}(y_{ij}, y_0) = 1$ if the refined response y_{ij} is preferred over the initial response y_0 , $\text{PS}(y_{ij}, y_0) = 0.5$ if both responses are considered equally good, and $\text{PS}(y_{ij}, y_0) = 0$ if y_0 is preferred over y_{ij} . This process is repeated for each critique c_i .

Training Objective Derivation. The goal of RCO is to reward the generated critique by critique

utility. We derive our training objective starting from the following optimization problem:

$$\max_{\theta} \mathbb{E}_{c \sim p_{\theta}} [R(c|y_0, x)] - \beta \mathbb{D}_{\text{KL}} [p_{\theta}(c|y_0, x) \| p(c|y_0, x)] \quad (3)$$

where $R(c|y_0, x)$ is the reward function of the critique c given the prompt x and initial response y_0 . Since we use critique utility as the reward signal, we substitute $R(c|y_0, x)$ with $\text{CU}(c|y_0, x)$ for the rest of the derivation. According to Rafailov et al. (2023), the optimal solution of Eq.3 takes the following form:

$$p^*(c|y_0, x) = \frac{p(c|y_0, x) \exp\left(\frac{1}{\beta} \text{CU}(c|y_0, x)\right)}{Z_{\beta}(y_0, x)} \quad (4)$$

where $Z_{\beta}(y_0, x)$ is the regularization term. With some algebra, we obtain:

$$\frac{1}{\beta} \text{CU}(c|y_0, x) = \log \frac{p^*(c|y_0, x)}{p(c|y_0, x)} + \log Z_{\beta}(y_0, x) \quad (5)$$

Note that the regularization term $Z_{\beta}(y_0, x)$ can be approximated via sampling in our method:

$$\begin{aligned} Z_{\beta}(y_0, x) &= \mathbb{E}_{c \sim p(c|y_0, x)} \exp\left(\frac{1}{\beta} \text{CU}(c|y_0, x)\right) \\ &\approx \frac{1}{N} \sum_{i=1}^N \exp\left(\frac{1}{\beta} \text{CU}(c_i|y_0, x)\right) \end{aligned} \quad (6)$$

Finally, inspired by Direct Reward Optimization (Richemond et al., 2024), we utilize a minimum square error objective as the training objective of our method, substituting $p^*(c_i|y_0, x)$ with

$p_\theta(c_i|y_0, x)$, the model to be optimized:

$$\mathcal{L}_{\text{RCO}} = \mathbb{E}_{(x, y_0) \in \mathcal{D}} \left[\frac{1}{2N} \sum_{i=1}^N \left(\log \frac{p_\theta(c_i|y_0, x)}{p(c_i|y_0, x)} + \log Z_\beta(y_0, x) - \frac{1}{\beta} \text{CU}(c_i|y_0, x) \right)^2 \right] \quad (7)$$

Compared to traditional preference-based learning methods such as DPO (Rafailov et al., 2023), our training objective leverages scalar reward values more effectively, enabling the critic model to learn a more nuanced reward representation. This advantage allows our method to better capture complex reward structures, thereby improving overall model performance.

3.1 Training Data Collection

Dataset Overview. Our method begins with the collection of the prompt dataset \mathcal{D} consisting of five tasks: dialog generation, summarization, question answering, mathematical reasoning, and code generation. These tasks are sourced from 14 datasets, as outlined in Table 6, with a total of 10,000 unique prompts collected for the experiment.

Collection of Initial Responses. Four actor models, i.e., *LLaMA-2-7B-Chat*, *LLaMA-2-13B-Chat*, *LLaMA-2-70B-Chat*, and *LLaMA-3-8B-Instruct* (Touvron et al., 2023; Dubey et al., 2024), are used to generate responses for the 10,000 prompts, yielding 40,000 unique responses. Among these, 8,000 responses are selected per model, ensuring 1,600 responses per task and 2,000 per actor model.

Critique Generation. To generate critiques, we employ five base critic models: *LLaMA-2-7B-Chat*, *LLaMA-2-13B-Chat*, *LLaMA-3-8B-Instruct*, *AutoJ-13B* (Li et al., 2023), and *UltraCM-13B* (Cui et al., 2024). Each model generates $N = 4$ critiques for every initial response y_0 in the training set, ensuring a more accurate approximation of the regularization term $Z_\beta(y_0, x)$ and improving the training procedure.

Refinement Generation Based on Critiques. In the refinement phase, the actor model that generates the initial response y_0 refines its output based on the critique it receives. For each critique c_i , the actor model generates $M = 5$ distinct refined responses, which enables a better approximation of critique utility.

Critique Utility Calculation. As shown in Figure 2, critique utility $\text{CU}(c_i|y_0, x)$ is calculated as the reward signal for each critique, following the method in Section 3. We use the *Qwen-2.5-72B-Instruct* (Team, 2024) model to evaluate the preference of each refined response y_{ij} relative to its initial response y_0 . To avoid positional bias, we alternate the positions of the refined and initial responses and perform response preference judgment again (Wang et al., 2024a). The critique utility for each critique c_i is computed as the average of 10 individual preferences ($2M = 10$). This process ensures robust evaluation while mitigating potential bias, with data collection prompts detailed in Appendix A.

4 Experiment Settings

In this section, we describe the benchmarks, evaluation metrics, baselines, and experimental results that assess the performance of RCO.

4.1 Benchmarks and Evaluation Metrics

Critique Utility and Refinement Quality Evaluation. We create a test dataset of 2,500 prompts, sourced from 7 datasets distinct from those in the training set, as detailed in Table 7. For initial responses, we use five actor models: the four models from the training dataset and *LLaMA-3-70B-Instruct*, a larger model not used for the training dataset. After generating critiques with the critic model, we evaluate its quality by greedy sampling a single refined response from the actor model of the initial response for each critique. We use two metrics: (1) average **CU**, calculated by prompting GPT-4 to compare the preference of refined responses to initial responses, and (2) **response quality score (RQS)**, where GPT-4 rates the responses on a 1-10 scale, with higher scores indicating better quality. We provide the prompts used for evaluation in Appendix B.

Refinement Accuracy Evaluation. To quantitatively assess the impact of critique-based refinement, we evaluate refined response **accuracy** across 10 established benchmarks spanning three critical reasoning domains: (1) **Question Answering**: BBH (Suzgun et al., 2022), GPQA-Diamond (Rein et al., 2024), MMLU (Hendrycks et al., 2020), MMLU-Pro (Wang et al., 2024c), MMLU-Redux (Gema et al., 2024), and TruthfulQA (Lin et al., 2021). (2) **Mathematical Reasoning**: MATH (Hendrycks et al., 2021) and

GSM8K (Cobbe et al., 2021). (3) **Code Generation:** MBPP (Austin et al., 2021) and MBPP-Plus (Cassano et al., 2023). Our evaluation methodology follows a rigorous three-stage pipeline: (1) initial response generation using the same actor models as in CU/RQS evaluation, (2) comprehensive critique generation for all initial responses, and (3) refined response generation and evaluation. For QA and math tasks, we employ *Qwen2.5-72B-Instruct* for answer consistency checking against gold standards. Code generation tasks are evaluated using the *bigcode-project-harness* (Ben Allal et al., 2022) testing framework, which provides deterministic verification through test case execution. This multi-domain evaluation strategy ensures comprehensive assessment of our method’s generalization capability across different types of reasoning tasks while maintaining objective, verifiable measurement standards.

Pairwise Critique Evaluation. We further investigate the model’s capability as a discriminative judge through rigorous testing on RewardBench (Lambert et al., 2024), a standardized benchmark for reward model evaluation. Our evaluation protocol strictly adheres to the benchmark’s original design while adapting the judgment generation process as specified in Tables 28-32. This evaluation provides crucial insights into the model’s ability to function as a general-purpose preference judge model beyond the critique refinement paradigm.

Human Evaluation. The second benchmark combines CriticEval (Lan et al., 2024b) and CriticBench (Lin et al., 2024) datasets to evaluate the critique ability of models across tasks. Human evaluation assesses critique and refinement quality: (1) **Human preferences of critiques** compare critiques generated by our method and baselines, and (2) **Human preferences of refinements** compare refinements generated from both critiques by *LLaMA-2-7B-Chat*. We sample 200 responses, ensuring 40 responses per task and disjoint prompt sources from those used in training. Details of the human evaluation process are provided in Appendix D.

4.2 Baselines

We evaluate our method against 5 baseline types: (1) **Base critic models:** The five base models described in Section 3.1. (2) **Self-refinement:** Following prior works (Akyürek et al., 2023), we use

the actor model to directly refine its own responses in the test set and evaluate the average preference score (PS) and RQS of the refined responses. (3) **Open-source LLMs:** We compare against models such as *LLaMA-2-70B-Instruct* and *LLaMA-3-70B-Instruct*. (4) **Aligner:** *Aligner* (Ji et al., 2024) is a model-agnostic module that refines responses; we use *Aligner-7B-V1.0* and evaluate the average PS and RQS of its refined responses. (5) **Direct Preferences of Critique Optimization (DPCO):** We use *Qwen-2.5-72B-Instruct* to assess the preferences of critique pairs in the training dataset, following the process in Figure 1. For the $N = 4$ setting, we label the preferences of critique pairs (c_1, c_2) and (c_3, c_4) and train critic models using the DPO algorithm.

5 Main Results

5.1 Critique Utility and Response Quality

Our comprehensive evaluation demonstrates that RCO-trained critic models consistently outperform all baseline approaches across multiple dimensions. As shown in Table 1, RCO achieves significant improvements in both CU and RQS metrics compared to baseline models. This performance advantage holds true across all base model architectures and sizes, validating the effectiveness of our training paradigm.

Specifically, RCO-trained critic models consistently outperform baselines across all model sizes and architectures. Notably, smaller RCO-trained models (e.g., *LLaMA-2-7B-Chat*) surpass their larger counterparts (e.g., *LLaMA-2-70B-Chat*), demonstrating strong parameter efficiency. The performance gains are particularly significant in mathematical reasoning and code generation tasks. In contrast, DPCO shows only marginal improvements over base models, indicating limited effectiveness in leveraging critique signals. While DPCO performs comparably in dialog and summarization tasks, RCO maintains a clear advantage in complex reasoning domains.

5.2 Refined Response Accuracy Results

Table 2 presents comprehensive results across multiple benchmarks. Our analysis yields two important findings: (1) RCO consistently outperforms both base models and DPCO-trained variants across all benchmarks, including challenging datasets like GPQA-Diamond, MMLU-Pro, and MBPP-Plus. This demonstrates the method’s ro-

Base Model	Method	Dialog		Summ.		QA.		Math		Code		Overall	
		CU	RQS	CU	RQS	CU	RQS	CU	RQS	CU	RQS	CU	RQS
Initial Answer		–	6.23	–	6.52	–	5.33	–	3.73	–	4.42	46.9	5.25
BASELINES													
LLaMA-2-70B-Chat		82.7	7.09	68.3	7.44	88.8	6.75	62.7	4.19	59.7	5.59	72.4	6.21
LLaMA-3-70B-Instruct		82.6	7.08	87.8	7.68	86.3	6.47	76.2	4.75	78.1	6.24	82.2	6.44
Self-refinement		75.2	6.96	77.6	7.46	79.9	6.08	64.5	4.32	65.8	5.06	72.6	5.97
Aligner		47.3	6.30	50.2	6.64	47.6	5.79	51.4	3.78	38.0	4.49	46.9	5.40
OUR METHOD													
LLaMA-2-7B-Chat	Base model	83.5	7.16	63.4	7.43	87.1	6.48	60.1	4.03	59.7	5.29	70.8	6.08
	+DPCO	79.2	7.13	70.2	7.37	91.2	6.85	58.7	3.94	62.1	5.50	72.3	6.16
	+RCO (Ours)	90.4	7.28	77.4	7.59	94.3	7.28	70.7	4.45	72.5	5.86	81.1	6.49
LLaMA-2-13B-Chat	Base model	78.4	7.06	74.7	7.56	86.9	6.64	66.3	4.21	61.1	5.39	73.5	6.17
	+DPCO	76.5	6.91	79.9	7.56	86.0	6.51	56.1	3.68	69.2	5.70	73.5	6.08
	+RCO (Ours)	86.9	7.20	85.3	7.60	96.2	7.22	71.0	4.38	70.1	5.69	81.9	6.42
LLaMA-3-8B-Instruct	Base model	75.8	7.02	78.4	7.60	82.0	6.55	69.8	4.41	73.1	6.08	75.8	6.33
	+DPCO	68.7	6.71	79.0	7.30	82.4	6.76	63.8	4.26	70.5	5.32	72.9	6.07
	+RCO (Ours)	87.0	7.17	86.0	7.74	94.2	7.03	76.2	4.84	78.3	6.33	84.3	6.62
Auto-J-13B	Base model	60.0	6.83	85.6	7.60	79.1	6.50	60.4	3.85	63.1	5.22	69.7	6.00
	+DPCO	68.3	6.99	80.3	7.47	87.9	6.57	64.5	4.06	70.2	5.73	74.2	6.16
	+RCO (Ours)	77.5	7.09	83.7	7.54	91.9	7.17	68.5	4.38	72.3	5.63	78.8	6.36
UltraCM-13B	Base model	53.9	6.50	72.6	7.24	63.8	5.88	65.2	4.27	62.4	5.25	63.6	5.83
	+DPCO	69.8	6.92	85.1	7.61	84.8	6.47	62.4	3.98	67.0	5.33	73.8	6.06
	+RCO (Ours)	74.7	7.05	89.6	7.66	88.4	6.79	72.0	4.50	69.3	5.46	78.8	6.29

Table 1: Evaluation results of our method and baselines, in terms of critique utility (CU) and refinement quality score (RQS). Summ. and QA. are the shorter form of summarization and question answering tasks, respectively. The critique utility results reported in the table are multiplied by 100.

bustness in generating actionable critiques that lead to verifiable improvements. (2) The performance advantage holds across different model scales, with RCO-trained smaller models frequently matching or exceeding the capabilities of much larger base models. This suggests our training paradigm effectively distills critique generation expertise.

5.3 Pairwise Critique Ability Results

The RewardBench evaluation (Table 2, rightmost column) reveals an emergent capability: RCO-trained models achieve superior performance on pairwise preference judgment despite no explicit training for this task. This suggests that: (1) Critique generation training may develop generalizable preference modeling abilities. (2) The RCO objective aligns well with the underlying mechanisms of reward modeling. (3) There exists potential synergy between critique generation and preference learning.

