ARIES: STIMULATING SELF-REFINEMENT OF LARGE LANGUAGE MODELS WITH AND FOR ITERATIVE PREFERENCE OPTIMIZATION

Yongcheng Zeng[1,3] **Xuanfa Jin**[1,3] **Guoqing Liu**[4] **Quan He**[2] **Dong Li**[2]
Jianye Hao[2] **Haifeng Zhang**[1,3,†] **Jun Wang**[5,†]
[1]Institute of Automation, Chinese Academy of Sciences [2]Huawei Noah's Ark Lab, China
[3]School of Artificial Intelligence, University of Chinese Academy of Sciences
[4]University of Science and Technology of China [5]University College London

ABSTRACT

A truly intelligent Large Language Model (LLM) should be capable of correcting errors in its responses through external interactions. However, even the most advanced models often face challenges in improving their outputs. In this paper, we explore how to cultivate LLMs with the self-refinement capability through iterative preference training, and how this ability can be leveraged to improve model performance during inference. To this end, we introduce a novel post-training and inference framework, called **ARIES**: Adaptive Refinement and Iterative Enhancement Structure. This method iteratively performs preference training and self-refinement-based data collection. During training, ARIES strengthen the model's direct question-answering capability while simultaneously unlocking its self-refinement potential. During inference, ARIES harnesses this self-refinement capability to generate a series of progressively refined responses, which are then filtered using either the Reward Model Scoring or a simple yet effective Rule-Based Selection mechanism, specifically tailored to our approach, to construct a dataset for the next round of preference training. Experimental results demonstrate the remarkable performance of ARIES. When applied to the Llama-3.1-8B model and under the self-refinement setting, ARIES surpasses powerful models such as GPT-4o, achieving 62.3% length-controlled (LC) and a 63.3% raw win rates on AlpacaEval 2, outperforming Iterative DPO by 27.8% and 35.5% respectively, as well as a 50.3% win rate on Arena-Hard, surpassing Iterative DPO by 26.6%. Furthermore, ARIES consistently enhances performance on mathematical reasoning tasks like GSM8K and MATH.

1 INTRODUCTION

Reinforcement Learning from Human Feedback (RLHF) has been demonstrated as an effective pathway to enhance Large Language Models' performance across downstream tasks (Ouyang et al., 2022; Bai et al., 2022a). The classic RLHF approach utilizes the PPO (Schulman et al., 2017) algorithm to train Large Language Models (LLMs) for alignment, but its major drawback is the substantial resource overhead. As an alternative, a more efficient and simple method directly optimizes the LLM itself (Rafailov et al., 2023; Azar et al., 2024; Ethayarajh et al., 2024; Zeng et al., 2024; Meng et al., 2024; Hong et al., 2024). However, both approaches critically depend on access to high-quality, human-annotated preference datasets. As these datasets become increasingly scarce, the challenge of identifying novel strategies to further enhance LLM capabilities becomes increasingly urgent.

Recent research highlights that LLMs themselves can serve as a substitute for human feedback, reducing the reliance on human annotations while significantly increasing the automation of the training process (Lee et al.; Wang et al., 2022; Yuan et al., 2024; Dubois et al., 2024b; Li et al., 2023). To further improve LLM performance, the multiple parallel sampling strategy is integrated with

†Correspondence to: Jun Wang <jun.wang@cs.ucl.ac.uk>, Haifeng Zhang <haifeng.zhang@ia.ac.cn>.

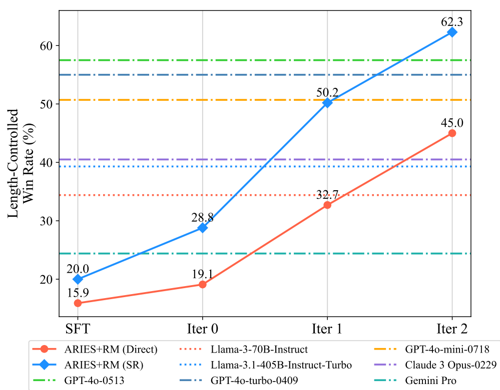

Figure 1: Length-controlled win rate on AlpacaEval 2 improves with ARIES iterations, surpassing GPT-4 level for the base versions of Llama-3.1-8B when utilizing the self-refinement strategy.

Model	Size	LC(%)	WR(%)
Llama-3.1-8B-Base-ARIES ($iter2$, SR)	8B	62.3	63.3
GPT-4o (05/13)	-	57.5	51.3
GPT-4-turbo (04/09)	-	55.0	46.1
GPT-4o-mini (07/18)	-	50.7	44.7
Llama-3.1-8B-Base-ARIES ($iter1$, SR)	8B	50.2	49.9
GPT-4_1106_preview	-	50.0	50.0
Llama-3.1-8B-Base-ARIES ($iter2$, Direct)	8B	45.0	46.8
Claude 3 Opus (02/29)	-	40.5	29.1
Llama-3.1-405B-Instruct-Turbo	405B	39.3	39.1
Qwen2-72B-Instruct	72B	38.1	29.9
Llama-3-70B-Instruct	70B	34.4	33.2
Llama-3.1-8B-Base-ARIES ($iter1$, Direct)	8B	32.7	33.5
Mistral Large (24/02)	123B	32.7	21.4
Gemini Pro	-	24.4	18.2
Llama-3.1-8B-Instruct	8B	20.9	21.8

Table 1: Results on AlpacaEval 2 leaderboard. LC and WR represent length-controlled and raw win rate, respectively. "Direct" refers to the direct response generation strategy, while "SR" denotes the self-refinement generation strategy.

self-evaluation mechanisms to curate high-quality datasets for downstream tasks (Yuan et al., 2024; Wu et al., 2024).

In this work, we propose an alternative approach to dataset generation inspired by the human problem-solving process, which starts with an initial draft and is refined iteratively. In this iterative process, self-refinement plays a critical role in progressively enhancing the quality of the answer. This prompts us to consider whether the self-refinement capability of LLMs can be harnessed specifically to generate higher-quality datasets. Unlike methods that rely on parallel sampling, sequential generation allows models to build on previous responses, fostering deeper reflection and resulting in datasets with greater intellectual depth (Snell et al., 2024; Qu et al., 2024; Kumar et al., 2024).

However, preliminary results reveal that smaller models often exhibit weak self-refinement, which can even degrade performance. This suggests that models aligned exclusively through RLHF like DPO, struggle with this generation strategy, as shown in Figure 3. To address this, we introduce a novel algorithm that gradually instills the self-refinement capability in the model. Afterward, we leverage the model's activated self-refinement ability to collect higher-quality datasets, thereby enabling such iterative training.

In conclusion, we present the Adaptive Refinement and Iterative Enhancement Structure (ARIES), a novel framework for iterative preference training. ARIES introduces a plug-in algorithm that complements existing DPO-like methods, fostering the development of self-refinement capability in LLMs. Building upon this, we leverage the self-refinement ability of LLMs to autonomously generate high-quality preference datasets, enabling an iterative training process that harmonizes both training and inference. Through rigorous empirical evaluations, ARIES demonstrates remarkable efficacy, achieving state-of-the-art performance across a variety of benchmark datasets, including AlpacaEval 2 (Li et al., 2023), Arena-Hard (Li et al., 2024), and MT-Bench (Zheng et al., 2023). As shown in Table 1, ARIES employs the self-refinement generation strategy to achieve a length-controlled win rate of 62.3% and a raw win rate of 63.3% on AlpacaEval 2, alongside a win rate of 50.3% on Arena-Hard, underscoring the significant impact of our approach in boosting the performance of LLMs.