5.4 Human Evaluation

In the human evaluation, we focus on the top-performing base model *LLaMA-3-8B-Instruct* for cost efficiency, comparing RCO against three baselines: its base version, *LLaMA-3-70B-Instruct*, and DPCO with. Figure 3 shows RCO’s consistent superiority in both critique and refinement quality. These results underscore the efficacy of our approach in enhancing actor model responses through structured critique generation and refinement.

Analyzing human evaluation results across different tasks reported in Table 3 we conclude that RCO excels in mathematical reasoning and code generation versus base/DPCO models, and outperforms larger models in dialogue/summarization tasks. A detailed case analysis on the reason of these distinctions is presented in Section 6.4 and Appendix F.

An intriguing observation from the human evaluation is that human preferences for critiques do not always align with the preferences for the refinements generated from those critiques. Specif-

Base Model	Method	QA.					Math			Code		Reward
		BBH	GPQA-Diamond	MMLU	MMLU-Pro	MMLU-Redux	Truthful	MATH	GSM-8K	MBPP	MBPP+	Reward-Bench
	Initial Answer	39.67	27.65	52.48	33.14	48.79	49.12	17.10	42.05	35.84	32.85	–
BASELINES												
	LLaMA-2-70B-Chat	44.19	26.77	60.02	37.59	53.55	50.23	21.75	52.50	39.00	36.72	65.9
	LLaMA-3-70B-Instruct	58.01	30.30	67.39	46.36	61.24	58.21	36.90	76.14	41.60	39.26	77.1
	Self-refinement	43.65	29.60	54.79	35.49	49.52	52.39	22.91	54.42	37.20	36.50	–
	Aligner	42.39	24.15	52.56	32.39	48.59	46.61	15.62	42.74	30.88	30.54	–
OUR METHOD												
LLaMA-2-7B-Chat	Base model	38.22	27.47	56.96	35.29	50.82	48.32	19.65	48.92	36.16	34.74	59.5
	+DPCO	45.52	28.79	60.46	37.22	53.81	50.43	20.50	55.31	36.44	35.12	55.8
	+RCO (Ours)	48.06	28.83	63.17	42.30	57.67	54.96	24.20	64.31	39.72	38.42	67.1
LLaMA-2-13B-Chat	Base model	40.30	26.49	57.33	34.93	50.23	49.81	18.13	44.90	38.24	35.05	62.0
	+DPCO	43.41	27.17	59.42	36.52	53.04	50.28	16.00	57.21	38.56	35.82	64.9
	+RCO (Ours)	47.89	28.35	62.48	43.01	57.98	55.31	23.60	65.54	39.92	37.69	68.3
LLaMA-3-8B-Instruct	Base model	45.33	25.47	59.41	37.88	53.24	54.32	25.36	58.11	39.32	37.73	69.7
	+DPCO	45.37	25.76	59.21	36.69	53.52	50.70	21.91	60.79	39.04	39.63	65.6
	+RCO (Ours)	53.51	30.06	67.74	46.23	61.30	59.58	32.31	70.97	41.08	40.34	77.6
Auto-J-13B	Base model	37.75	23.56	52.90	28.65	46.63	50.40	16.28	41.93	34.07	32.86	61.9
	+DPCO	43.22	26.61	58.47	36.21	53.88	54.72	19.95	54.68	34.93	34.01	64.2
	+RCO (Ours)	46.18	28.04	60.37	40.89	56.47	56.19	26.52	63.92	38.80	37.23	67.3
UltraCM-13B	Base model	45.93	25.96	57.19	33.73	50.90	55.98	20.85	56.27	35.40	34.46	67.8
	+DPCO	47.29	26.35	60.03	35.98	55.94	54.31	22.40	58.93	36.85	35.27	71.0
	+RCO (Ours)	49.75	27.83	64.42	40.74	57.89	55.62	25.17	63.45	38.92	37.74	71.9

Table 2: Accuracy results on multiple established benchmarks and RewardBench.

ically, while DPCO critiques are most preferred than the other two baselines, all baselines yield comparable refinement quality. This suggest that preferred critiques don’t guarantee better refinements, highlighting the inherent complexity of the critique-refinement process.

6 Analysis

6.1 Weak-to-Strong Generalization Ability

Recent studies in LLMs have highlighted the growing trend of scalable oversight (Burns et al., 2023), which requires smaller models to provide meaningful feedback to enhance the performance of larger models. To evaluate whether RCO facilitates weak-to-strong generalization, we reanalyze the experimental results presented in Table 1, displaying the CU and RPS results with respect to actor models generating the initial responses in the test dataset.

Based on the findings in Table 4, the following key observations can be made: (1) Both CU and RPS improve as the actor model increases in size and capability (from LLaMA-2 to LLaMA-3), primarily because the stronger models exhibit a greater ability to interpret critiques and follow refinement instructions. (2) When compared to self-refinement (refinement without critique), the RCO-trained models show superior performance, partic-

ularly for initial responses generated by *LLaMA-3-70B-Instruct*. In contrast, DPCO-trained models perform worse than self-refinement in this case. (3) A comparison between RCO, base models, and DPCO reveals that RCO significantly enhances the performance of larger models (70B) on refined responses. For smaller actor models, however, RCO only slightly outperforms DPCO. These findings suggest that RCO enables improved weak-to-strong generalization over DPCO and other baseline methods by supervising critique training with the outcome of refinement.

6.2 Scaling Preference Judge Model for RCO

The training process of RCO depends on the preference judgment of both refined and initial responses. As such, the accuracy of preference judgment plays a crucial role in the overall performance of RCO. In our experiment, we leverage *Qwen2.5-72B-Instruct* (Qwen) to evaluate the preferences of responses. To assess the impact of the selection of preference judge model, we investigate four distinct variations: (1) **Stronger LLMs**, such as *GPT-4o* (Achiam et al., 2023); (2) **Open-source LLM Critic Models**, such as *Skywork-Critic-LLaMA-3.1-8B* (Skywork) (Shiwen et al., 2024); (3) **Reward Models**, such as *InternLM2-7B-Reward* (InternLM) (Cai et al., 2024) and PairRM (Jiang et al.,

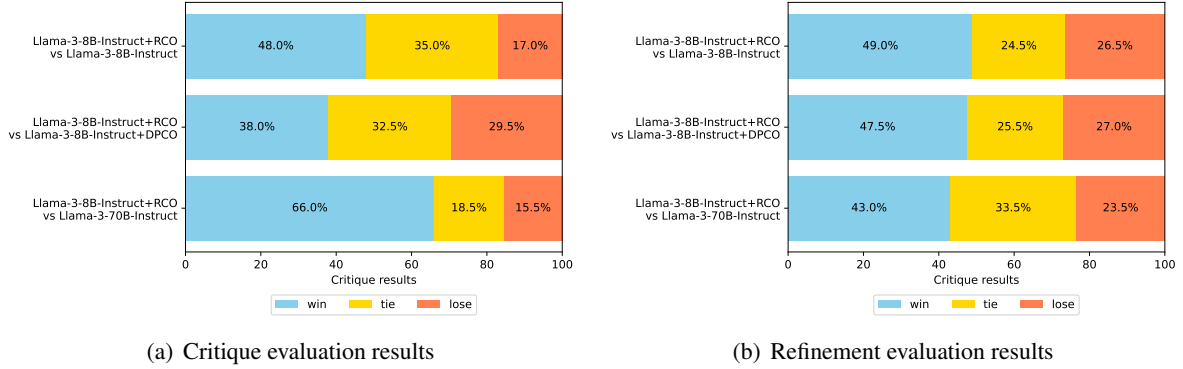


Figure 3: Visualization of human evaluation results in terms of critique quality evaluation and refinement quality evaluation.

RCO (Ours)	VS.	Task	Critique				Refinement			
			Win (%)	Tie (%)	Loss (%)	Gap (%)	Win (%)	Tie (%)	Loss (%)	Gap (%)
	LLaMa-3-8B-Instruct	Dialog	50.0	30.0	20.0	+30.0	55.0	22.5	22.5	+32.5
		Summ.	60.0	25.0	15.0	+45.0	57.5	17.5	25.0	+32.5
		QA.	42.5	35.0	22.5	+20.0	37.5	20.0	42.5	-5.0
		Math	47.5	35.0	17.5	+30.0	55.0	30.0	15.0	+40.0
		Code	40.0	50.0	10.0	+30.0	40.0	32.5	27.5	+12.5
		Overall	48.0	35.0	17.0	+31.0	49.0	24.5	26.5	+22.5
LLaMa-3-8B-Instruct+MSE	LLaMa-3-8B-Instruct+DPO	Dialog	25.0	32.5	42.5	-17.5	35.0	22.5	42.5	-7.5
		Summ.	52.5	15.0	32.5	+20.0	30.0	50.0	20.0	+10.0
		QA.	32.5	27.5	40.0	-7.5	47.5	15.0	37.5	+10.0
		Math	37.5	45.0	17.5	+20.0	57.5	22.5	20.0	+37.5
		Code	42.5	42.5	15.0	+27.5	67.5	17.5	15.0	+52.5
		Overall	38.0	32.5	29.5	+8.5	47.5	25.5	27.0	+20.5
	LLaMa-3-70B-Instruct	Dialog	80.0	12.5	7.5	+72.5	40.0	32.5	27.5	+12.5
		Summ.	75.0	15.0	10.0	+65.0	40.0	45.0	15.0	+25.0
		QA.	75.0	17.5	7.5	+67.5	62.5	15.0	22.5	+40.0
		Math	50.0	10.0	40.0	+10.0	42.5	32.5	25.0	+17.5
		Code	50.0	37.5	12.5	+37.5	30.0	42.5	27.5	+2.5
		Overall	66.0	18.5	15.5	+50.5	43.0	33.5	23.5	+19.5

Table 3: Human evaluation results across different tasks.

2023); and (4) **Self-rewarding** method (Yuan et al., 2024), in which the base critic model evaluates the preferences for refinements under its own outputs. We conduct experiments using the *LLaMA-3-8B-Instruct* as the base critic model and report the results of RCO with different preference judge models, as shown in Table 5. Additionally, we provide the accuracy of preferences for each judge model using the established pairwise judgment benchmark, RewardBench (Lambert et al., 2024).

Our findings indicate that PairRM, which has the lowest pairwise accuracy on RewardBench, results in the worst performance. However, despite achieving the best pairwise accuracy, the InternLM model does not yield the top scores in both CU and RQS. Overall, the performances of RCO training with judge models except PairRM are quite

similar, with CU scores ranging from 82% to 84% and RQS scores between 6.55 and 6.65. Notably, the self-rewarding method produces performance on par with that of stronger preference judge models, underscoring the potential of a self-supervised training paradigm.

6.3 Scaling Numbers of Critiques and Refinements

In the data construction phase, we establish $N = 4$ critiques per initial answer to ensure reliable estimation of the partition function $Z_\beta(y_0, x)$, and $M = 5$ refinements per critique for accurate Critique Utility (CU) estimation. This configuration generates $2 \times N \times M$ pairwise comparisons per training sample, presenting substantial computational demands. To investigate cost-reduction op-

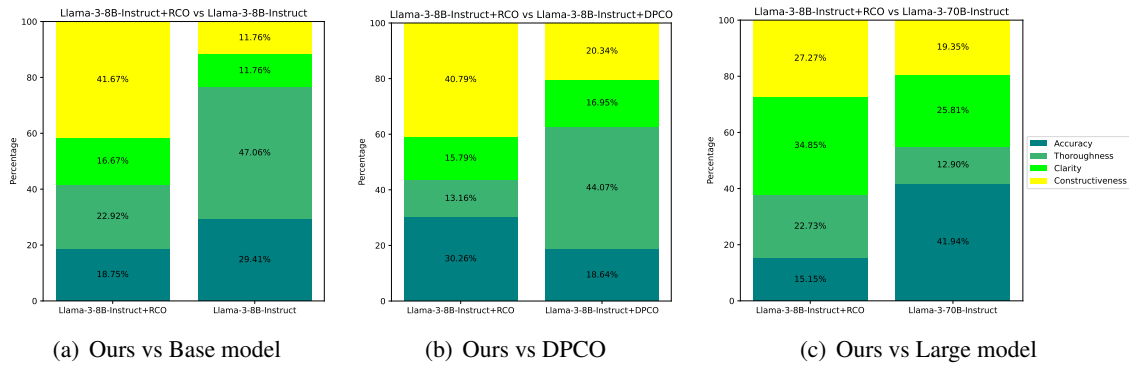


Figure 4: The percentage distribution of the reasons why each chosen critique is preferred in human evaluation.

portunities, we systematically evaluate the impact of decreasing N and M values on model performance. As shown in Figure 6, we report the CU and RQS results of two base models *LLaMA-2-7B-Chat* and *LLaMA-3-8B-Instruct* over theoretically minimal settings ($N = 2, M = 1$). Our findings reveals that (1) CU, RQS and accuracy metrics exhibit slight but consistent degradation as N or M decreases, confirming that higher sample sizes contribute to training effectiveness. (2) Remarkably, RCO maintains significant superiority over DPCO across all tested configurations, demonstrating the inherent advantages of our approach regardless of sampling parameters.

6.4 Why RCO Generate Effective Critiques for Refinement

To further explore why our method produces more effective critiques for refining actor models, we conduct an analysis asking human evaluators to select the primary reason for their preference of a given critique from the following four options: “The critique is more accurate and correct (Accuracy)”, “The critique provides a more thorough analysis of errors in the initial response (Thoroughness)”, “The critique is clearer and better structured (Clarity)” and “The critique offers constructive suggestions or provides detailed steps for revision (Constructiveness)”. The percentage distribution of reasons of critique preferences for each evaluated model (our method, the base model, the large model, and DPCO) is shown in Figure 4. Additionally, we conduct a case study with five representative cases from the human evaluation dataset, including critiques generated by our method and baseline approaches, as well as the refinements produced by *LLaMA-2-7B-Chat*, in Figures 7-11 of Appendix F due to space constraints.

The results in Figure 4 clearly demonstrate that our method excels in generating critiques that are not only correct, clear, and well-structured, but also offer constructive and feasible suggestions or detailed revision steps that are easy for the actor model to follow. In contrast, DPCO, which rely on LLM-annotated preferences of critique pairs, often produce critiques that are richer and more thorough in their analysis but tend to offer vague or less specific suggestions. Furthermore, DPCO occasionally generate incorrect critiques or misidentify the target of critique. These findings underscore the effectiveness of our method in training critic models that generate precise, helpful critiques, significantly enhancing the iterative improvement of actor models.