2 Related Works

Reinforcement Learning from Human Feedback (RLHF) RLHF has proven effective in aligning LLMs with human values Christiano et al. (2017); Ouyang et al. (2022); Bai et al. (2022a); Song et al. (2023); Touvron et al. (2023). This approach uses human-annotated preference datasets to train a Reward Model, guiding LLM optimization through reinforcement learning. However, due to the high cost of human annotations, AI-generated feedback has been proposed to automate this process (Bai et al., 2022b; Lee et al., 2023). Additionally, to reduce training costs, Direct Preference Optimization (DPO) Rafailov et al. (2023) bypasses the reward modeling process and directly aligns LLMs using preference datasets. However, the effectiveness of these methods heavily depends on the quality of the preference dataset, making the acquisition of high-quality preference data a critical challenge.

Improving LLMs via Data Augmentation As high-quality human-labeled datasets become scarcer, methods leveraging LLMs to generate training data have emerged. Some approaches use powerful LLMs to generate datasets, which are then distilled into smaller models, enabling weaker LLMs to perform better with high-quality data (Chen et al., 2023). Other methods focus on self-generating problems or responses to expand datasets (Wang et al., 2022). Additionally, *LLM-as-a-Judge* prompting has become a standard technique for evaluating model outputs and optimizing datasets (Dubois et al., 2024b; Bai et al., 2024; Saha et al., 2023; Yuan et al., 2024). These advancements offer valuable strategies for efficient data augmentation and optimization.

In-Context Learning (ICL) ICL has become a fundamental capability of LLMs, enabling them to perform tasks by conditioning on a few input examples without requiring parameter updates (Brown et al., 2020). Recent studies, such as OPRO (Yang et al., 2024), show that LLMs can leverage their ICL abilities to function as implicit optimizers, progressively improving performance on complex problems. LLMs can also act as in-context reinforcement learners, optimizing behavior via reward feedback (Monea et al., 2024). The SELF-REFINE (Madaan et al., 2024) is a special form of ICL. It significantly enhances model performance through the FEEDBACK and REFINE mechanisms, achieving remarkable results across multiple benchmarks. These findings indicate that integrating ICL with model training presents a compelling strategy for constructing self-optimizing frameworks.

3 Preliminaries

DPO The standard DPO algorithm usually involves two stages: (1) Supervised Fine-Tuning (SFT) and (2) DPO training.

In the SFT stage, the DPO algorithm fine-tunes a pre-trained language model π_θ with the loss function defined as:

$$\mathcal{L}_{\mathrm{SFT}}(\pi_\theta; \pi_{\mathrm{ref}}) = -\mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}}[\log \pi_\theta(y|x)], \tag{1}$$

which yields a fine-tuned model, denoted as $\pi_{\mathrm{ref}} = \pi_{\mathrm{SFT}}$, to be used as the reference model in subsequent stages. If given a preference dataset $\mathcal{D} = \{x^{(i)}, y_w^{(i)}, y_l^{(i)}\}_{i=1}^N$ for fine-tuning, the chosen response y_w is typically selected as the target for SFT. In this case, the loss becomes:

$$\mathcal{L}_{\mathrm{SFT}}(\pi_\theta; \pi_{\mathrm{ref}}) = -\mathop{\mathbb{E}}_{(x,y_w,y_l)\sim\mathcal{D}}[\log \pi_\theta(y_w|x)]. \tag{2}$$

In the DPO preference training stage, DPO uses the fine-tuned model as initialization, i.e., $\pi_\theta = \pi_{\mathrm{SFT}}$, and directly optimizes the policy model using the following negative log-likelihood loss:

$$\mathcal{L}_{\mathrm{DPO}}(\pi_\theta; \pi_{\mathrm{ref}}) = \mathop{\mathbb{E}}_{(x,y_w,y_l)\sim\mathcal{D}}\left[\log \sigma\left(\beta_{\mathrm{DPO}} \log \frac{\pi_\theta(y_w \mid x)}{\pi_{\mathrm{ref}}(y_w \mid x)} - \beta_{\mathrm{DPO}} \log \frac{\pi_\theta(y_l \mid x)}{\pi_{\mathrm{ref}}(y_l \mid x)}\right)\right]. \tag{3}$$

Self-Refinement In language generation, given a question input x, the model typically generates a response $y \sim \pi(\cdot|x)$. We refer to this approach as *direct response generation* in this paper. Building on this, our work advances further by assuming that we are given both a question input x and an existing response y_1. Our goal is to generate a better response $y_2 \sim \pi(\cdot|x, y_1)$, where this generation method is termed as *Self-Refinement* in our work.

4 Methodology

How to enable large models to correct their responses when provided with additional information is both an intriguing and valuable challenge. However, existing research indicates that even the most advanced models struggle with self-improvement, often failing to refine prior outputs effectively (Huang et al., 2023). In some cases, repeated correction attempts can even diminish the models' confidence in their responses, resulting in progressively worse revisions, as shown in Figure 3. In this section, we investigate how to stimulate the model's self-refinement capability within the RLHF framework, and how leveraging the self-refinement capability can, in turn, enhance overall model training performance. We will elaborate on our approach, ARIES, covering both the training methods and the inference framework. Finally, we will summarize the complete workflow of the proposed framework.

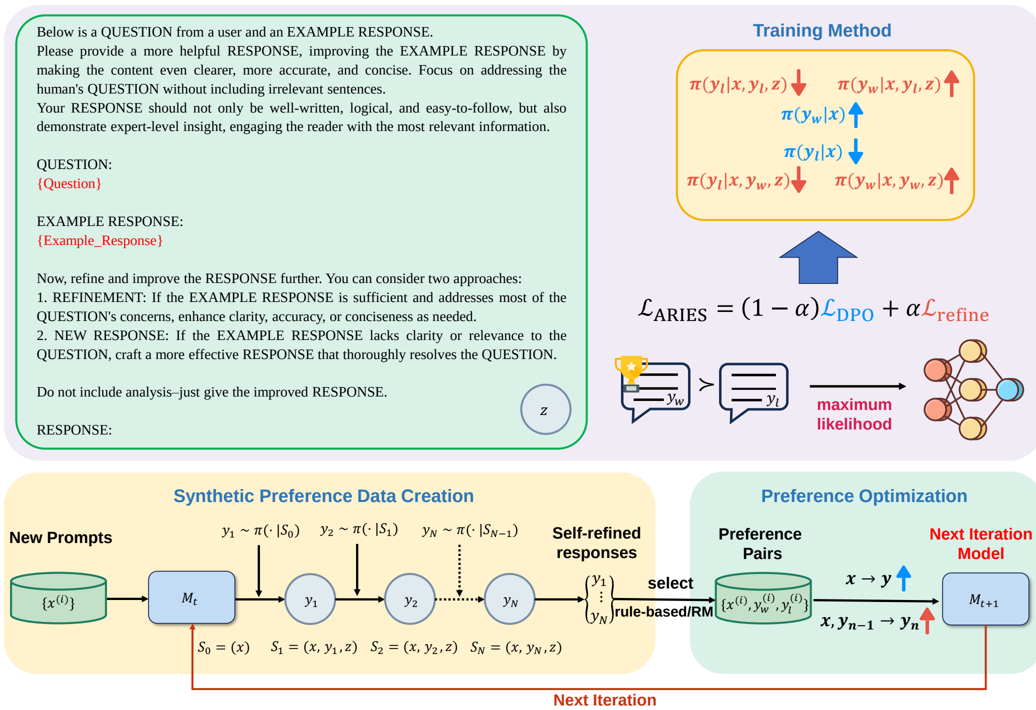

Figure 2: **ARIES: Adaptive Refinement and Iterative Enhancement Structure.** Our method iteratively alternates between inference and training processes. In the inference phase, we utilize the model M_t from the previous training round to generate a series of self-refined responses to the prompt dataset through the self-refinement strategy. These responses are then filtered using either Rule-Based Selection or Reward Model Scoring mechanism to construct a preference dataset for training. In the training phase, we apply a novel preference-based training algorithm to train the model M_t using the collected preference dataset. This algorithm primarily focuses on activating and strengthening the model's self-refinement ability to assist in generation during the inference phase. The top-left template is our self-refinement template, used in both the training and inference stages.