7 Conclusion

In this paper we introduced RCO, a novel approach for training critic models to improve actor model refinement. By proposing a supervision scheme based on preferences for refined responses, we eliminate the need for direct critique quality assessment while rewarding critiques that drive meaningful improvements. Rigorous evaluations across five tasks show significant advancements in both critique quality and refinement capabilities compared to existing methods. Our in-depth analysis underscores the effectiveness of RCO and its potential to enhance the alignment between critic and actor models. These contributions provide valuable insights for designing more efficient and scalable systems for model refinement, laying the foundation for future research in this field.

Limitation

Despite the effectiveness and strong potential of RCO for broader applications, several limitations warrant further investigation and improvement. One key limitation is the inaccurate estimation of critique utility $CU(c_i|y_0, x)$ and regularization term $Z_\beta(y_0, x)$. In our study, we sample 4 critiques to estimate $Z_\beta(y_0, x)$ and 5 refined responses to estimate $CU(c_i|y_0, x)$, which maybe not sufficient. However, to achieve more accurate estimates, more data and preference judgments are required, which increase the cost for data collection. Additionally, our approach focus solely on critic models, failing to train actor models for improved utilization of critiques for refinement. Moving forward, we aim to develop more efficient methods for training the critic model. Furthermore, we are interested in advancing techniques for actor models to better interpret natural language critiques and leverage them to enhance their responses.

Ethical Consideration

In this work, we leveraged several available datasets to construct the training and test dataset of RCO. The HH-RLHF (Bai et al., 2022), TL;DR (Stiennon et al., 2020), Commonsense QA (Talmor et al., 2019), MATH (Hendrycks et al., 2021), GSM8K (Cobbe et al., 2021), TheoremQA (Chen et al., 2023), TabMWP (Lu et al., 2022), BBH (Suzgun et al., 2022), GPQA (Rein et al., 2024), MMLU (Hendrycks et al., 2020) and MBPP-Plus (Cassano et al., 2023) are under MIT licenses; the CNN DailyMail (See et al., 2017), MathQA (Amini et al., 2019), AQuA (Ling et al., 2017), PiQA (Bisk et al., 2020), MBPP (Austin et al., 2021), HumanEval (Zheng et al., 2023b), Chatbot Arena (Chiang et al., 2024), MMLU-Pro (Wang et al., 2024c), TruthfulQA (Lin et al., 2021), CriticEval (Lan et al., 2024b) and Reward-Bench (Lambert et al., 2024) are under Apache licenses; the AmbigQA (Min et al., 2020), ARC-Challenge (Clark et al., 2018), DS-1000 (Lai et al., 2022) and MMLU-Redux (Gema et al., 2024) are under CC BY-SA licenses; the NYT (Sandhaus, 2008) dataset is under LDC license; the ELI5 (Fan et al., 2019) dataset is under BSD license.

In these datasets, there exists some instructions with security issues. However, in RCO training, we constructed optimized prompt pairs that provide safety enhancements to these unsafe instructions, further mitigating the security issues.

Acknowledgements

This work was supported by the National Science Foundation for Distinguished Young Scholars (with No. 62125604). We would also like to thank China Telecom Research INstitute and Zhipu AI for sponsoring the computational resources in this work.

References

- Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.
- Afra Feyza Akyürek, Ekin Akyürek, Aman Madaan, Ashwin Kalyan, Peter Clark, Derry Wijaya, and Niket Tandon. 2023. RL4f: Generating natural language feedback with reinforcement learning for repairing model outputs. *arXiv preprint arXiv:2305.08844*.
- Aida Amini, Saadia Gabriel, Shanchuan Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. 2019. **MathQA: Towards interpretable math word problem solving with operation-based formalisms**. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2357–2367, Minneapolis, Minnesota. Association for Computational Linguistics.
- Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. 2021. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.
- Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. 2022. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*.
- Loubna Ben Allal, Niklas Muennighoff, Logesh Kumar Umapathi, Ben Lipkin, and Leandro von Werra. 2022. A framework for the evaluation of code generation models. <https://github.com/bigcode-project/bigcode-evaluation-harness>.
- Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, pages 7432–7439.
- Collin Burns, Pavel Izmailov, Jan Hendrik Kirchner, Bowen Baker, Leo Gao, Leopold Aschenbrenner, Yining Chen, Adrien Ecoffet, Manas Joglekar, Jan Leike, et al. 2023. Weak-to-strong generalization:

- Eliciting strong capabilities with weak supervision. *arXiv preprint arXiv:2312.09390*.
- 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*.
- Maosong Cao, Alexander Lam, Haodong Duan, Hongwei Liu, Songyang Zhang, and Kai Chen. 2024. Compassjudge-1: All-in-one judge model helps model evaluation and evolution. *arXiv preprint arXiv:2410.16256*.
- Federico Cassano, John Gouwar, Daniel Nguyen, Sydney Nguyen, Luna Phipps-Costin, Donald Pinckney, Ming-Ho Yee, Yangtian Zi, Carolyn Jane Anderson, Molly Q Feldman, et al. 2023. Multipl-e: a scalable and polyglot approach to benchmarking neural code generation. *IEEE Transactions on Software Engineering*, 49(7):3675–3691.
- Justin Chih-Yao Chen, Archiki Prasad, Swarnadeep Saha, Elias Stengel-Eskin, and Mohit Bansal. 2024. Magicore: Multi-agent, iterative, coarse-to-fine refinement for reasoning. *arXiv preprint arXiv:2409.12147*.
- Wenhu Chen, Ming Yin, Max Ku, Pan Lu, Yixin Wan, Xueguang Ma, Jianyu Xu, Xinyi Wang, and Tony Xia. 2023. Theoremqa: A theorem-driven question answering dataset. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7889–7901.
- Wei-Lin Chiang, Lianmin Zheng, Ying Sheng, Anastasios Nikolas Angelopoulos, Tianle Li, Dacheng Li, Hao Zhang, Banghua Zhu, Michael Jordan, Joseph E Gonzalez, et al. 2024. Chatbot arena: An open platform for evaluating llms by human preference. *arXiv preprint arXiv:2403.04132*.
- Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv:1803.05457v1*.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.
- Ganqu Cui, Lifan Yuan, Ning Ding, Guanming Yao, Bingxiang He, Wei Zhu, Yuan Ni, Guotong Xie, Ruobing Xie, Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2024. ULTRAFEEDBACK: boosting language models with scaled AI feedback. In *Forty-first International Conference on Machine Learning*.
- 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*.
- Angela Fan, Yacine Jernite, Ethan Perez, David Grangier, Jason Weston, and Michael Auli. 2019. Eli5: Long form question answering. *arXiv preprint arXiv:1907.09190*.
- Aryo Pradipta Gema, Joshua Ong Jun Leang, Giwon Hong, Alessio Devoto, Alberto Carlo Maria Mancino, Rohit Saxena, Xuanli He, Yu Zhao, Xiaotang Du, Mohammad Reza Ghasemi Madani, et al. 2024. Are we done with mmlu? *arXiv preprint arXiv:2406.04127*.
- Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.
- Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.
- Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. 2023. Large language models cannot self-correct reasoning yet. *arXiv preprint arXiv:2310.01798*.
- Jiaming Ji, Boyuan Chen, Hantao Lou, Donghai Hong, Borong Zhang, Xuehai Pan, Juntao Dai, and Yaodong Yang. 2024. Aligner: Achieving efficient alignment through weak-to-strong correction. *arXiv preprint arXiv:2402.02416*.
- Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. 2023. Llm-blender: Ensembling large language models with pairwise ranking and generative fusion. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14165–14178.
- Dongwei Jiang, Jingyu Zhang, Orion Weller, Nathaniel Weir, Benjamin Van Durme, and Daniel Khashabi. 2024. Self-[in] correct: Llms struggle with refining self-generated responses. *arXiv preprint arXiv:2404.04298*.
- Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. triviaqa: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension. *arXiv e-prints*, arXiv:1705.03551.
- Pei Ke, Bosi Wen, Andrew Feng, Xiao Liu, Xuanyu Lei, Jiale Cheng, Shengyuan Wang, Aohan Zeng, Yuxiao Dong, Hongning Wang, et al. 2024. Critiquellm: Towards an informative critique generation model for evaluation of large language model generation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13034–13054.
- Seungone Kim, Juyoung Suk, Shayne Longpre, Bill Yuchen Lin, Jamin Shin, Sean Welleck, Graham Neubig, Moontae Lee, Kyungjae Lee, and Minjoon Seo. 2024. Prometheus 2: An open source language model specialized in evaluating other language models. *arXiv preprint arXiv:2405.01535*.

- Yuhang Lai, Chengxi Li, Yiming Wang, Tianyi Zhang, Ruiqi Zhong, Luke Zettlemoyer, Scott Wen tau Yih, Daniel Fried, Sida Wang, and Tao Yu. 2022. Ds-1000: A natural and reliable benchmark for data science code generation. *ArXiv*, abs/2211.11501.
- Nathan Lambert, Valentina Pyatkin, Jacob Morrison, LJ Miranda, Bill Yuchen Lin, Khyathi Chandu, Nouha Dziri, Sachin Kumar, Tom Zick, Yejin Choi, et al. 2024. Rewardbench: Evaluating reward models for language modeling. *arXiv preprint arXiv:2403.13787*.
- Tian Lan, Wenwei Zhang, Chengqi Lyu, Shuaibin Li, Chen Xu, Heyan Huang, Dahua Lin, Xian-Ling Mao, and Kai Chen. 2024a. Training language models to critique with multi-agent feedback. *arXiv preprint arXiv:2410.15287*.
- Tian Lan, Wenwei Zhang, Chen Xu, Heyan Huang, Dahua Lin, Kai Chen, and Xian-ling Mao. 2024b. Criticbench: Evaluating large language models as critic. *arXiv preprint arXiv:2402.13764*.
- Dawei Li, Bohan Jiang, Liangjie Huang, Alimohammad Beigi, Chengshuai Zhao, Zhen Tan, Amrita Bhat-tacharjee, Yuxuan Jiang, Canyu Chen, Tianhao Wu, et al. 2024a. From generation to judgment: Opportunities and challenges of llm-as-a-judge. *arXiv preprint arXiv:2411.16594*.
- Haitao Li, Qian Dong, Junjie Chen, Huixue Su, Yujia Zhou, Qingyao Ai, Ziyi Ye, and Yiqun Liu. 2024b. Llm-as-judges: a comprehensive survey on llm-based evaluation methods. *arXiv preprint arXiv:2412.05579*.
- Junlong Li, Shichao Sun, Weizhe Yuan, Run-Ze Fan, Hai Zhao, and Pengfei Liu. 2023. Generative judge for evaluating alignment. *arXiv preprint arXiv:2310.05470*.
- Stephanie Lin, Jacob Hilton, and Owain Evans. 2021. Truthfulqa: Measuring how models mimic human falsehoods. *arXiv preprint arXiv:2109.07958*.
- Zicheng Lin, Zhibin Gou, Tian Liang, Ruilin Luo, Haowei Liu, and Yujiu Yang. 2024. Criticbench: Benchmarking llms for critique-correct reasoning. *arXiv preprint arXiv:2402.14809*.
- Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. *ACL*.
- Pan Lu, Liang Qiu, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, Tanmay Rajpurohit, Peter Clark, and Ashwin Kalyan. 2022. Dynamic prompt learning via policy gradient for semi-structured mathematical reasoning. *arXiv preprint arXiv:2209.14610*.
- Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegrefe, Uri Alon, Nouha Dziri, Shrimai Prabhunoye, Yiming Yang, et al. 2023. Self-refine: Iterative refinement with self-feedback. *arXiv preprint arXiv:2303.17651*.
- Nat McAleese, Rai Michael Pokorny, Juan Felipe Ceron Uribe, Evgenia Nitishinskaya, Maja Trebacz, and Jan Leike. 2024. Llm critics help catch llm bugs. *arXiv preprint arXiv:2407.00215*.
- Sewon Min, Julian Michael, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2020. Ambigqa: Answering ambiguous open-domain questions. *arXiv preprint arXiv:2004.10645*.
- Bhuvanashree Murugadoss, Christian Poelitz, Ian Drosos, Vu Le, Nick McKenna, Carina Suzana Negreanu, Chris Parnin, and Advait Sarkar. 2024. Evaluating the evaluator: Measuring llms' adherence to task evaluation instructions. *arXiv preprint arXiv:2408.08781*.
- Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D Manning, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *arXiv preprint arXiv:2305.18290*.
- David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. 2024. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*.
- Pierre Harvey Richemond, Yunhao Tang, Daniel Guo, Daniele Calandriello, Mohammad Gheshlaghi Azar, Rafael Rafailov, Bernardo Avila Pires, Eugene Tarassov, Lucas Spangher, Will Ellsworth, et al. 2024. Offline regularised reinforcement learning for large language models alignment. *arXiv preprint arXiv:2405.19107*.
- Evan Sandhaus. 2008. The new york times annotated corpus. *Linguistic Data Consortium, Philadelphia*, 6(12):e26752.
- William Saunders, Catherine Yeh, Jeff Wu, Steven Bills, Long Ouyang, Jonathan Ward, and Jan Leike. 2022. Self-critiquing models for assisting human evaluators. *arXiv preprint arXiv:2206.05802*.
- Jérémy Scheurer, Jon Ander Campos, Tomasz Korbak, Jun Shern Chan, Angelica Chen, Kyunghyun Cho, and Ethan Perez. 2023. Training language models with language feedback at scale. *arXiv preprint arXiv:2303.16755*.
- Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. *arXiv preprint arXiv:1704.04368*.
- Tu Shiwen, Zhao Liang, Chris Yuhao Liu, Liang Zeng, and Yang Liu. 2024. [Skywork critic model series](https://huggingface.co/Skywork). <https://huggingface.co/Skywork>.
- Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. 2020. Learning to summarize with human feedback. *Advances in Neural Information Processing Systems*, 33:3008–3021.

- Shichao Sun, Junlong Li, Weizhe Yuan, Ruifeng Yuan, Wenjie Li, and Pengfei Liu. 2024. The critique of critique. *arXiv preprint arXiv:2401.04518*.
- Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, et al. 2022. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*.
- Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. **CommonsenseQA: A question answering challenge targeting commonsense knowledge**. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158, Minneapolis, Minnesota. Association for Computational Linguistics.
- Qwen Team. 2024. **Qwen2.5: A party of foundation models**.
- Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.
- Pat Verga, Sebastian Hofstatter, Sophia Althammer, Yixuan Su, Aleksandra Piktus, Arkady Arkhangorodsky, Minjie Xu, Naomi White, and Patrick Lewis. 2024. Replacing judges with juries: Evaluating llm generations with a panel of diverse models. *arXiv preprint arXiv:2404.18796*.
- Manya Wadhwa, Xinyu Zhao, Junyi Jessy Li, and Greg Durrett. 2024. Learning to refine with fine-grained natural language feedback. *arXiv preprint arXiv:2407.02397*.
- Peiyi Wang, Lei Li, Liang Chen, Zefan Cai, Dawei Zhu, Binghuai Lin, Yunbo Cao, Lingpeng Kong, Qi Liu, Tianyu Liu, and Zhifang Sui. 2024a. Large language models are not fair evaluators. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pages 9440–9450.
- Tianlu Wang, Ilia Kulikov, Olga Golovneva, Ping Yu, Weizhe Yuan, Jane Dwivedi-Yu, Richard Yuanzhe Pang, Maryam Fazel-Zarandi, Jason Weston, and Xian Li. 2024b. Self-taught evaluators. *arXiv preprint arXiv:2408.02666*.
- Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyang Jiang, et al. 2024c. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.
- Siwei Wu, Zhongyuan Peng, Xinrun Du, Tuney Zheng, Minghao Liu, Jialong Wu, Jiachen Ma, Yizhi Li, Jian Yang, Wangchunshu Zhou, et al. 2024. A comparative study on reasoning patterns of openai’s o1 model. *arXiv preprint arXiv:2410.13639*.
- Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Sainbayar Sukhbaatar, Jing Xu, and Jason Weston. 2024. Self-rewarding language models. *arXiv preprint arXiv:2401.10020*.
- Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2023a. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36:46595–46623.
- Qinkai Zheng, Xiao Xia, Xu Zou, Yuxiao Dong, Shan Wang, Yufei Xue, Zihan Wang, Lei Shen, Andi Wang, Yang Li, Teng Su, Zhilin Yang, and Jie Tang. 2023b. Codegeex: A pre-trained model for code generation with multilingual benchmarking on humaneval-x. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 5673–5684.
- Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. 2019. Fine-tuning language models for human preferences. *arXiv preprint arXiv:1909.08593*.

A Data Construction Prompts

Since our dataset construction process involves multiple stages, including generation of initial responses, critiques, and refined responses, and 5 different tasks, we design distinct prompts for each stage and task. The prompts for initial response generation are shown in Table 10. The prompts for critique collection are shown in Table 11-15. The prompts for refinement generation are shown in Table 18-22. The prompts for self-refinement generation are shown in Table 23-27. Note that we have distinct prompts for two code generation datasets, HumanEval and DS-1000, since a prompt format is already provided for DS-1000 in its repository. For *Auto-J-13B* and *UltraCM-13B* whose input prompt formats are given, we directly adopt their prompt format in data construction, training and testing, as shown in Table 16 and 17. For the rest base models, we use the designed input prompt formats.

The prompt for judging preferences for refinements are shown in Table 28-32, and the prompt for directly judging preferences for critiques are shown in Table 33-37.

B Evaluation Prompts

Our evaluation process involves GPT-4 scoring and preference evaluation, and we design distinct

Base Model	Method	LLaMA-2-7B		LLaMA-2-13B		LLaMA-2-70B		LLaMA-3-8B		LLaMA-3-70B		Overall	
		CU	RQS	CU	RQS	CU	RQS	CU	RQS	CU	RQS	CU	RQS
	Initial Answer	–	4.84	–	5.21	–	4.93	–	5.44	–	5.82	–	5.25
BASELINES													
	Self-refinement	58.7	5.34	71.4	6.09	72.2	5.59	73.2	6.08	<u>87.5</u>	<u>6.78</u>	72.6	5.98
	LLaMA-2-70B-Chat	69.1	5.91	76.2	6.28	71.4	5.99	66.4	6.32	79.1	6.58	72.4	6.21
	LLaMA-3-70B-Instruct	78.9	6.23	85.0	6.64	85.6	6.30	75.9	6.35	85.5	6.70	82.2	6.44
OUR METHOD													
LLaMA-2-7B-Chat	Base model	64.2	5.63	73.9	6.19	69.0	5.94	65.2	6.08	81.5	6.54	70.8	6.08
	+DPCO	68.2	5.74	74.9	6.20	70.2	5.99	67.2	6.18	80.8	6.67	72.3	6.16
	+RCO (Ours)	70.0	5.82	81.7	6.51	80.5	6.29	81.0	6.62	<u>92.1</u>	<u>7.21</u>	81.1	6.49
LLaMA-2-13B-Chat	Base model	65.3	5.62	75.2	6.20	71.3	6.02	72.8	6.33	82.9	6.60	73.5	6.17
	+DPCO	66.5	5.75	73.4	6.23	73.8	5.94	73.6	6.03	80.4	6.41	73.5	6.07
	+RCO (Ours)	72.8	5.74	81.4	6.32	82.0	6.14	79.7	6.54	<u>93.6</u>	<u>7.21</u>	81.9	6.42
LLaMA-3-8B-Instruct	Base model	72.1	6.03	78.5	6.55	78.2	6.14	71.0	6.38	79.4	6.56	75.8	6.33
	+DPCO	68.5	5.83	74.1	6.35	67.9	5.80	71.9	6.21	82.2	6.64	72.9	6.17
	+RCO (Ours)	82.6	6.37	86.3	6.75	85.2	6.34	78.4	6.69	<u>89.1</u>	<u>6.95</u>	84.3	6.62
Auto-J-13B	Base model	60.1	5.54	72.4	6.14	69.3	5.73	67.9	6.05	78.5	6.53	69.7	6.00
	+DPCO	68.1	5.77	74.1	6.33	70.7	5.93	74.2	6.29	83.8	6.49	74.2	6.16
	+RCO (Ours)	68.5	5.72	76.6	6.43	76.1	6.09	80.8	6.56	<u>91.9</u>	<u>7.00</u>	78.8	6.36
UltraCM-13B	Base model	61.1	5.63	63.1	5.97	60.6	5.51	61.6	5.88	71.4	6.14	63.6	5.83
	+DPCO	66.4	5.66	71.8	6.10	70.5	5.76	75.6	6.16	84.8	6.61	73.8	6.06
	+RCO (Ours)	68.0	5.75	76.2	6.33	76.9	6.07	82.6	6.45	<u>90.3</u>	<u>6.88</u>	78.8	6.29

Table 4: Evaluation results of our method and baselines with respect to actor models that generate the initial response, in terms of critique utility (CU) and refinement quality score (RQS). We omit the suffixes (“-Chat” and “-Instruct”) of actor models (the first row) for better appearance. The critique utility results reported in the table are multiplied by 100.

Base Model	Preference Judge	Acc. @ RewardBench	Dialog		Summ.		QA.		Math		Code		Overall	
			CU	RQS	CU	RQS	CU	RQS	CU	RQS	CU	RQS	CU	RQS
LLaMA-3-8B-Instruct	Qwen	86.1%	87.0	7.17	86.0	7.74	94.2	7.03	76.2	4.84	78.3	6.33	84.3	6.62
	GPT-4o	84.6%	87.7	7.15	86.7	7.76	94.7	7.09	74.9	4.82	79.3	6.40	84.7	6.64
	Skywork	89.3%	83.3	7.08	86.3	7.69	91.2	7.02	73.2	4.76	73.7	6.22	83.9	6.55
	InternLM	90.1%	84.6	7.15	88.1	7.59	93.4	7.13	73.4	4.69	78.3	6.39	83.6	6.59
	PairRM	54.2%	79.9	7.04	82.9	7.57	83.8	6.49	69.8	4.58	71.2	6.20	77.5	6.38
	Self-rewarding	69.7%	85.3	7.13	84.1	7.65	92.3	7.27	74.3	4.76	76.9	6.30	82.6	6.62

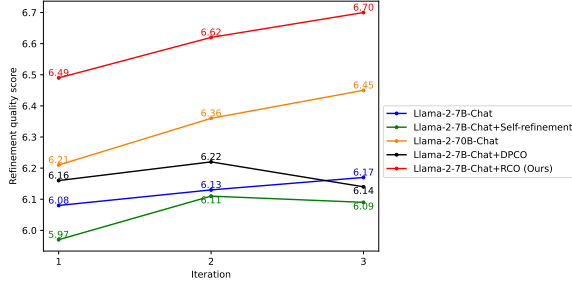
Table 5: Evaluation results of RCO training varying the preference judge model.

prompts for each of the evaluation settings and task. The prompts for preference judgment and preference scoring are shown in Table 28-32, and the prompts for response quality scoring are shown in Table 38-42.

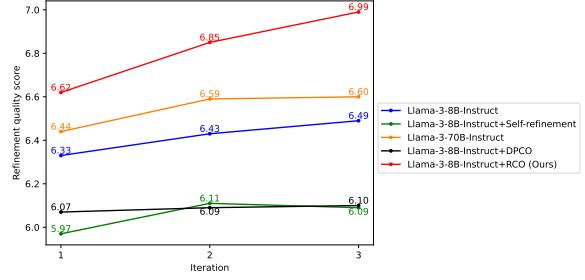
C Training Details

In our experiment, we train RCO and DPCO on 5 base critic models: *LLaMA-2-7B-Chat*, *LLaMA-2-13B-Chat*, *LLaMA-3-8B-Instruct*, *Auto-J-13B*, and *UltraCM-13B*. We train the smallest model *LLaMA-2-7B-Chat* on 4 NVIDIA H800 80GB GPUs with a batch size of 2, a gradient accumulation of 4 and 100 warmup steps. We train the medium-sized

model *LLaMA-3-8B-Instruct* on 4 NVIDIA H800 80GB GPUs with a batch size of 1, a gradient accumulation of 8 and 100 warmup steps. For the largest base models *LLaMA-3-8B-Instruct*, *Auto-J-13B*, and *UltraCM-13B*, we train each of these models on 6 NVIDIA H800 80GB GPUs with a batch size of 1, a gradient accumulation of 8 and 50 warmup steps. Each of these models are fully trained 5 epochs for RCO and 1 epoch for DPCO, ensuring equalized numbers of optimization steps, with a learning rate of 1×10^{-6} and a linear warmup schedule. We use $\beta = 0.1$ throughout our study. For DPCO, we incorporate all 10,000 prompts to train DPCO (which is $M = 5$ times of RCO),



(a) LLaMA-2-7B-Chat



(b) LLaMA-3-8B-Instruct

Figure 5: Refinement quality score of iterative refinement results with different base models.

Task	Dataset	Amount
Dialog	HH-RLHF (Bai et al., 2022)	2,000
Summ.	TL;DR (Stiennon et al., 2020)	710
	CNN DailyMail (See et al., 2017)	1,000
QA.	Commonsense QA (Talmor et al., 2019)	500
	Trivia QA (Joshi et al., 2017)	500
	AmbigQA (Min et al., 2020)	500
	ARC-Challenge (Clark et al., 2018)	500
	ELI5 (Fan et al., 2019)	500
Math	MathQA (Amini et al., 2019)	500
	TheoremQA (Chen et al., 2023)	500
	AQuA (Ling et al., 2017)	500
	TabMWP (Lu et al., 2022)	500
Code	HumanEval (Zheng et al., 2023b)	820
	DS-1000 (Lai et al., 2022)	970
Total		10,000

Table 6: Statistics of all the 14 source datasets in the training dataset.

in order to ensure equalized preference judgment times for RCO and DPCO. This is because RCO requires $N \times M \times 2$ refinement judgments and DPCO requires $N \times 2$ critique judgments. After filtering out invalid and inconsistent judgments, we gather 56,535 preference pairs for each critic model. These preference pairs are used to train critic models via the DPO algorithm.

We report the training time of RCO and DPCO in Table 9.

D Human Evaluation Details

For the human evaluation, we sample 200 responses from the benchmark, ensuring that each task has 40 responses. Given that the code generation task in CriticEval overlaps with our training dataset, and that CriticBench contains only reasoning and code generation tasks, we select the code generation task from CriticBench and the remaining four tasks from CriticEval. We report the

Task	Dataset	Amount
Dialog	Chatbot Arena (Chiang et al., 2024)	500
Summ.	New York Times (Sandhaus, 2008)	500
QA.	PiQA (Bisk et al., 2020)	500
Math	MATH (Hendrycks et al., 2021)	250
	GSM8K (Cobbe et al., 2021)	250
Code	MBPP (Austin et al., 2021)	500
Total		2,500

Table 7: Statistics of all the 7 source datasets in the test dataset.

Evaluators Pair	(1, 2)	(1, 3)	(2, 3)
Critique	63%	58.5%	61.5%
Refinement	79.5%	73.5%	77%

Table 8: Agreement rate between human evaluators pairs for both critique and refinement evaluation.

guidelines of human evaluation as Table 43. Three Ph.D students specializing in computer technology and NLP independently label their preferences for critiques and refinements. We report the agreement rate of the evaluation in Table 8.