4.1 Training: Plug-in Extension for DPO Framework

Considering that existing models generally struggle to achieve self-improvement, the primary challenge we aim to address is how to enable models to undergo effective self-refinement. Conventional RLHF algorithms, such as DPO, do not inherently provide this capability. Therefore, we extend the DPO framework, incorporating both Supervised Fine-Tuning (SFT) and preference training, to empower models with the self-refinement ability.

Supervised Fine-Tuning (SFT) During the SFT stage, in addition to the standard Negative Log-Likelihood Loss, we seek to enhance the model's ability to refine suboptimal responses. Specifically, we aim to improve the model's ability to assign a higher probability to a good response y_w given the input x, the existing suboptimal response y_l and a special refinement template z. Thus, the loss function at the SFT stage is defined as follows:

$$\mathcal{L}_{\text{ARIES-SFT}}(\pi_\theta; \pi_{\text{ref}}) = -\mathop{\mathbb{E}}_{(x, y_w, y_l)\sim\mathcal{D}}[\log \pi_\theta(y_w|x) + \log \pi_\theta(y_w|x, y_l, z)]. \tag{4}$$

Preference Training In the preference training stage, we begin by presenting the following scenario: Given a prompt x, assume we have already sampled an initial response y_1. How can we utilize the prompt x and the response y_1 to generate a more refined response y_2 guided by a refinement template z? To achieve this, we define the following objective function:

$$\max_{\pi} \mathop{\mathbb{E}}_{y_2\sim\pi(\cdot|x, y_1, z)} \Big[p(y_2 \succ y_1|x) - \beta D_{\text{KL}}(\pi||\pi_{\text{ref}}|x, y_1, z) \Big]. \tag{5}$$

Here, $p(y_2 \succ y_1|x)$ represents the human preference probability, indicating the likelihood that y_2 is preferred over y_1 given x:

$$p(y_2 \succ y_1 \mid x) = \mathbb{E}_h[\mathbb{I}\{h \text{ prefers } y_2 \text{ over } y_1 \text{ given } x\}], \tag{6}$$

where the expectation is taken over the distribution of humans h. The optimization process of the objective function in Eq. 5 follows a similar approach to SRPO (Choi et al., 2024). Solving Eq. 5 yields the following equality:

$$p(y_2 \succ y_1|x) - \frac{1}{2} = \beta \log \frac{\pi^*(y_2|x, y_1, z))}{\pi_{\text{ref}}(y_2|x, y_1, z))} - \beta \log \frac{\pi^*(y_1|x, y_1, z))}{\pi_{\text{ref}}(y_1|x, y_1, z))}. \tag{7}$$

The derivation is presented in Appendix A.1. According to Eq. 7, we adopt the mean squared error as the loss function and parametrize the policy model as π_θ, iterating over all prompts x and responses y_1, y_2, which leads to:

$$\mathcal{L}(\pi_\theta; \pi_{\text{ref}}) = \mathop{\mathbb{E}}_{(x,y_1,y_2)\sim\rho}\left[p(y_2 \succ y_1|x) - \frac{1}{2} - \beta\left[\log\left(\frac{\pi_\theta(y_2|x, y_1, z)}{\pi_{\text{ref}}(y_2|x, y_1, z)}\right) - \log\left(\frac{\pi_\theta(y_1|x, y_1, z)}{\pi_{\text{ref}}(y_1|x, y_1, z)}\right)\right]\right]^2, \tag{8}$$

where ρ represents the true distribution. We substitute Eq. 6 into Eq. 8. For a well-curated preference dataset $\mathcal{D} = \{(x^{(i)}, y_w^{(i)}, y_l^{(i)})\}_{i=1}^N$, we obtain the following loss function:

$$\begin{aligned}\mathcal{L}_{\text{refine}}(\pi_\theta; \pi_{\text{ref}}) = &\mathop{\mathbb{E}}_{(x,y_w,y_l)\sim\mathcal{D}}\left[\frac{1}{2} - \beta\left[\log\left(\frac{\pi_\theta(y_w|x, y_l, z)}{\pi_{\text{ref}}(y_w|x, y_l, z))}\right) - \log\left(\frac{\pi_\theta(y_l|x, y_l, z)}{\pi_{\text{ref}}(y_l|x, y_l, z)}\right)\right]\right]^2 \\ &+ \mathop{\mathbb{E}}_{(x,y_w,y_l)\sim\mathcal{D}}\left[\frac{1}{2} - \beta\left[\log\left(\frac{\pi_\theta(y_w|x, y_w, z)}{\pi_{\text{ref}}(y_w|x, y_w, z))}\right) - \log\left(\frac{\pi_\theta(y_l|x, y_w, z)}{\pi_{\text{ref}}(y_l|x, y_w, z)}\right)\right]\right]^2.\end{aligned} \tag{9}$$

Finally, we combine the DPO loss and the self-refinement loss to derive our final loss function:

$$\mathcal{L}_{\text{ARIES}}(\pi_\theta; \pi_{\text{ref}}) = (1 - \alpha)\mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) + \alpha\mathcal{L}_{\text{refine}}(\pi_\theta; \pi_{\text{ref}}). \tag{10}$$

Through this optimization, we not only enhance the direct conversational capabilities of LLMs but also strengthen the models' self-refinement ability. Notably, our approach does not impose any restrictions on the choice of RLHF algorithm. In other words, any RLHF algorithm's loss function can be substituted into Eq. 10 to replace the DPO loss term.

It is essential to emphasize that the effectiveness of our approach, ARIES, does not stem from a particular refinement algorithm, such as Eq. 9, but rather from the fundamental refinement principle it encapsulates. In Appendix C, we introduce a new refinement loss function derived from the perspective of the Bradley-Terry model, which we designate as BT_ARIES. Experimental results show that both BT_ARIES and ARIES yield comparable effectiveness, driving substantial improvements in model performance. For further details, please refer to Appendix C.

4.2 Inference: Self-Refinement for Data Collection

Response Refinement During the inference phase, we exploit the self-refinement capability instilled in the model during training to generate a sequence of progressively refined responses to the new prompt dataset, as illustrated in Figure 2. This iterative process mirrors human-like problem-solving, where the model revisits the same question multiple times, refining its understanding and improving the quality of its responses. By using the response from the previous iteration as a prior, the model incrementally enhances its performance, ensuring that each successive turn builds upon the insights of the last.