E Iterative Refinement Experiment

Previous studies on iterative self-critique and refinement for enhancing LLMs have faced criticism, as LLMs may not always improve through this method (Huang et al., 2023; Jiang et al., 2024). In contrast, our work demonstrates that critic models, trained with our refinement-oriented methodology, can effectively drive continuous improvement in responses generated by actor models. We conducted experiments with two base models, *LLaMA-2-7B-Chat* and *LLaMA-3-8B-Instruct*, over a three-turn critique-refinement cycle. The refinements' quality

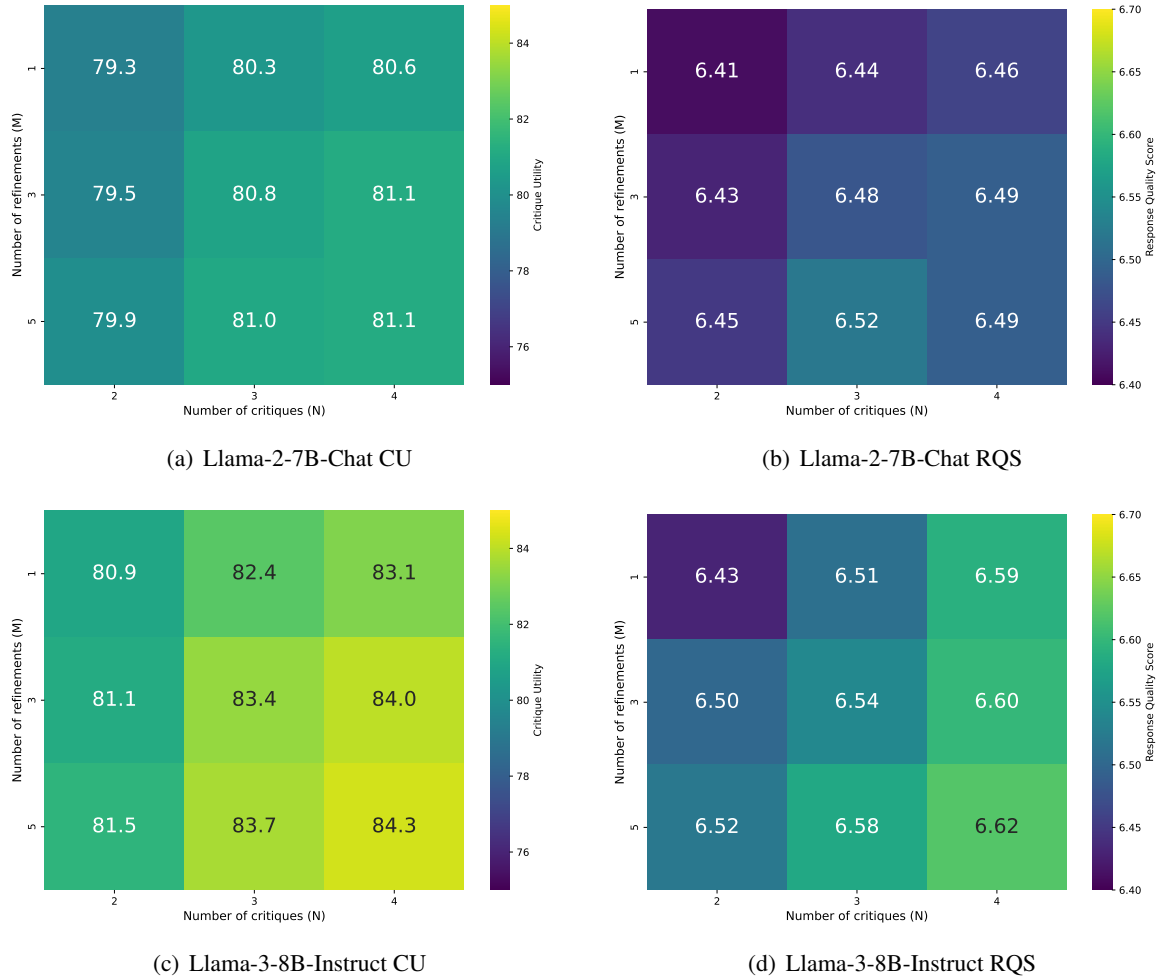


Figure 6: Heatmap results of CU and RQS metrics varying number of critiques (N) and number of refinements (M).

Models	DPCO	RCO
LLaMA-2-7B-Chat	4.5h	6.0h
LLaMA-2-13B-Chat	13.4h	16.9h
LLaMA-3-8B-Instruct	7.1h	8.8h
Auto-J-13B	13.6h	17h
UltraCM-13B	13.2h	16.4h

Table 9: Training time of DPCO and RCO.

was evaluated across iterations for our method and several baselines, as shown in Figure 5. Our results reveal a consistent upward performance trend with our approach, while baselines show limited improvement after the second iteration, highlighting the superiority of our method in guiding iterative refinement.

F Case Studies

To further investigate why our method produces more effective critiques for refining actor models, we selected 5 representative cases from the human evaluation dataset, one case belonging to each task. These cases, along with the critiques generated by our method and the baseline approaches, as well as the refinements produced by *LLaMA-2-7B-Chat*, are presented in Figure 7-11.

The case analysis reveals that our proposed method is capable of generating correct and concise critiques, offering clear, feasible suggestions that are easy for the actor model to follow. In contrast, DPCO tend to provide more detailed analyses but their suggestions are often vague and less specific. Specifically, in the first case from dialog generation task presented in Figure 7, DPCO makes a factual error suggesting that Koldskål is a traditional Danish cold sour soup made with buttermilk, eggs, and pickled vegetables. In the second case from the

summarization task presented in Figure 8, DPCO misidentifies the target of critique, mistakenly focusing on the article rather than the summary. In the third case from the question answering task presented in Figure 9, DPCO offers unclear and infeasible suggestions to actors, such as including citations to academic studies, causing the actor model to hallucinate. In the fourth case from the mathematical reasoning task presented in Figure 10, the analysis and suggestions of DPCO are either incorrect or unnecessary, failing to find the main mistake of the initial response. In the fifth case from the code generation task presented in Figure 11, the suggestions of DPCO are vague and not understandable for actor models. critic models trained by our RCO method, on the other hand, are able to offer constructive suggestions or detailed steps for refinement, which is beneficial for actor models to improve their answers. These observations highlight the efficacy of our method in training critic models that generate more precise and helpful critiques for the iterative improvement of actor models, especially when compared to DPCO.

Prompts for Initial Response Generation

Dialog Generation

Directly adopt the original prompt

Summarization - TL;DR

Summarize the following Reddit Post:

SUBREDDIT: {subreddit}

TITLE: {title}

POST: {post}

Your Summary:

Summarization - CNN-DailyMail

Summarize the following news:

{news}

Your Summary:

Question Answering - Multiple Choices (Commonsense QA, ARC-Challenge)

Question: {prompt}

Answer choices: {choices}

Explain your reasoning. You must choose only one option from A to E. Your final answer should be a single letter from A to E, in the form [[answer]], at the end of your response. Your answer:

Question Answering - Others (Trivia-QA, ELI5, AmbigQA)

Question: {prompt}

Your Answer:

Mathematical Reasoning - Multiple Choices (MathQA, AQuA)

Can you solve the following math problem? {prompt}

Answer choices: {choices}

Explain your reasoning. You must choose only one option from A to E. Your final answer should be a single letter from A to E, in the form [[answer]], at the end of your response. Your answer:

Mathematical Reasoning - Table (TabMWP)

Can you solve the following math problem according to the information given in the table? {prompt}

Table: {table_title}

{table_content}

Explain your reasoning. Your final answer should be a single numerical number, in the form $\boxed{\text{answer}}$, at the end of your response.

Mathematical Reasoning - Others (TheoremQA)

Can you solve the following math problem? {prompt}

Explain your reasoning. Your final answer should be a single numerical number, in the form $\boxed{\text{answer}}$, at the end of your response.

Code Generation - HumanEval

Complete the code following the instruction given in the comment, as well as the format and the indentation.

{prompt}

Code Generation - DS-1000

Write a short code to solve the problem following the given format and indentation. Place the executable code between `<code>` and `</code>` tags, without any other non-executable things.

{prompt}

Table 10: Prompts for collecting initial responses from actor models.

Prompts for Critique Generation

Dialog Generation

You are an impartial judge. You are shown a dialog between a human and an AI assistant. Comment on the assistant's response to the dialog according to the criteria of helpful, harmless and correct. After that, provide suggestions for improving the assistant's response according to your comment, starting with "Suggestions for improvement:".

{prompt}

Assistant's Response: {response}

Your comment:

Table 11: Prompts for collecting critiques of dialog generation tasks, for *LLaMA-2-7B-Chat*, *LLaMA-2-13B-Chat* and *LLaMA-3-8B-Instruct* base and trained models.

Summarization - TL;DR

You are an impartial judge. You are shown a Reddit post and a summary. Comment on the summary by finding if it misses any key information from the post or contains any unnecessary information. After that, provide suggestions for improving the summary according to your comment, starting with "Suggestions for improvement:".

SUBREDDIT: {subreddit}

TITLE: {title}

POST: {post}

Assistant's Summary: {response}

Your comment:

Summarization - CNN-DailyMail

You are an impartial judge. You are shown a piece of news and a summary. Comment on the summary by finding if it misses any key information from the post or contains any unnecessary information. After that, provide suggestions for improving the summary according to your comment, starting with "Suggestions for improvement:".

{news}

Assistant's Summary: {response}

Your comment:

Table 12: Prompts for collecting critiques of summarization tasks, for *LLaMA-2-7B-Chat*, *LLaMA-2-13B-Chat* and *LLaMA-3-8B-Instruct* base and trained models.

Question Answering - Multiple Choices (Commonsense QA, ARC-Challenge)

You are an impartial judge. You are shown a question and an answer. Comment on the answer and find problems with it. After that, provide suggestions for improving the answer according to your comment, starting with "Suggestions for improvement:".

Question: {prompt}

Answer choices: {choices}

Assistant's Answer: {response}

Your comment:

Question Answering - Others (Trivia-QA, ELI5, AmbigQA)

You are an impartial judge. You are shown a question and an answer. Comment on the answer and find problems with it. After that, provide suggestions for improving the answer according to your comment, starting with "Suggestions for improvement:".

Question: {prompt}

Assistant's Answer: {response}

Your comment:

Table 13: Prompts for collecting critiques of question answering tasks, for *LLaMA-2-7B-Chat*, *LLaMA-2-13B-Chat* and *LLaMA-3-8B-Instruct* base and trained models.

Mathematical Reasoning - Multiple Choices (MathQA, AQuA)

You are an expert on mathematics. You are shown a math problem and the answer to it. Comment on the answer and find problems with it. After that, provide suggestions for improving the answer according to your comment, starting with "Suggestions for improvement:".

Problem: {prompt}

Answer choices: {choices}

Assistant's Answer: {response}

Your comment:

Mathematical Reasoning - Table (TabMWP)

You are an expert on mathematics. You are shown a math problem and the answer to it. Comment on the answer and find problems with it. After that, provide suggestions for improving the answer according to your comment, starting with "Suggestions for improvement:".

Problem: {prompt}

Table: {table_title}

{table_content}

Assistant's Answer: {response}

Your comment:

Mathematical Reasoning - Others (TheoremQA)

You are an expert on mathematics. You are shown a math problem and the answer to it. Comment on the answer and find problems with it. After that, provide suggestions for improving the answer according to your comment, starting with "Suggestions for improvement:".

Problem: {prompt}

Assistant's Answer: {response}

Your comment:

Table 14: Prompts for collecting critiques of mathematical reasoning tasks, for *LLaMA-2-7B-Chat*, *LLaMA-2-13B-Chat* and *LLaMA-3-8B-Instruct* base and trained models.

Code Generation - HumanEval

You are an expert on computer programming. You are shown a code completion according to the requirements presented in the comment line of the code. Evaluate the correctness and readability of the code, and find if it meet the presented requirements. After that, provide suggestions for improving the code according to your evaluation, starting with "Suggestions for improvement:".

{prompt} {response}

Your evaluation:

Code Generation - DS-1000

You are an expert on computer programming. You are shown a code that proposes to solve the coding problem. Evaluate the correctness and readability of the code completion, and find if it meet the presented requirements. Remember not to comment on anything between 'A:' and 'BEGIN SOLUTION'. After that, provide suggestions for improving the code according to your evaluation, starting with "Suggestions for improvement":

{prompt} {response}

Your evaluation:

Table 15: Prompts for collecting critiques of code generation tasks, for *LLaMA-2-7B-Chat*, *LLaMA-2-13B-Chat* and *LLaMA-3-8B-Instruct* base and trained models.

Auto-J-13B Prompt

[INST] Write critiques for a submitted response on a given user's query, and grade the response:

[BEGIN DATA]

[Query]: {prompt}

[Response]: {answer} ***

[END DATA]

Write critiques for this response. After that, you should give a final rating for the response on a scale of 1 to 10 by strictly following this format: "[[rating]]", for example: "Rating: [[5]]". [/INST]

Table 16: Prompt for collecting critiques for *Auto-J-13B* base and trained models.

UltraCM-13B Prompt

Given my answer to an instruction, your role is to provide specific and constructive feedback for me. You should find the best way for me to learn from your feedback and improve my performance. You should consider multiple aspects of my answer, including helpfulness, truthfulness, honesty, and to what extent the answer follows instructions.

—
Instruction

{prompt}

Answer

{answer}

—

Please act as a teacher and provide specific and constructive feedback. Besides describing the weaknesses of the answer, you should also provide specific suggestions to guide me toward understanding how to improve. Please note, however, that your suggestions should help me better complete the instructions, but you should not introduce new requirements that are not mentioned in the instructions. Your feedback should focus on enhancing my ability to think critically and respond accurately. However, never explicitly provide the reference answer, nor do polite phrases be required. Only respond with concise feedback in chat style. Finally, score the overall quality of the answer from 1 to 10, where 1 is the worst and 10 is the best.

Format

Feedback

Overall Score: [1-10]

[Your feedback]

—

Feedback

Overall Score:

Table 17: Prompts for collecting critiques for *Ultra-CM-13B* base and trained models.

Prompts for Refined Response Generation

Dialog Generation

You are shown a dialog between a human and an AI assistant. An impartial judge on AI assistants has made comments on the assistant's response to the dialog. Please revise the assistant's response to improve its quality according to the suggestions for improvement provided in the comment, starting with "My revised response:".

—
{prompt}

—
Assistant's Response: {response}

—
Comment by the judge: {critique}

—
Your revision:

Table 18: Prompts for collecting refined responses of dialog generation tasks.

Summarization - TL;DR

You are shown a Reddit post and a summary. An impartial judge has made comments on the summary. Please revise the summary to improve its quality according to the suggestions for improvement provided in the comment, starting with "My revised summary:".

SUBREDDIT: {subreddit}

TITLE: {title}

POST: {post}

Original Summary: {response}

Comment by the judge: {critique}

Your revision:

Summarization - CNN-DailyMail

You are shown a piece of news and a summary. An impartial judge has made comments on the summary. Please revise the summary to improve its quality according to the suggestions for improvement provided in the comment, starting with "My revised summary:".

{news}

Original Summary: {response}

Comment by the judge: {critique}

Your revision:

Table 19: Prompts for collecting refined responses of summarization tasks.

Question Answering - Multiple Choices (Commonsense QA, ARC-Challenge)

You are shown a question and an answer. An impartial judge has made comments on the answer. Please revise the answer to improve its quality according to the suggestions for improvement provided in the comment, starting with "My revised answer:".

Question: {prompt}

Answer choices: {choices}

Original Answer: {response}

Comment by the judge: {critique}

Your revision:

Question Answering - Others (Trivia-QA, ELI5, AmbigQA)

You are shown a question and an answer. An impartial judge has made comments on the answer. Please revise the answer to improve its quality according to the suggestions for improvement provided in the comment, starting with "My revised answer:".

Question: {prompt}

Original Answer: {response}

Comment by the judge: {critique}

Your revision:

Table 20: Prompts for collecting refined responses of question answering tasks.

Mathematical Reasoning - Multiple Choices (MathQA, AQuA)

You are shown a math problem and an answer. An expert on mathematics has made comments on the answer. Please revise the answer to improve its quality according to the suggestions for improvement provided in the comment, starting with "My revised answer:".

Problem: {prompt}

Answer choices: {choices}

Original Answer: {response}

Comment by the expert: {critique}

Your revision:

Mathematical Reasoning - Table (TabMWP)

You are shown a math problem and an answer. An expert on mathematics has made comments on the answer. Please revise the answer to improve its quality according to the suggestions for improvement provided in the comment, starting with "My revised answer:".