Specifically, given a problem x, we first use the model to directly generate an answer y_1. Next, we apply a fixed refinement template z that concatenates the problem x and the answer y_1, with the objective of improving the answer's quality. The model is then provided with two options for refinement:

1. If the model believes that the existing answer is sufficient, it only needs to modify the given answer to further enhance its quality.
2. If the model finds the existing answer lacking in clarity or relevance to the problem, it disregards the previous answer and directly generates a more effective new response.

This approach grants the model some degree of backtracking, preventing it from repeatedly getting stuck in a dead-end that could lead to performance degradation. The refinement template is illustrated in Figure 2. By applying the refinement template iteratively, we generate y_2 from x and y_1, then y_3 from x and y_2. This process continues for N rounds, where each answer y_n is generated based on the given problem x and the previous round's answer y_{n-1}:

$$y_n \sim \pi(\cdot|x, y_{n-1}) \tag{11}$$

Through this iterative generation process, we obtain a progressively refined set of responses for the problem x, denoted as $\{x, y_1, y_2, \ldots, y_N\}$.

Data Selection Given a new prompt dataset $\mathcal{D} = \{x^{(i)}\}_{i=1}^N$, through the iterative self-refinement process described above, we can obtain a self-refined prompt-responses dataset $\mathcal{D} = \{(x^{(i)}, y_1^{(i)}, y_2^{(i)}, \ldots, y_N^{(i)})\}_{i=1}^N$. However, not every round of model generation improves the response quality. Therefore, we must implement a selection mechanism to filter the data, enhancing the robustness of the final preference dataset. We primarily explore two data filtering methods: **Rule-Based Selection** and **Reward Model Scoring**. The Rule-Based Selection mechanism is an intriguing approach that relies purely on the model's self-refinement capability without any external supervision. Specifically, we directly select the responses from round 0 (i.e., the initial answers to the questions) as rejected responses, while empirically designating the responses from round N as chosen responses to construct the preference dataset. This mechanism plays a crucial role in validating the effectiveness of our proposed technique, highlighting the remarkable efficacy of ARIES.

4.3 WORKFLOW: STIMULATING MODEL SELF-REFINEMENT

The goal of our work is to activate the model's self-refinement ability and leverage this capability to assist the model in achieving improvement during the iterative training. Since current open-source models face challenges in enhancing their own responses, our first task is to trigger the model's self-refinement ability using our proposed algorithm. Starting from a base model, we perform two main steps in this phase:

1. Apply the ARIES-SFT loss function, as defined in Eq. 4, to Supervise Fine-Tuning the base model, resulting in the fine-tuned model;
2. Perform preference training on the fine-tuned model with an offline preference dataset and the loss function Eq. 10, obtaining the ARIES *offline* model.

The reason for conducting preference training on the ARIES-SFT model with the offline preference dataset is that we observe this process further activates the model's self-refinement capability, as shown in Figure 3.

In subsequent phases, we iteratively perform data collection and preference training to continuously improve the model's performance. Specifically, we employ our inference framework and data filtering mechanism to assemble a high-quality preference dataset, which is then utilized for the next round of preference training. In each iteration, the model not only enhances its direct problem-solving capabilities but also strengthens its self-refinement ability. The synergistic evolution of both aspects ensures the efficient operation of our inference framework, while progressively improving the quality of the collected datasets, leading to a continuous performance boost and ultimately achieving a significant breakthrough in model performance.

5 EXPERIMENTS

In this section, we will demonstrate the effectiveness of ARIES in gradually instilling models to self-improve their responses over turns. Specifically, we will primarily investigate the following questions: (1) How significantly does ARIES improve model performance compared to prior methods during the iterative process of data collection and training (§5.1)? (2) What components contribute to the performance improvements of ARIES (§5.2)? (3) Can the self-refinement strategy induced by ARIES generalize to problems out of the training domains (§5.3)? By answering these questions, we aim to highlight the effectiveness, generality, and underlying mechanisms of ARIES.

Models and Training Settings We primarily focus on preference optimization and capability analysis of Llama-3.1-8B Base (Dubey et al., 2024). During the SFT phase, we fine-tune Llama-3.1-8B Base

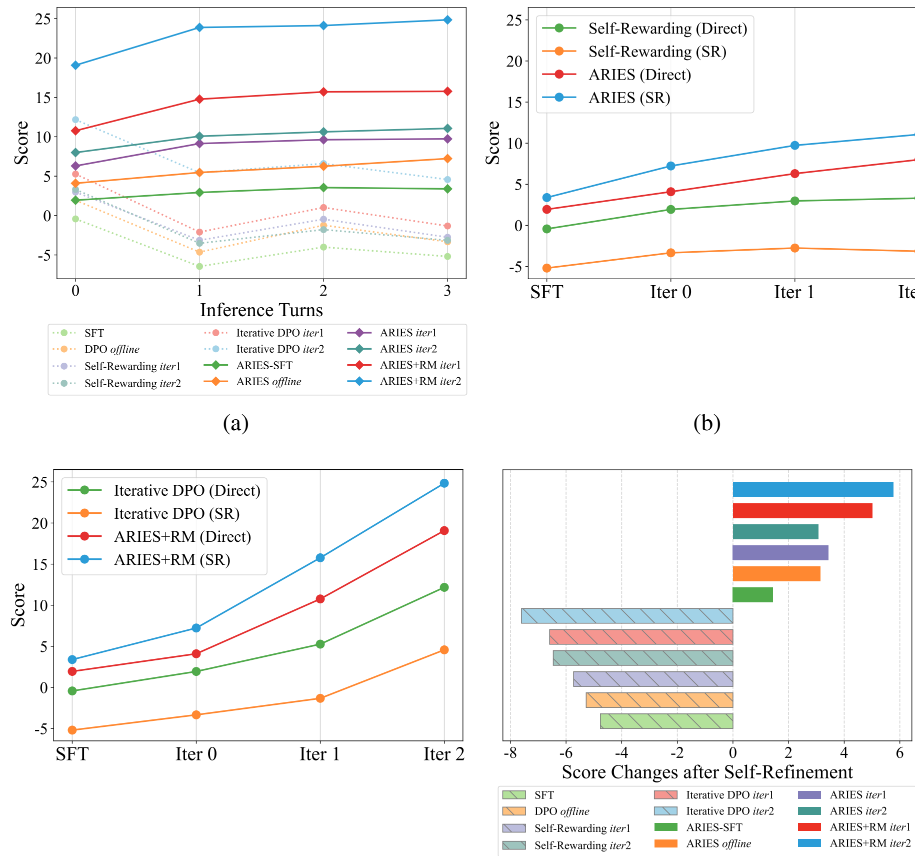

Figure 3: Qualitative analysis of various methods on the UltraFeedback test set. We use the reward model Skywork/Skywork-Reward-Llama-3.1-8B-v0.2 (Liu et al., 2024) for scoring, which is a relatively small model but demonstrates strong performance on the RewardBench leaderboard (Lambert et al., 2024b). (a) shows the variation in model performance as the number of inference turns increases. (b) illustrates the performance gains of different methods during the iterative training process without external supervision signals. (c) depicts the performance improvement during the iterative training process with Reward Model Scoring. (d) demonstrates the performance improvement after 3 rounds of self-refinement.

on the llama-3.1-tulu-3-70b-preference-mixture dataset (Lambert et al., 2024a) using Eq. 4, yielding the ARIES-SFT model. Subsequently, we use the first 30K preference data from the UltraFeedback dataset (Cui et al., 2023) and apply preference training on top of the ARIES-SFT model, guided by Eq. 10, resulting in the ARIES *offline* model. Next, we extract a new 5K prompt dataset from the UltraFeedback dataset and generate responses by iterating 4 times per prompt using the ARIES *offline* model and our inference framework, forming a dataset $\mathcal{D} = \{x^{(i)}, y_1^{(i)}, y_2^{(i)}, y_3^{(i)}, y_4^{(i)}\}$. After applying a data filtering mechanism, we construct a new preference dataset, which is used for the next round of preference training to obtain the ARIES *iter*1 model. In the data filtering process, for the Rule-Based Selection mechanism, we directly select the response y_1 from round 0 as the rejected response, while empirically treating the response after 3 rounds of self-refinement, i.e., y_4, as the chosen response. For the Reward Model Scoring mechanism, we employ Skywork/Skywork-Reward-Llama-3.1-8B-v0.2 (Liu et al., 2024) to score the generated responses, selecting the highest-scoring response as the chosen response and the lowest-scoring response as the rejected response. Finally, we extract another 10K prompt dataset from the UltraFeedback dataset, repeat the data generation and preference training process, producing the ARIES *iter*2 model. For convenience, we denote the results trained with the Rule-Based Selection dataset filtering mechanism as **ARIES**, and the results trained with the Reward Model Scoring mechanism as **ARIES+RM** in the following experiment.

Table 2: Results on AlpacaEval 2 (Li et al., 2023), Arena-Hard (Li et al., 2024), and MT-Bench (Zheng et al., 2023) under the **Direct** and **SR** settings. Here, **Direct** denotes direct response generation, while **SR** indicates three rounds of self-refinement on the responses. LC and WR represent length-controlled win rate and raw win rate, respectively.

Method	**Direct**					**Self-Refinement (SR)**		
	AlpacaEval 2		**Arena-Hard**	**MT-Bench**		**AlpacaEval 2**		**Arena-Hard**
	LC (%)	WR (%)	WR (%)	GPT-4 Turbo	GPT-4	LC (%)	WR (%)	WR (%)
SFT	15.9	12.7	12.7	6.4	6.9	13.8	8.1	8.0
DPO *offline*	17.9	16.7	16.5	6.9	7.4	18.3	12.6	12.6
Self-Rewarding *iter*1	19.3	17.2	14.2	6.9	7.5	19.3	12.6	11.6
Self-Rewarding *iter*2	18.2	14.2	15.9	6.7	7.4	19.0	11.2	11.0
Iterative DPO *iter*1	24.6	22.3	22.4	6.9	7.6	25.1	16.9	17.1
Iterative DPO *iter*2	34.1	33.5	29.6	7.1	7.8	34.5	27.8	23.7
ARIES-SFT	15.9	15.5	16.5	6.4	7.0	20.0	18.6	18.0
ARIES *offline*	19.1	18.6	17.4	7.0	7.6	28.8	27.1	23.5
ARIES *iter*1	23.9	24.5	22.0	7.1	7.5	37.4	35.7	31.1
ARIES *iter*2	28.4	29.7	24.9	7.1	7.7	41.3	39.5	32.0
ARIES+RM *iter*1	32.7	33.5	31.9	7.3	7.7	50.2	49.9	37.5
ARIES+RM *iter*2	**45.0**	**46.8**	**38.0**	**7.7**	**8.1**	**62.3**	**63.3**	**50.3**

Baselines To evaluate the effectiveness of our proposed preference optimization approach, we compare it with other preference optimization methods. Specifically, we replicate Self-Rewarding (Yuan et al., 2024) and Iterative DPO (Snorkel, 2024; Xu et al., 2023; Xiong et al., 2023; Dong et al., 2024) as the baseline. These methods leverage the parallel sampling mechanism to iteratively generate data, followed by data filtering to construct a preference dataset, which is then used for the next round of training. Self-Rewarding employs an *LLM-as-a-Judge* mechanism (Zheng et al., 2023) for data filtering, while Iterative DPO relies on reward model scoring for selection. Similarly to our approach, for the reward model scoring mechanism, we utilize Skywork/Skywork-Reward-Llama-3.1-8B-v0.2 to score the generated responses and construct the preference dataset. For a fair comparison, we follow the same training procedure as our method. Both Self-Rewarding and Iterative DPO conduct two rounds of online preference training based on the DPO *offline* model.

Evaluation Benchmarks We evaluate our models using three most popular open-ended instruction-following benchmarks: AlpacaEval 2 (Li et al., 2023), MT-Bench (Zheng et al., 2023), and Arena-Hard (Li et al., 2024), along with two cross-domain mathematical reasoning tasks: GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021). For AlpacaEval 2, we provide the raw win rate (WR) and length-controlled win rate (LC) (Dubois et al., 2024a). For MT-Bench, we report the average score using GPT-4 and GPT-4-Preview-1106 as judges. For Arena-Hard, we report the win rate relative to the baseline model. For GSM8K and MATH, we use the Step-DPO evaluation script. Specifically, we evaluate the entire GSM8K test set, which contains 1319 math problems, and for MATH, we evaluate the first 1024 samples of the test set. Since MT-Bench is a multi-turn dialogue dataset, all evaluations, except for MT-Bench, assess both direct response generation (labeled as "**Direct**") and responses refined through 3 rounds of self-refinement using our inference framework (denoted as "**SR**").

5.1 Boost in Model Performance with ARIES

We first evaluate the performance improvements on the UltraFeedback test set, focusing on both the inference phase and the iterative training process, as shown in Figure 3. In Figure 3(a), we observe that the models trained with ARIES consistently demonstrate gradual improvements during inference, while models trained by self-rewarding and iterative DPO struggle to enhance response quality. This highlights that ARIES effectively stimulates the self-refinement capability of LLMs. In multi-round interactions, the models trained with ARIES achieve incremental performance gains. Figure 3(b) and Figure 3(c) show that, compared to baseline methods, ARIES significantly enhances the model's ability to directly answer questions and self-refine previous responses, even without external reward signals. This demonstrates the superior efficacy of ARIES in improving overall model performance. Notably, while models trained with other methods exhibit a decline in self-refinement, we find that the models' self-refinement abilities positively correlate with their base performance in

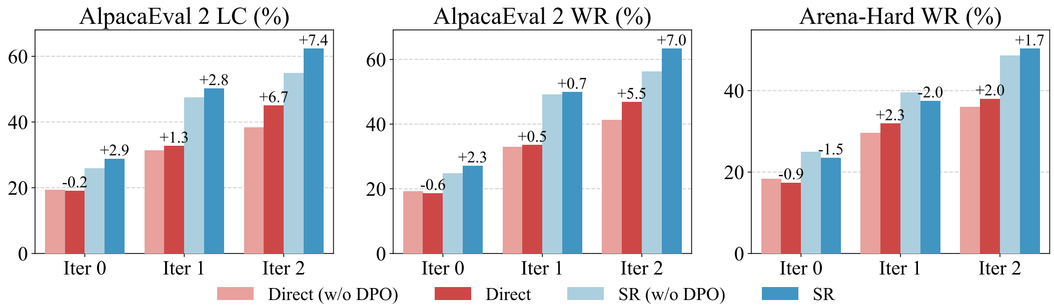

Figure 4: Impact analysis of DPO Loss on ARIES Performance under the Reward Model Scoring mechanism.