Problem: {prompt}

Table: {table_title}

{table_content}

Original Answer: {response}

Comment by the expert: {critique}

Your revision:

Mathematical Reasoning - Others (TheoremQA)

You are shown a math problem and an answer. An expert on mathematics has made comments on the answer. Please revise the answer to improve its quality according to the suggestions for improvement provided in the comment, starting with "My revised answer:".

Problem: {prompt}

Original Answer: {response}

Comment by the expert: {critique}

Your revision:

Table 21: Prompts for collecting refined responses of mathematical reasoning tasks.

Code Generation - HumanEval

You are shown a code completion according to the requirements presented in the comment. An expert on computer programming has made critiques and advice for improvement on the code. Please revise the code completion to improve its quality according to the suggestions for improvement provided in the critique, starting with "My revised code:".

_____Original Code_____

{prompt} {response}

_____Critiques and Advice_____

{critique}

_____Your Revision_____

{prompt}

Code Generation - DS-1000

You are shown a code that proposes to solve the coding problem. An expert on computer programming has made critiques and advice for improvement on the code. Please revise the code completion to improve its quality according to the suggestions for improvement provided in the critique, starting with "My revised code:".

_____Original Code_____

{prompt} {response}

_____Critiques and Advice_____

{critique}

_____Your Revision_____

Table 22: Prompts for collecting refined responses of code generation tasks.

*Prompts for Self-Refinement Response Generation***Dialog Generation**

You are shown a dialog between a human and an AI assistant. Please revise the assistant's response to improve its quality according to your analysis, starting with "My revised response:".

{prompt}

Assistant's Response: {response}

Your revision:

Table 23: Prompts for collecting self-refinement responses of dialog generation tasks.

Summarization - TL;DR

You are shown a Reddit post and a summary of it. Please revise the summary to improve its quality according to your analysis, starting with "My revised summary:".

SUBREDDIT: {subreddit}

TITLE: {title}

POST: {post}

Original Summary: {response}

Your revision:

Summarization - CNN-DailyMail

You are shown a piece of news and a summary of it. Please revise the summary to improve its quality according to your analysis, starting with "My revised summary:".

{news}

Original Summary: {response}

Your revision:

Table 24: Prompts for collecting self-refinement responses of summarization tasks.

Question Answering - Multiple Choices (Commonsense QA, ARC-Challenge)

You are shown a question and an answer. Please revise the answer to improve its quality according to your analysis, starting with "My revised answer:".

Question: {prompt}

Answer choices: {choices}

Original Answer: {response}

Your revision:

Question Answering - Others (Trivia-QA, ELI5, AmbigQA)

You are shown a question and an answer. Please revise the answer to improve its quality according to your analysis, starting with "My revised answer:".

Question: {prompt}

Original Answer: {response}

Your revision:

Table 25: Prompts for collecting self-refinement responses of question answering tasks.

Mathematical Reasoning - Multiple Choices (MathQA, AQuA)

You are shown a math problem and an answer. Please revise the answer to improve its quality according to your analysis, starting with "My revised answer:".

Problem: {prompt}

Answer choices: {choices}

Original Answer: {response}

Your revision:

Mathematical Reasoning - Table (TabMWP)

You are shown a math problem and an answer. Please revise the answer to improve its quality according to your analysis, starting with "My revised answer:".

Problem: {prompt}

Table: {table_title}

{table_content}

Original Answer: {response}

Your revision:

Mathematical Reasoning - Others (TheoremQA)

You are shown a math problem and an answer. Please revise the answer to improve its quality according to your analysis, starting with "My revised answer:".

Problem: {prompt}

Original Answer: {response}

Your revision:

Table 26: Prompts for collecting self-refinement responses of mathematical reasoning tasks.

Code Generation - HumanEval

You are shown a code completion according to the requirements presented in the comment. Please revise the code to make it more correct and readable, starting with "My revised code:".

_____Original Code_____

{prompt} {response}

_____Your Revision_____

{prompt}

Code Generation - DS-1000

You are shown a code that proposes to solve the coding problem. Please revise the code to make it more correct and readable, starting with "My revised code:".

_____Original Code_____

{prompt} {response}

_____Your Revision_____

Table 27: Prompts for collecting self-refinement responses of code generation tasks.

Dialog Generation

[SYSTEM]

Please act as an impartial judge and evaluate the quality of the responses provided by two AI assistants to the conversation displayed below. You should choose the assistant that follows the user’s instructions better. Your evaluation should consider factors such as the helpfulness, relevance, accuracy, depth, creativity, and level of detail of their responses. You should focus on who provides a better response. Begin your evaluation by comparing the responses of the two assistants and provide a short explanation. Avoid any position biases and ensure that the order in which the responses were presented does not influence your decision. Do not allow the length of the responses to influence your evaluation. Do not favor certain names of the assistants. Be as objective as possible. After providing your explanation, output your final verdict by strictly following this format: "[[A]]" if assistant A is better, "[[B]]" if assistant B is better, and "[[C]]" for a tie.

[USER]

[Conversation]

{prompt}

[The Start of Assistant A’s Response]

{answer_0}

[The End of Assistant A’s Response]

[The Start of Assistant B’s Response]

{answer_1}

[The End of Assistant B’s Response]

Table 28: Prompt for refinement preference judgment and evaluation for dialog generation tasks.

Summarization

[SYSTEM]

Please act as an impartial judge and evaluate the quality of the summaries provided by two AI assistants to the {kind} displayed below. Your evaluation should consider whether their summaries include all key information from the original article and avoid false or unnecessary sentences. You should decide which assistant’s summary is better. Begin your evaluation by comparing both assistants’ summaries and provide a short explanation. Avoid any position biases and ensure that the order in which the responses were presented does not influence your decision. Do not allow the length of the responses to influence your evaluation. Do not favor certain names of the assistants. Be as objective as possible. After providing your explanation, output your final verdict by strictly following this format: "[[A]]" if assistant A is better, "[[B]]" if assistant B is better, and "[[C]]" for a tie.

[USER]

[{kind}]

{prompt}

[The Start of Assistant A’s Summary]

{answer_0}

[The End of Assistant A’s Summary]

[The Start of Assistant B’s Summary]

{answer_1}

[The End of Assistant B’s Summary]

Table 29: Prompt for refinement preference judgment and evaluation for summarization tasks. The “kind” field will be “Reddit post” or “News”, conditioned to whether this prompt is from TL;DR or CNN-DailyMail dataset.

Question Answering

[SYSTEM]

Please act as an impartial judge and evaluate the quality of the answers provided by two AI assistants to the user question displayed below. You should choose the assistant that follows the user's instructions and answers the user's question better. Your evaluation should consider factors such as the helpfulness, relevance, accuracy, depth, creativity, and level of detail of their responses. Begin your evaluation by comparing the two responses and provide a short explanation. Avoid any position biases and ensure that the order in which the responses were presented does not influence your decision. Do not allow the length of the responses to influence your evaluation. Do not favor certain names of the assistants. Be as objective as possible. After providing your explanation, output your final verdict by strictly following this format: "[[A]]" if assistant A is better, "[[B]]" if assistant B is better, and "[[C]]" for a tie.

[USER]

[User Question]

{prompt}

[The Start of Assistant A's Answer]

{answer_0}

[The End of Assistant A's Answer]

[The Start of Assistant B's Answer]

{answer_1}

[The End of Assistant B's Answer]

Table 30: Prompt for refinement preference judgment and evaluation for question answering tasks.

Mathematical Reasoning

[SYSTEM]

Please act as an impartial judge and evaluate the quality of the answers provided by two AI assistants to the math problem displayed below. Your evaluation should consider correctness and helpfulness. You will be given a reference answer, assistant A's answer, and assistant B's answer. Your job is to evaluate which assistant's answer is better. Begin your evaluation by comparing both assistants' answers with the reference answer. Identify and correct any mistakes. Avoid any position biases and ensure that the order in which the responses were presented does not influence your decision. Do not allow the length of the responses to influence your evaluation. Do not favor certain names of the assistants. Be as objective as possible. After providing your explanation, output your final verdict by strictly following this format: "[[A]]" if assistant A is better, "[[B]]" if assistant B is better, and "[[C]]" for a tie.

[USER]

[Math Problem]

{prompt}

[The Start of Reference Answer]

{ref_answer}

[The End of Reference Answer]

[The Start of Assistant A's Answer]

{answer_0}

[The End of Assistant A's Answer]

[The Start of Assistant B's Answer]

{answer_1}

[The End of Assistant B's Answer]

Table 31: Prompt for refinement preference judgment and evaluation for mathematical reasoning tasks.

Code Generation

[SYSTEM]

Please act as an impartial judge and evaluate the quality of the code provided by two AI assistants to the requirements displayed below. Your evaluation should consider correctness and helpfulness. You should decide which assistant's provided code is better. Begin your evaluation by comparing both assistants' codes and provide a short explanation. Identify and correct any mistakes. Avoid any position biases and ensure that the order in which the responses were presented does not influence your decision. Do not allow the length of the responses to influence your evaluation. Do not favor certain names of the assistants. Be as objective as possible. After providing your explanation, output your final verdict by strictly following this format: "[[A]]" if assistant A is better, "[[B]]" if assistant B is better, and "[[C]]" for a tie.

[USER]

[Code Requirements]

{prompt}

[The Start of Assistant A's Code]

{answer_0}

[The End of Assistant A's Code]

[The Start of Assistant B's Code]

{answer_1}

[The End of Assistant B's Code]

Table 32: Prompt for refinement preference judgment and evaluation for code generation tasks.

Dialog Generation

Please act as an impartial judge and evaluate the quality of the critiques provided by two AI assistants for my response to the conversation displayed below. Your evaluation should focus on the quality, clarity, and constructiveness of the critiques, particularly the "Suggestions for improvement" field. You will be given the conversation, my response, assistant A's critique, and assistant B's critique. Your job is to assess which assistant's critique is better based on the following criteria:

1. **Accuracy:** Does the critique accurately identify any issues with my response? Are any mistakes or shortcomings in my response correctly pointed out?
2. **Clarity:** Is the critique clearly written, easy to understand, and well-structured? Does it explain the issues in a way that is accessible and straightforward?
3. **Constructiveness:** Does the critique provide practical and actionable suggestions for improvement? Are the suggestions detailed, specific, and relevant to the issues identified?
4. **Objectivity:** Is the critique unbiased and impartial? Does it focus solely on the quality of my response and avoid unnecessary personal opinions or judgments?
5. **Thoroughness:** Does the critique cover all significant aspects of my response, or does it overlook any important issues? Does it delve into the reasoning behind the suggestions for improvement?
6. **Tone:** Is the critique delivered in a respectful and professional tone, avoiding any condescension or harshness?

You should focus particularly on the "Suggestions for improvement" field in each critique and evaluate how well each assistant has provided guidance to improve the response. Avoid being influenced by the length of the critiques or the order in which they are presented. Do not favor one assistant over the other based on irrelevant factors. Be as objective as possible. After providing your explanation, output your final verdict by strictly following this format: "[[A]]" if assistant A is better, "[[B]]" if assistant B is better, and "[[C]]" for a tie.

[USER]

[Conversation]

{prompt}

[The Start of My Response]

{answer}

[The End of My Response]

[The Start of Assistant A's Critique]

{critique_0}

[The End of Assistant A's Critique]

[The Start of Assistant B's Critique]

{critique_1}

[The End of Assistant B's Critique]

Table 33: Prompt for critique preference judgment for dialog generation tasks.

Summarization

[SYSTEM]

Please act as an impartial judge and evaluate the quality of the critiques provided by two AI assistants for my summary to the {kind} displayed below. Your evaluation should focus on the quality, clarity, and constructiveness of the critiques, particularly the "Suggestions for improvement" field.

You will be given the {kind}, my summary, assistant A's critique, and assistant B's critique. Your job is to assess which assistant's critique is better based on the following criteria:

1. **Accuracy:** Does the critique accurately identify any issues with my summary? Are any mistakes or shortcomings in my summary correctly pointed out?
2. **Clarity:** Is the critique clearly written, easy to understand, and well-structured? Does it explain the issues in a way that is accessible and straightforward?
3. **Constructiveness:** Does the critique provide practical and actionable suggestions for improvement? Are the suggestions detailed, specific, and relevant to the issues identified?
4. **Objectivity:** Is the critique unbiased and impartial? Does it focus solely on the quality of my summary and avoid unnecessary personal opinions or judgments?
5. **Thoroughness:** Does the critique cover all significant aspects of my summary, or does it overlook any important issues? Does it delve into the reasoning behind the suggestions for improvement?
6. **Tone:** Is the critique delivered in a respectful and professional tone, avoiding any condescension or harshness?

You should focus particularly on the "Suggestions for improvement" field in each critique and evaluate how well each assistant has provided guidance to improve the summary. Avoid being influenced by the length of the critiques or the order in which they are presented. Do not favor one assistant over the other based on irrelevant factors. Be as objective as possible. After providing your explanation, output your final verdict by strictly following this format: "[[A]]" if assistant A is better, "[[B]]" if assistant B is better, and "[[C]]" for a tie.

[USER]

[{kind}]

{prompt}

[The Start of My Summary]

{answer}

[The End of My Summary]

[The Start of Assistant A's Critique]

{critique_0}

[The End of Assistant A's Critique]

[The Start of Assistant B's Critique]

{critique_1}

[The End of Assistant B's Critique]

Table 34: Prompt for critique preference judgment for summarization tasks. The "kind" field will be "Reddit post" or "News", conditioned to whether this prompt is from TL;DR or CNN-DailyMail dataset.

Question Answering

[SYSTEM]

Please act as an impartial judge and evaluate the quality of the critiques provided by two AI assistants for my answer to the question displayed below. Your evaluation should focus on the quality, clarity, and constructiveness of the critiques, particularly the "Suggestions for improvement" field.

You will be given the question, my answer, assistant A's critique, and assistant B's critique. Your job is to assess which assistant's critique is better based on the following criteria:

1. **Accuracy:** Does the critique accurately identify any issues with my answer? Are any mistakes or shortcomings in my answer correctly pointed out?
2. **Clarity:** Is the critique clearly written, easy to understand, and well-structured? Does it explain the issues in a way that is accessible and straightforward?
3. **Constructiveness:** Does the critique provide practical and actionable suggestions for improvement? Are the suggestions detailed, specific, and relevant to the issues identified?
4. **Objectivity:** Is the critique unbiased and impartial? Does it focus solely on the quality of my answer and avoid unnecessary personal opinions or judgments?
5. **Thoroughness:** Does the critique cover all significant aspects of my answer, or does it overlook any important issues? Does it delve into the reasoning behind the suggestions for improvement?
6. **Tone:** Is the critique delivered in a respectful and professional tone, avoiding any condescension or harshness?

You should focus particularly on the "Suggestions for improvement" field in each critique and evaluate how well each assistant has provided guidance to improve the answer. Avoid being influenced by the length of the critiques or the order in which they are presented. Do not favor one assistant over the other based on irrelevant factors. Be as objective as possible. After providing your explanation, output your final verdict by strictly following this format: "[[A]]" if assistant A is better, "[[B]]" if assistant B is better, and "[[C]]" for a tie.

[USER]

[User Question]

{prompt}

[The Start of My Answer]

{answer}

[The End of My Answer]

[The Start of Assistant A's Critique]

{critique_0}

[The End of Assistant A's Critique]

[The Start of Assistant B's Critique]

{critique_1}

[The End of Assistant B's Critique]

Table 35: Prompt for critique preference judgment for question answering tasks.

Mathematical Reasoning

[SYSTEM]

"Please act as an impartial judge and evaluate the quality of the critiques provided by two AI assistants for my answer to the math problem displayed below. Your evaluation should focus on the quality, clarity, and constructiveness of the critiques, particularly the "Suggestions for improvement" field.

You will be given the question, my answer, the reference answer, assistant A's critique, and assistant B's critique. Your job is to assess which assistant's critique is better based on the following criteria:

1. **Accuracy:** Does the critique accurately identify any issues with my answer? Are any mistakes or shortcomings in my answer correctly pointed out?
2. **Clarity:** Is the critique clearly written, easy to understand, and well-structured? Does it explain the issues in a way that is accessible and straightforward?
3. **Constructiveness:** Does the critique provide practical and actionable suggestions for improvement? Are the suggestions detailed, specific, and relevant to the issues identified?
4. **Objectivity:** Is the critique unbiased and impartial? Does it focus solely on the quality of my answer and avoid unnecessary personal opinions or judgments?
5. **Thoroughness:** Does the critique cover all significant aspects of my answer, or does it overlook any important issues? Does it delve into the reasoning behind the suggestions for improvement?
6. **Tone:** Is the critique delivered in a respectful and professional tone, avoiding any condescension or harshness?

You should focus particularly on the "Suggestions for improvement" field in each critique and evaluate how well each assistant has provided guidance to improve the answer. Avoid being influenced by the length of the critiques or the order in which they are presented. Do not favor one assistant over the other based on irrelevant factors. Be as objective as possible. After providing your explanation, output your final verdict by strictly following this format: "[[A]]" if assistant A is better, "[[B]]" if assistant B is better, and "[[C]]" for a tie.

[USER]

[Math Problem]

{prompt}

[The Start of Reference Answer]

{ref_answer}

[The End of Reference Answer]

[The Start of My Answer]

{answer}

[The End of My Answer]

[The Start of Assistant A's Critique]

{critique_0}

[The End of Assistant A's Critique]

[The Start of Assistant B's Critique]

{critique_1}

[The End of Assistant B's Critique]

Table 36: Prompt for critique preference judgment for mathematical reasoning tasks.

Code Generation

[SYSTEM]

Please act as an impartial judge and evaluate the quality of the critiques provided by two AI assistants for my code to the requirements displayed below. Your evaluation should focus on the quality, clarity, and constructiveness of the critiques, particularly the "Suggestions for improvement" field.

You will be given the question, my code, assistant A's critique, and assistant B's critique. Your job is to assess which assistant's critique is better based on the following criteria:

1. **Accuracy:** Does the critique accurately identify any issues with my code? Are any mistakes or shortcomings in my code correctly pointed out?
2. **Clarity:** Is the critique clearly written, easy to understand, and well-structured? Does it explain the issues in a way that is accessible and straightforward?
3. **Constructiveness:** Does the critique provide practical and actionable suggestions for improvement? Are the suggestions detailed, specific, and relevant to the issues identified?
4. **Objectivity:** Is the critique unbiased and impartial? Does it focus solely on the quality of my code and avoid unnecessary personal opinions or judgments?
5. **Thoroughness:** Does the critique cover all significant aspects of my code, or does it overlook any important issues? Does it delve into the reasoning behind the suggestions for improvement?
6. **Tone:** Is the critique delivered in a respectful and professional tone, avoiding any condescension or harshness?

You should focus particularly on the "Suggestions for improvement" field in each critique and evaluate how well each assistant has provided guidance to improve the code. Avoid being influenced by the length of the critiques or the order in which they are presented. Do not favor one assistant over the other based on irrelevant factors. Be as objective as possible. After providing your explanation, output your final verdict by strictly following this format: "[[A]]" if assistant A is better, "[[B]]" if assistant B is better, and "[[C]]" for a tie.

[USER]

[Code Requirements]

{prompt}

[The Start of My Code]

{answer}

[The End of My Code]

[The Start of Assistant A's Critique]

{critique_0}

[The End of Assistant A's Critique]

[The Start of Assistant B's Critique]

{critique_1}

[The End of Assistant B's Critique]

Table 37: Prompt for critique preference judgment for code generation tasks.

Response Quality Scoring: Dialog Generation

[SYSTEM]

Please act as an impartial judge and evaluate the quality of the response provided by an AI assistant to the conversation displayed below. Your evaluation should consider factors such as the helpfulness, relevance, accuracy, depth, creativity, and level of detail of the response. Begin your evaluation by providing a short explanation. Do not allow the length of the response to influence your evaluation. Be as objective as possible. After providing your explanation, you must rate the response on a scale of 1 to 10 by strictly following this format: "[[rating]]", for example: "Rating: [[5]]".

[USER]

[Conversation]

{prompt}

[The Start of Assistant's Response]

{answer}

[The End of Assistant's Response]

Table 38: Prompt for refinement quality scoring for dialog generation tasks.

Response Quality Scoring: Summarization

[SYSTEM]

Please act as an impartial judge and evaluate the quality of the summary provided by an AI assistant to the kind displayed below. Your evaluation should consider whether the summary include all key information from the original article and avoid false or unnecessary sentences. Begin your evaluation by providing a short explanation. Do not allow the length of the summary to influence your evaluation. Be as objective as possible. After providing your explanation, you must rate the summary on a scale of 1 to 10 by strictly following this format: "[[rating]]", for example: "Rating: [[5]]".

[USER]

[{kind}]

{prompt}

[The Start of Assistant's Summary]

{answer}

[The End of Assistant's Summary]

Table 39: Prompt for refinement quality scoring for summarization tasks. The "kind" field will be "Reddit post" or "News", conditioned to whether this prompt is from TL;DR or CNN-DailyMail dataset.

Response Quality Scoring: Question Answering

[SYSTEM]

Please act as an impartial judge and evaluate the quality of the answer provided by an AI assistant to the user question. Your evaluation should consider factors such as the helpfulness, relevance, accuracy, depth, creativity, and level of detail in the answer. Begin your evaluation by providing a short explanation. Do not allow the length of the answer to influence your evaluation. Be as objective as possible. After providing your explanation, you must rate the answer on a scale of 1 to 10 by strictly following this format: "[[rating]]", for example: "Rating: [[5]]".

[USER]

[User Question]

{prompt}

[The Start of Assistant's Answer]

{answer}

[The End of Assistant's Answer]

Table 40: Prompt for refinement quality scoring for question answering tasks.

Response Quality Scoring: Mathematical Reasoning

[SYSTEM]

Please act as an impartial judge and evaluate the quality of the answer provided by an AI assistant to the math problem. Your evaluation should consider correctness and helpfulness. You will be given a reference answer and the assistant's answer. Begin your evaluation by comparing the assistant's answer with the reference answer. Identify and correct any mistakes. Do not allow the length of the answer to influence your evaluation. Be as objective as possible. After providing your explanation, you must rate the answer on a scale of 1 to 10 by strictly following this format: "[[rating]]", for example: "Rating: [[5]]".

[USER]

[Math Problem]

{prompt}

[The Start of Reference Answer]

{ref_answer}

[The End of Reference Answer]

[The Start of Assistant's Answer]

{answer}

[The End of Assistant's Answer]

Table 41: Prompt for refinement quality scoring for mathematical reasoning tasks.

Response Quality Scoring: Code Generation

[SYSTEM]

Please act as an impartial judge and evaluate the quality of the code provided by an AI assistant to the requirements displayed below. Your evaluation should consider correctness and helpfulness. Begin your evaluation by providing a short explanation. Identify and correct any mistakes. Do not allow the length of the response to influence your evaluation. Be as objective as possible. After providing your explanation, you must rate the code on a scale of 1 to 10 by strictly following this format: "[[rating]]", for example: "Rating: [[5]]".

[USER]

[Code Requirements]

{prompt}

[The Start of Assistant's Code]

{answer}

[The End of Assistant's Code]

Table 42: Prompt for refinement quality scoring for code generation tasks.

Guidelines for human evaluation of critique:

You are given a user question, an initial response and two critiques of the response. Decide which critique is more accurate, thorough, clear and constructive for refinement of the actor model. Please carefully check and compare the given critiques, especially the "suggestions for improvement" section which is mainly used for refinement. Finally, label your verdict: "A" if critique A is better, "B" if critique B is better, and "C" for a tie. If you choose "A" or "B", please choose the main reason of your preference from the four criteria: accurate, thorough, clear and constructive.

Guidelines for human evaluation of refinement:

You are given a user question and two responses. You should choose the response that follows the user's instructions and answers the user's question better. Your evaluation should consider factors such as the helpfulness, harmlessness, relevance, accuracy, depth, creativity, and level of detail of the responses. Do not allow the length of the responses to influence your evaluation. Be as objective as possible. Finally, label your verdict: "A" if response A is better, "B" if response B is better, and "C" for a tie.

Table 43: Guidelines for human evaluation.

Question

Human: I like to host guests at my home from time to time, and I am gathering recipes of different dishes and drinks to keep things interesting. I am interested in trying some Danish dishes. Can you give me a recipe for Koldskål?
Assistant:

Initial Response

Here is a recipe for Koldskål: Ingredients: 4 Quart Jar, 3/4 cup chopped onion 1/2 cup chopped carrot 1/2 cup chopped celery 1 1/2 cups chopped cabbage 1/2 cup chopped mushrooms 1/2 cup chopped tomatoes 1/2 cup chopped zucchini 1/2 cup chopped yellow squash 1/2 cup chopped bell peppers 1/2 cup chopped yellow onion 1/2 cup chopped potatoes 1/2 cup chopped apples 1/2 cup chopped peaches 1/2 cup chopped dried apricots 1/2 cup chopped dried dates 1/2 cup chopped pecans 1/2 cup chopped walnuts 1/2 cup chopped dried cranberries 1/2 cup chopped dried blueberries 1/2 cup chopped dried raspberries 1/2 cup chopped dried maraschino cherries 1/2 cup chopped dried apricots 1/2 cup chopped dried figs (Repeat)

Critique by Llama-3-8B-Instruct+DPO

The assistant's response is a mess, to put it mildly. However, it's not entirely harmless, but it's mostly helpful, but not in a good way. However, it's not a recipe for Koldskål, but a list of ingredients that seems to be a repetition of the same ingredients over and over again, with no measurements, no cooking method, and no nutritional information.

The assistant's response is missing some crucial information, such as the fact that Koldskål is a traditional Danish cold sour soup made with buttermilk, eggs, and pickled vegetables, not a salad or a fruit salad. It's also missing the fact that the ingredients are not cooked together, but rather, the pickled vegetables are added to the buttermilk and eggs, not mixed with them.

Suggestions for improvement:

1. Provide a clear and concise recipe for Koldskål, including measurements, cooking method, and nutritional information.
2. Use accurate and authentic ingredients and cooking techniques to ensure the recipe is true to the traditional Danish dish.
3. Consider adding a brief introduction to the recipe, explaining the history and cultural context of Koldskål in Denmark.
4. Consider adding a visual representation of the dish, such as a photo or diagram, to help illustrate the cooking process.

Refinement based on critique by Llama-3-8B-Instruct+DPO

Koldskål is a traditional Danish cold sour soup made with buttermilk, eggs, and pickled vegetables. Here's a simple recipe to make this refreshing and tangy soup at home.

Ingredients:

- * 1 cup buttermilk
- * 2 eggs
- * 1/2 cup pickled red cabbage, chopped
- * 1/4 cup pickled carrots, chopped
- * Salt and pepper to taste

Cooking Method:

1. In a large bowl, whisk together the buttermilk and eggs until smooth.
2. Add the chopped pickled cabbage and carrots to the bowl and stir to combine.
3. Season with salt and pepper to taste.
4. Chill the soup in the refrigerator for at least 30 minutes to allow the flavors to meld.
5. Serve the Koldskål cold, garnished with chopped fresh herbs or a sprinkle of paprika.

Critique by Llama-3-8B-Instruct+RCO

Helpfulness: 0/5 - The assistant's response is not helpful at all. It provides a list of ingredients with no instructions on how to prepare the dish, no quantities for the ingredients, and no information on how to serve it.

Harmlessness: 5/5 - The assistant's response does not contain any harmful or offensive content.

Correctness: 0/5 - The assistant's response is not correct. Koldskål is a traditional Danish cold sour milk dessert, not a savory dish with a long list of ingredients. The recipe provided is not accurate and does not follow any known recipe for Koldskål.

Suggestions for improvement:

1. Provide a correct and accurate recipe for Koldskål, including ingredients, quantities, and instructions on how to prepare and serve the dish.
2. Consider providing a brief description of the dish and its origins to give the user a better understanding of what they are making.
3. Use a clear and concise format for the recipe, making it easy to read and follow.
4. Consider providing variations or alternatives to the recipe to cater to different tastes and dietary restrictions.

By following these suggestions, the assistant can provide a helpful, harmless, and correct response that will assist the user in making a delicious Danish dish.

Refinement based on critique by Llama-3-8B-Instruct+RCO

Thank you for your interest in trying a traditional Danish dish! Koldskål is a cold sour milk dessert that is popular in Denmark. Here is a recipe for Koldskål that you can try:

Ingredients:

- * 1 cup sour milk
- * 1/2 cup granulated sugar
- * 1/4 cup cornstarch
- * 1/4 teaspoon salt
- * 1/4 teaspoon vanilla extract
- * 1/4 cup chopped fresh fruit (such as strawberries or blueberries)

Instructions:

1. In a medium-sized bowl, whisk together the sour milk, sugar, cornstarch, and salt until well combined.
2. Add the vanilla extract and mix well.
3. Fold in the chopped fresh fruit.
4. Cover the bowl with plastic wrap and refrigerate for at least 2 hours or overnight.
5. Serve the Koldskål chilled, garnished with additional fresh fruit if desired.