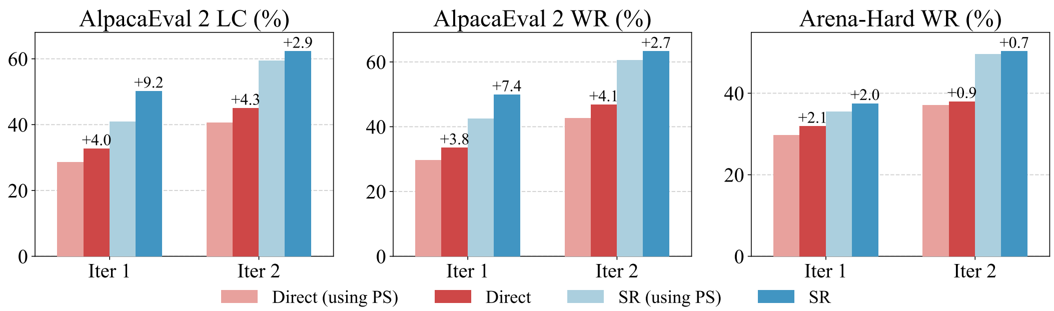

Figure 5: Effect of Parallel Sampling (PS) vs. Self-Refinement in preference dataset construction on ARIES Performance under the Reward Model Scoring mechanism.

direct question answering. As iterative training progresses, models trained with alternative methods also show slow improvements in self-refinement. Finally, in Figure 3(d), we find that under the Reward Model Scoring mechanism, ARIES's self-refinement gain improves progressively during the iterative training process, compared to direct response generation. This indicates that ARIES enables the models to gradually master this cognitive mode. In contrast, under the Rule-Based Selection mechanism, the self-refinement gains of ARIES *iter*2 are lower than those of ARIES *iter*1. This decline can be attributed to the limitations of rule-based data filtering, which inevitably introduces noise into the collected dataset, ultimately impacting the model's performance.

Subsequently, we evaluate the model performance on AlpacaEval 2, Arena-Hard, and MT-Bench, as demonstrated in Table 2. Our method not only significantly improves model performance under the "**Direct**" setting but also achieves a qualitative leap in performance under the "**SR**" setting. In contrast, other baseline methods exhibit varying degrees of performance degradation after applying self-refinement. Under the "**Direct**" setting, ARIES+RM *iter*2 outperforms iterative DPO *iter*2 by 10.9% in LC win rate and 13.3% in raw win rate on AlpacaEval 2, while achieving an 8.4% improvement on Arena-Hard. In the "**SR**" setting, ARIES+RM *iter*2 shows a 27.8% improvement in LC win rate and a 35.5% improvement in raw win rate on AlpacaEval 2, with a 26.6% gain on Arena-Hard. With self-refinement, ARIES+RM *iter*2 reaches performance levels comparable to GPT-4 on these benchmarks.

5.2 Ablation Studies of ARIES

Here, we examine the impact of two key aspects on ARIES: (1) whether the training algorithm includes DPO loss (i.e., $\alpha = 0$), and (2) replacing the self-refinement sequential generation strategy with Parallel Sampling (PS) for data collection during the preference dataset construction process. These results are presented in Figure 4 and Figure 5.

In Figure 4, we observe that omitting DPO loss progressively affects the performance of ARIES. As iterative training advances, the absence of DPO loss leads to a slower rate of improvement in both "**Direct**" and "**SR**" results. For example, the performance of direct answers decreases by 6.7% LC win rate, while self-refined answers drop by 7.4% LC win rate in AlpacaEval 2. We attribute this decline to the critical role of DPO loss in improving the model's capacity to generate higher-quality answers directly. Without DPO loss, the performance of direct response generation deteriorates, which in turn negatively impacts the performance of self-refinement based on these suboptimal responses.

In Figure 5, we find that models trained on the dataset collected using the PS strategy exhibit lower overall performance compared to those trained on datasets collected with the self-refinement strategy. This suggests that, when coupled with the self-refinement strategy, ARIES is more effective in collecting high-quality datasets. Further details can be found in Appendix F, where we demonstrate that self-refinement enhances the model's responses, improving logical coherence and resulting in clearer, more concise expressions.

5.3 Generalization of ARIES in Reasoning Tasks

We further access the generalization of various methods on the GSM8K and MATH, as shown in Table 3. As iterative preference training progresses, although all methods gradually improve mathematical reasoning accuracy, ARIES significantly outperforms the baseline approaches. By employing the self-refinement strategy, ARIES achieves 73.6% accuracy on GSM8K, a 1.7% improvement over direct response generation, and 50.1% on MATH, a 1.4% improvement. These improvements stem from the self-refinement strategy, which enhances reasoning depth and maintains logical consistency during inference.

Table 3: Accuracy of different methods on GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021) tasks using **Direct** and **SR** generation strategies.

Method	**GSM8K**(%)		**MATH**(%)	
	Direct	**SR**	**Direct**	**SR**
SFT	60.9	62.3	18.2	29.5
DPO *offline*	66.3	66.6	35.1	34.6
Self-Rewarding *iter*1	65.4	66.2	35.0	34.6
Self-Rewarding *iter*2	66.3	66.5	35.3	34.7
Iterative DPO *iter*1	68.5	67.9	36.6	35.6
Iterative DPO *iter*2	68.4	67.2	36.7	36.3
ARIES-SFT	64.3	70.9	32.8	45.5
ARIES *offline*	68.3	72.2	38.3	42.4
ARIES *iter*1	68.2	71.2	39.9	45.3
ARIES *iter*2	69.1	71.2	39.5	43.9
ARIES+RM *iter*1	70.2	71.7	46.1	47.8
ARIES+RM *iter*2	**71.9**	**73.6**	**48.7**	**50.1**

However, as the model's accuracy on the GSM8K and MATH tasks reaches a certain level, the effect of self-refinement begins to plateau. We attribute this to the lack of domain-specific training, preventing the model from mastering certain problem-solving strategies. This observation has prompted us to plan further domain-specific training for reasoning tasks, to gain deeper insights into the enhancement of the model's reasoning capabilities.

6 Conclusion

In this paper, we introduce ARIES, an innovative framework for iterative preference training and self-refinement-based inference. During the training phase, ARIES enhances the model's ability to directly answer questions while simultaneously stimulating and strengthening the self-refinement capability. In the inference phase, ARIES leverages the self-refinement ability activated in the model to perform multi-step sequential inference, generating a series of self-refined outputs. These outputs are then filtered using either Rule-Based Selection or Reward Model Scoring mechanism to construct a preference dataset, which is used for the next round of preference training. The training and inference phases of ARIES are mutually reinforcing, collectively enhancing the performance of LLMs. Experimental results demonstrate the exceptional performance of ARIES, surpassing even GPT-4 on benchmarks such as AlpacaEval 2 and Arena-Hard. In the future, we will explore the application of our approach to reasoning tasks and continue to extend the existing framework.

References

Mohammad Gheshlaghi Azar, Zhaohan Daniel Guo, Bilal Piot, Remi Munos, Mark Rowland, Michal Valko, and Daniele Calandriello. A general theoretical paradigm to understand learning from human preferences. In *International Conference on Artificial Intelligence and Statistics*, pp. 4447–4455. PMLR, 2024.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022a.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022b.

Yushi Bai, Jiahao Ying, Yixin Cao, Xin Lv, Yuze He, Xiaozhi Wang, Jifan Yu, Kaisheng Zeng, Yijia Xiao, Haozhe Lyu, et al. Benchmarking foundation models with language-model-as-an-examiner. *Advances in Neural Information Processing Systems*, 36, 2024.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Lichang Chen, Shiyang Li, Jun Yan, Hai Wang, Kalpa Gunaratna, Vikas Yadav, Zheng Tang, Vijay Srinivasan, Tianyi Zhou, Heng Huang, et al. Alpagasus: Training a better alpaca with fewer data. *arXiv preprint arXiv:2307.08701*, 2023.

Eugene Choi, Arash Ahmadian, Matthieu Geist, Oilvier Pietquin, and Mohammad Gheshlaghi Azar. Self-improving robust preference optimization. *arXiv preprint arXiv:2406.01660*, 2024.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Ganqu Cui, Lifan Yuan, Ning Ding, Guanming Yao, Wei Zhu, Yuan Ni, Guotong Xie, Zhiyuan Liu, and Maosong Sun. Ultrafeedback: Boosting language models with high-quality feedback, 2023.

Hanze Dong, Wei Xiong, Bo Pang, Haoxiang Wang, Han Zhao, Yingbo Zhou, Nan Jiang, Doyen Sahoo, Caiming Xiong, and Tong Zhang. Rlhf workflow: From reward modeling to online rlhf. *arXiv preprint arXiv:2405.07863*, 2024.

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

Yann Dubois, Balázs Galambosi, Percy Liang, and Tatsunori B Hashimoto. Length-controlled alpacaeval: A simple way to debias automatic evaluators. *arXiv preprint arXiv:2404.04475*, 2024a.

Yann Dubois, Chen Xuechen Li, Rohan Taori, Tianyi Zhang, Ishaan Gulrajani, Jimmy Ba, Carlos Guestrin, Percy S Liang, and Tatsunori B Hashimoto. Alpacafarm: A simulation framework for methods that learn from human feedback. *Advances in Neural Information Processing Systems*, 36, 2024b.

Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. Kto: Model alignment as prospect theoretic optimization. *arXiv preprint arXiv:2402.01306*, 2024.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

Jiwoo Hong, Noah Lee, and James Thorne. Orpo: Monolithic preference optimization without reference model. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 11170–11189, 2024.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. *arXiv preprint arXiv:2310.01798*, 2023.

Aviral Kumar, Vincent Zhuang, Rishabh Agarwal, Yi Su, John D Co-Reyes, Avi Singh, Kate Baumli, Shariq Iqbal, Colton Bishop, Rebecca Roelofs, et al. Training language models to self-correct via reinforcement learning. *arXiv preprint arXiv:2409.12917*, 2024.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, et al. T\" ulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*, 2024a.

Nathan Lambert, Valentina Pyatkin, Jacob Morrison, LJ Miranda, Bill Yuchen Lin, Khyathi Chandu, Nouha Dziri, Sachin Kumar, Tom Zick, Yejin Choi, et al. Rewardbench: Evaluating reward models for language modeling. *arXiv preprint arXiv:2403.13787*, 2024b.

Harrison Lee, Samrat Phatale, Hassan Mansoor, Thomas Mesnard, Johan Ferret, Kellie Ren Lu, Colton Bishop, Ethan Hall, Victor Carbune, Abhinav Rastogi, et al. Rlaif vs. rlhf: Scaling reinforcement learning from human feedback with ai feedback. In *Forty-first International Conference on Machine Learning*.

Harrison Lee, Samrat Phatale, Hassan Mansoor, Thomas Mesnard, Johan Ferret, Kellie Lu, Colton Bishop, Ethan Hall, Victor Carbune, Abhinav Rastogi, et al. Rlaif: Scaling reinforcement learning from human feedback with ai feedback. *arXiv preprint arXiv:2309.00267*, 2023.

Tianle Li, Wei-Lin Chiang, Evan Frick, Lisa Dunlap, Banghua Zhu, Joseph E Gonzalez, and Ion Stoica. From live data to high-quality benchmarks: The arena-hard pipeline, 2024.

Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. Alpacaeval: An automatic evaluator of instruction-following models, 2023.

Chris Yuhao Liu, Liang Zeng, Jiacai Liu, Rui Yan, Jujie He, Chaojie Wang, Shuicheng Yan, Yang Liu, and Yahui Zhou. Skywork-reward: Bag of tricks for reward modeling in llms. *arXiv preprint arXiv:2410.18451*, 2024.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36, 2024.

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

Giovanni Monea, Antoine Bosselut, Kianté Brantley, and Yoav Artzi. Llms are in-context reinforcement learners. *arXiv preprint arXiv:2410.05362*, 2024.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35: 27730–27744, 2022.

Yuxiao Qu, Tianjun Zhang, Naman Garg, and Aviral Kumar. Recursive introspection: Teaching language model agents how to self-improve. *arXiv preprint arXiv:2407.18219*, 2024.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *arXiv preprint arXiv:2305.18290*, 2023.

Swarnadeep Saha, Omer Levy, Asli Celikyilmaz, Mohit Bansal, Jason Weston, and Xian Li. Branch-solve-merge improves large language model evaluation and generation. *arXiv preprint arXiv:2310.15123*, 2023.

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

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

Snorkel. New benchmark results demonstrate value of snorkel ai approach to llm alignment, 2024. URL https://snorkel.ai/new-benchmark-results-demonstrate-valueof-snorkel-ai-approach-to-llm-alignment.

Feifan Song, Bowen Yu, Minghao Li, Haiyang Yu, Fei Huang, Yongbin Li, and Houfeng Wang. Preference ranking optimization for human alignment. *arXiv preprint arXiv:2306.17492*, 2023.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-instruct: Aligning language models with self-generated instructions. *arXiv preprint arXiv:2212.10560*, 2022.

Tianhao Wu, Weizhe Yuan, Olga Golovneva, Jing Xu, Yuandong Tian, Jiantao Jiao, Jason Weston, and Sainbayar Sukhbaatar. Meta-rewarding language models: Self-improving alignment with llm-as-a-meta-judge. *arXiv preprint arXiv:2407.19594*, 2024.

Wei Xiong, Hanze Dong, Chenlu Ye, Ziqi Wang, Han Zhong, Heng Ji, Nan Jiang, and Tong Zhang. Iterative preference learning from human feedback: Bridging theory and practice for rlhf under kl-constraint. *arXiv preprint arXiv:2312.11456*, 2023.

Jing Xu, Andrew Lee, Sainbayar Sukhbaatar, and Jason Weston. Some things are more cringe than others: Preference optimization with the pairwise cringe loss. *arXiv preprint arXiv:2312.16682*, 2023.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers, 2024. URL https://arxiv.org/abs/2309.03409.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Sainbayar Sukhbaatar, Jing Xu, and Jason Weston. Self-rewarding language models. *arXiv preprint arXiv:2401.10020*, 2024.

Yongcheng Zeng, Guoqing Liu, Weiyu Ma, Ning Yang, Haifeng Zhang, and Jun Wang. Token-level direct preference optimization. *arXiv preprint arXiv:2404.11999*, 2024.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36:46595–46623, 2023.