I hope you enjoy this recipe for Koldskål! Let me know if you have any questions or if you would like any variations on the recipe.

Figure 7: One typical case from the dialog generation task in CriticEval dataset. Our critique is preferred mainly based on the criteria of **Accuracy**.

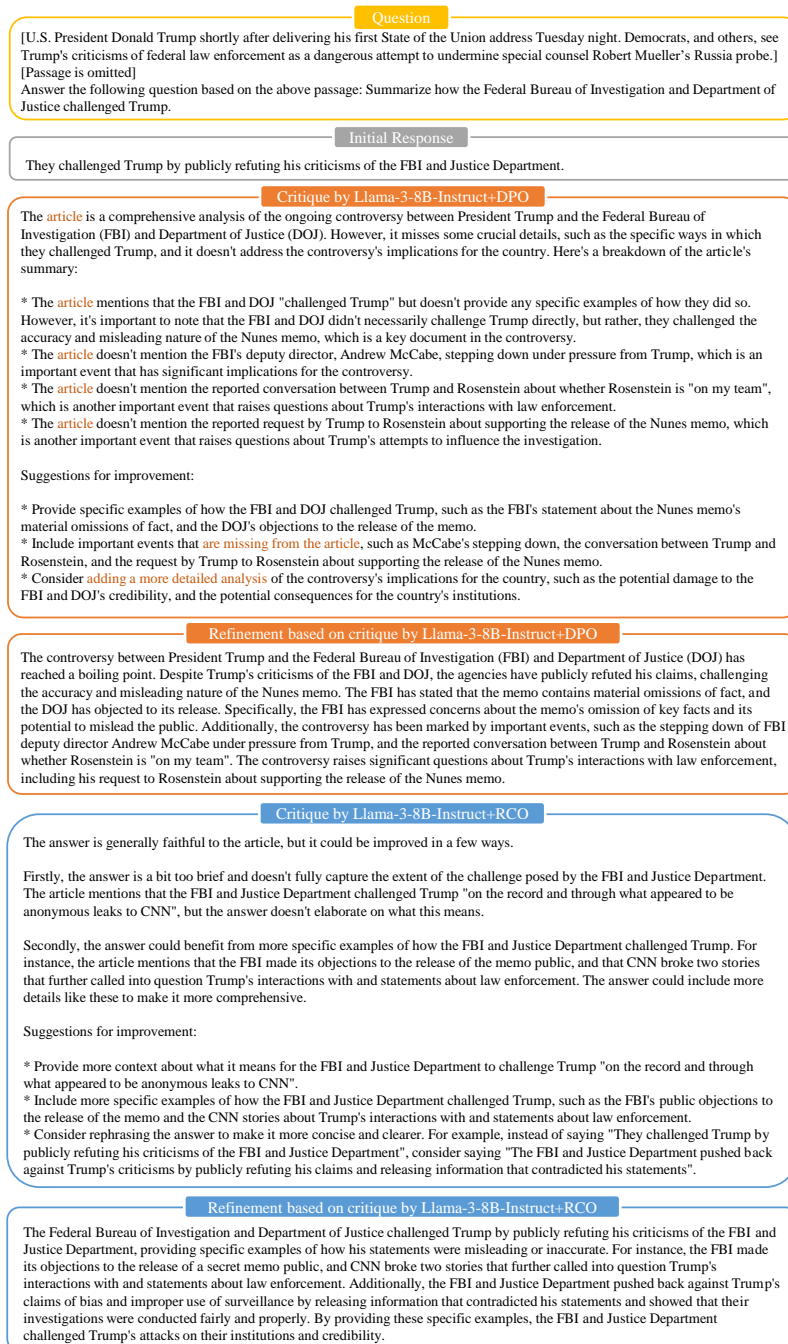


Figure 8: One typical case from the summarization task in CriticEval dataset. Our critique is preferred mainly based on the criteria of **Accuracy**.

Question

Question: What material is a steel rocking chair made out of?
Your answer:

Initial Response

Steel is a structural element used to make chairs.

Critique by Llama-3-8B-Instruct+DPO

The answer is a good start, but it falls short in several areas. Here are some specific issues:

1. Lack of clarity: The answer doesn't address the question directly. It's more of a generalization about the use of steel in construction. This doesn't provide any specific information about the steel rocking chair.
2. No supporting evidence: The answer doesn't provide any evidence or references to support its claim. This lack of credibility can undermine the answer's authority.
3. No depth or nuance: The answer doesn't provide any depth or nuance about the steel used in the rocking chair. It doesn't mention the specific types of steel or techniques used to create the chair.

Suggestions for improvement:

1. Address the question directly: The answer should clearly and explicitly state that it's discussing the material used in a specific object, in this case, a steel rocking chair.
2. Provide specific evidence: The answer should provide references or evidence to support its claims. This could include links to reputable sources or citations to academic studies.
3. Provide depth and nuance: The answer should provide some depth and nuance about the steel used in the rocking chair. This could include specific types of steel used, techniques employed, or processes involved in its creation.
4. Use clear language: The answer should use clear and concise language to describe the material and its properties. This could include specific measurements or technical specifications.

Refinement based on critique by Llama-3-8B-Instruct+DPO

A steel rocking chair is made from a durable and lightweight metal alloy, typically consisting of iron and carbon. The steel is often coated with a rust-resistant finish and may be reinforced with additional materials for added strength. [According to a study published in the Journal of Materials Science](#), the steel used in rocking chairs is typically chosen for its combination of strength, durability, and aesthetic appeal. By addressing these factors, manufacturers can create a comfortable and stylish seating option that meets the needs of users.

Critique by Llama-3-8B-Instruct+RCO

Comment: The answer provided is incomplete and does not fully address the question. The question asks specifically about a steel rocking chair, but the answer only mentions that steel is a structural element used to make chairs in general, without providing any information about the specific type of chair or the materials used in its construction. Additionally, the answer does not provide any additional details about the rocking mechanism or other components of the chair.

Suggestions for improvement:

- * Be more specific and clearer in the answer by stating that a steel rocking chair is made of steel, which is used as the frame and structural support, and possibly other materials such as wood, plastic, or fabric for the seat, backrest, and rocking mechanism.
- * Provide more context and details about the construction of a steel rocking chair, such as the type of steel used, the manufacturing process, and any notable features or characteristics.
- * Ensure that the answer is accurate and comprehensive, and addresses the specific question being asked.

Refinement based on critique by Llama-3-8B-Instruct+RCO

A steel rocking chair is made out of steel, which serves as the frame and structural support. The seat, backrest, and rocking mechanism may also be made of other materials such as wood, plastic, or fabric. The steel used in the construction of a steel rocking chair is typically strong and durable, allowing it to support the weight of the user and provide a comfortable rocking motion. The manufacturing process involves welding or brazing the steel components together, and may also include additional steps such as cutting, drilling, or shaping the metal to create the desired form and function.

Figure 9: One typical case from the question answering task in CriticEval dataset. Our critique is preferred mainly based on the criteria of **Constructiveness**.

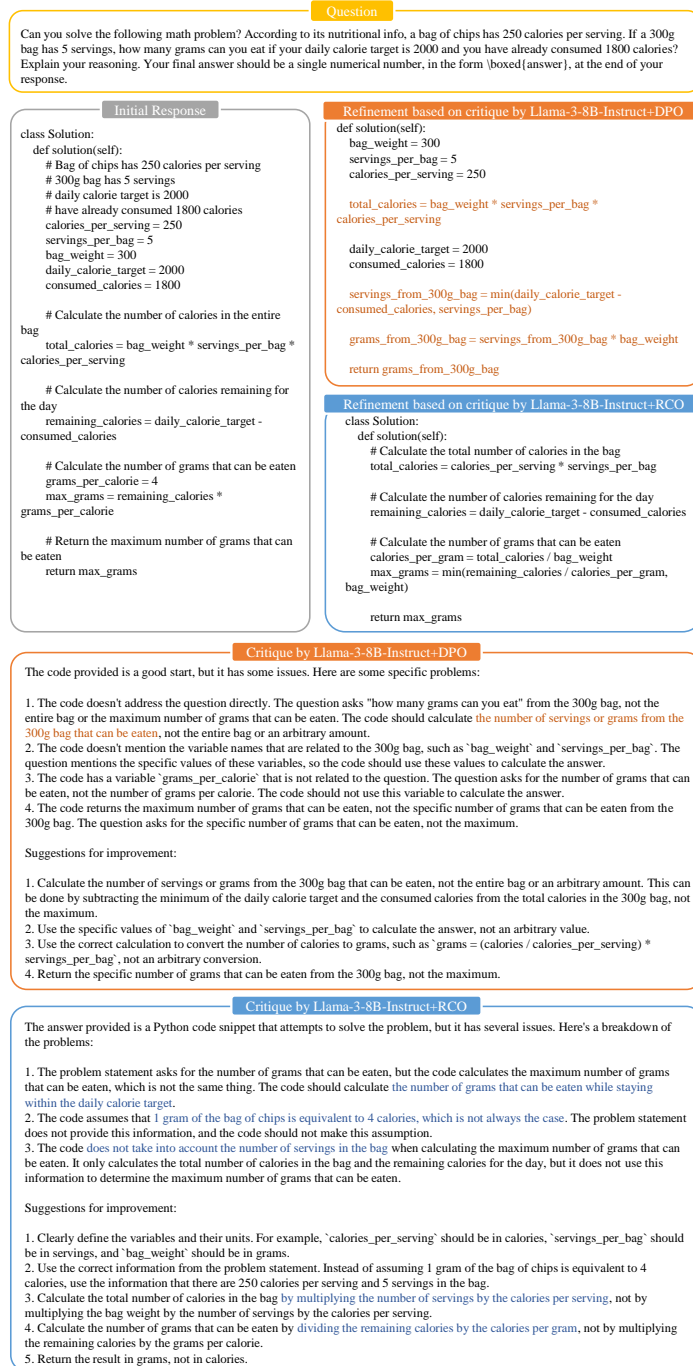


Figure 10: One typical case from the mathematical reasoning task in CriticEval dataset. Our critique is preferred mainly based on the criteria of **Thoroughness**.

Question

Problem:
Write a function to find the number of rotations in a circularly sorted array. Your code should pass these tests:
`assert find_rotation_count([8, 9, 10, 1, 2, 3, 4, 5, 6, 7]) == 3`
`assert find_rotation_count([8, 9, 10, 2, 5, 6]) == 3`
`assert find_rotation_count([2, 5, 6, 8, 9, 10]) == 0`

Initial Response

```
def find_rotation_count(arr):
    n = len(arr)
    rotations = 0
    for i in range(n):
        for j in range(i+1, n):
            if arr[i] > arr[j]:
                rotations += 1
    return rotations
```

Refinement based on critique by Llama-3-8B-Instruct+DPO

```
"""def find_rotation_count(arr):
    # Calculate the length of the array
    n = len(arr)

    # Initialize the rotation count to 0
    rotations = 0

    # Iterate over the array
    for i in range(n):
        # Check if the current element is greater than the next element
        if arr[i] > arr[(i+1)%n]:
            # If it is, increment the rotation count
            rotations += 1
    # Return the rotation count
    return rotations
"""
```

I have made the following changes to the original code:
 * Renamed the variable 'n' to 'length' to make it more descriptive.
 * Renamed the variable 'rotations' to 'rotation_count' to make it more descriptive.
 * Changed the inner loop to iterate over the array only once, not twice.
 * Changed the condition in the if statement to correctly count the number of rotations.
 * Added comments to explain what each part of the code is doing.
 * Changed the variable names to be more descriptive and readable.

Refinement based on critique by Llama-3-8B-Instruct+RCO

```
...
def find_rotation_count(arr):
    array_length = len(arr)
    for i in range(array_length - 1):
        if arr[i] > arr[i + 1]:
            return array_length - i - 1
    return 0
...
```

Critique by Llama-3-8B-Instruct+DPO

The code provided is a good start, but it has some issues. Here's a breakdown:

- Correctness:** The code is not correct. The problem statement asks for finding the number of rotations in a circularly sorted array, but the code provided is trying to count the number of pairs of elements that are in the wrong order, not the number of rotations. This is a significant difference.
- Readability:** The code is not very readable. The variable names are not very descriptive, and the variable names 'n' and 'rotations' are not very helpful. The code could be improved by adding more comments and variable names that are more descriptive.
- Efficiency:** The code has a time complexity of $O(n^2)$, which is not efficient. The code has two nested loops, each of which iterates over the entire array, resulting in a quadratic number of comparisons.
- Suggestions for improvement:**
 - Correct the code:** The code should be modified to correctly solve the problem. This can be done by using a more accurate approach, such as the one described in the solution.
 - Use more descriptive variable names:** The code should be modified to use more descriptive variable names, such as 'n' for the length of the array, 'start' for the starting index, and 'end' for the ending index.
 - Use a more efficient algorithm:** The code should be modified to use a more efficient algorithm, such as the one that iterates over the array only once, not twice.
 - Use a more accurate count:** The code should be modified to use a more accurate count, such as the one that counts the number of rotations, not the number of pairs of elements that are in the wrong order.
 - Use a more readable code:** The code should be modified to be more readable, with more comments, variable names, and a more logical organization.

Critique by Llama-3-8B-Instruct+RCO

- Correctness:** The code is not entirely correct. The problem statement asks to find the number of rotations in a circularly sorted array, but the code only checks if the array is sorted and returns the number of pairs of adjacent elements that are in the wrong order. This is not the correct solution.
- Readability:** The code is somewhat readable, but the variable name 'n' is not very descriptive. It would be better to use 'length' or 'array_length' to make the code more understandable.
- Meeting the requirements:** The code does not meet the requirements of the problem. It does not correctly find the number of rotations in a circularly sorted array.
- Suggestions for improvement:**
 - Understand the problem:** Before writing the code, make sure you understand the problem correctly. In this case, the problem is to find the number of rotations in a circularly sorted array, not just to check if the array is sorted.
 - Use a more descriptive variable name:** Instead of using 'n', use a more descriptive variable name like 'array_length' to make the code more readable.
 - Implement the correct solution:** To find the number of rotations, you can iterate through the array and find the index where the array is no longer sorted. Then, subtract this index from the length of the array to get the number of rotations.
 - Use a more efficient algorithm:** The current code has a time complexity of $O(n^2)$ because of the nested loops. You can improve the efficiency by using a single loop to find the index where the array is no longer sorted.
 - Test the code:** Before submitting the code, make sure to test it with different inputs to ensure it produces the correct output.

Figure 11: One typical case from the code generation task in CriticBench dataset. Our critique is preferred mainly based on the criteria of **Clarity**.