A MATHEMATICAL DERIVATIONS

A.1 THE OPTIMAL SOLUTION TO THE SELF-REFINEMENT OBJECTIVE IN THE PREFERENCE TRAINING PHASE

In this Appendix, we aim to derive the loss function corresponding to the following objective:

$$\max_{\pi} \mathbb{E}_{y_2 \sim \pi(\cdot|x,y_1,z)} \Big[p(y_2 \succ y_1|x) - \beta D_{\mathrm{KL}}(\pi || \pi_{\mathrm{ref}}|x, y_1, z) \Big]. \tag{12}$$

The derivation process follows a structure similar to SRPO (Choi et al., 2024). First, we can obtain the optimal solution of the objective:

$$\max_{\pi} \mathbb{E}_{y_2 \sim \pi(\cdot|x,y_1,z)} \Big[p(y_2 \succ y_1|x) - \beta D_{\mathrm{KL}}(\pi || \pi_{\mathrm{ref}}|x, y_1, z) \Big] \tag{13}$$

$$= \max_{\pi} \mathbb{E}_{y_2 \sim \pi(\cdot|x,y_1,z)} \left[p(y_2 \succ y_1|x) - \beta \log \frac{\pi(y_2|x, y_1, z)}{\pi_{\mathrm{ref}}(y_2|x, y_1, z)} \right] \tag{14}$$

$$= \max_{\pi} \beta \mathbb{E}_{y_2 \sim \pi(\cdot|x,y_1,z)} \left[-\log \frac{\pi(y_2|x, y_1, z)}{\pi_{\mathrm{ref}}(y_2|x, y_1, z) \exp\left(\frac{p(y_2 \succ y_1|x)}{\beta}\right)} \right] \tag{15}$$

$$= \max_{\pi} -\beta \mathbb{E}_{y_2 \sim \pi(\cdot|x,y_1,z)} \left[\log \frac{\pi(y_2|x, y_1, z) Z(x, y_1, z)}{\pi_{\mathrm{ref}}(y_2|x, y_1, z) \exp\left(\frac{p(y_2 \succ y_1|x)}{\beta}\right)} \right] + \beta \log Z(x, y_1, z) \tag{16}$$

$$= \max_{\pi} -\beta D_{\mathrm{KL}} \left(\pi(y_2|x, y_1, z) \middle\| \frac{\pi_{\mathrm{ref}}(y_2|x, y_1, z) \exp\left(\frac{p(y_2 \succ y_1|x)}{\beta}\right)}{Z(x, y_1, z)} \right) + \beta \log Z(x, y_1, z) \tag{17}$$

where $Z(x, y_1, z)$ is the partition function. Considering the non-negativity of the KL divergence, the optimal solution is:

$$\pi^*(y_2|x, y_1, z) = \frac{\pi_{\mathrm{ref}}(y_2|x, y_1, z) \exp\left(\frac{p(y_2 \succ y_1|x)}{\beta}\right)}{Z(x, y_1, z)}, \tag{18}$$

For convenience in subsequent derivations, we reorganize the representation of Eq. 18 as follows:

$$p(y_2 \succ y_1|x) = \beta \log \frac{\pi^*(y_2|x, y_1, z)}{\pi_{\mathrm{ref}}(y_2|x, y_1, z)} + \beta \log Z(x, y_1, z) \tag{19}$$

Noting that $p(y_1 \succ y_1|x) = 1/2$, we derive the following expression:

$$\frac{1}{2} = \beta \log \frac{\pi^*(y_1|x, y_1, z)}{\pi_{\mathrm{ref}}(y_1|x, y_1, z)} + \beta \log Z(x, y_1, z) \tag{20}$$

Subtracting Eq. 20 from Eq. 19, we obtain the following expression:

$$p(y_2 \succ y_1|x) - \frac{1}{2} = \beta \log \frac{\pi^*(y_2|x, y_1, z))}{\pi_{\mathrm{ref}}(y_2|x, y_1, z))} - \beta \log \frac{\pi^*(y_1|x, y_1, z))}{\pi_{\mathrm{ref}}(y_1|x, y_1, z))} \tag{21}$$

According to Eq. 21, we adopt the mean squared error as the loss function and parametrize the policy model as π_θ, while iterating over all prompts x and responses y_1, y_2, which leads to:

$$\mathcal{L}(\pi_\theta; \pi_{\mathrm{ref}}) = \mathop{\mathbb{E}}_{(x,y_1,y_2)\sim\rho} \left[p(y_2 \succ y_1|x) - \frac{1}{2} - \beta \left[\log \left(\frac{\pi_\theta(y_2|x, y_1, z)}{\pi_{\mathrm{ref}}(y_2|x, y_1, z)} \right) - \log \left(\frac{\pi_\theta(y_1|x, y_1, z)}{\pi_{\mathrm{ref}}(y_1|x, y_1, z)} \right) \right] \right]^2, \tag{22}$$

where ρ represents the true distribution.

B Implementation Details

B.1 Self-Refinement Template

The self-refinement template used in this paper is as follows:

Self-Refinement Template

Below is a QUESTION from a user and an EXAMPLE RESPONSE.
Please provide a more helpful RESPONSE, improving the EXAMPLE RESPONSE by making the content even clearer, more accurate, and concise. Focus on addressing the human's QUESTION without including irrelevant sentences.
Your RESPONSE should not only be well-written, logical, and easy-to-follow, but also demonstrate expert-level insight, engaging the reader with the most relevant information.

QUESTION:
{Question}

EXAMPLE RESPONSE:
{Example_Response}

Now, refine and improve the RESPONSE further. You can consider two approaches:
1. REFINEMENT: If the EXAMPLE RESPONSE is sufficient and addresses most of the QUESTION's concerns, enhance clarity, accuracy, or conciseness as needed.
2. NEW RESPONSE: If the EXAMPLE RESPONSE lacks clarity or relevance to the QUESTION, craft a more effective RESPONSE that thoroughly resolves the QUESTION.

Do not include analysis-just give the improved RESPONSE.

RESPONSE:

B.2 Training Details

In the SFT phase, we set the learning rate to 5×10^{-6}, with a batch size of 128 and a maximum sequence length of 1024. We employed a cosine learning rate schedule with 3% warm-up steps for 1 epoch and used the AdamW optimizer.

In the preference training phase, the learning rate was reduced to 1×10^{-6}. Additionally, we set $\alpha = 0.8$ and $\beta_{\text{DPO}} = 0.1$. For training with the Reward Model Scoring dataset filtering mechanism, we set $\beta_{\text{refine}} = 0.01$, while for the Rule-Based Selection mechanism, $\beta_{\text{refine}} = 0.05$. The higher value of β_{refine} in the Rule-Based Selection process is due to the presence of noisy data in the filtered preference dataset, which requires stronger regularization.

B.3 Inference Details

During the iterative training and dataset collection process, we employed a sampling decoding strategy with a temperature of 0.7 for direct response generation and self-refinement. For AlpacaEval 2, we set the temperature to 0.9 for generation, while for MT-Bench and Arena-Hard, we followed the official decoding configuration. For GSM8K and MATH, we utilized a greedy decoding strategy.

B.4 Evaluation Benchmarks Details

AlpacaEval 2 (Li et al., 2023) consists of 805 questions from 5 datasets, MT-Bench (Zheng et al., 2023) covers 80 questions across 8 categories in a multi-turn dialogue format, and Arena-Hard (Li et al., 2024) is an enhanced version of MT-Bench with 500 well-defined technical questions. GSM8K (Cobbe et al., 2021) includes elementary and middle school-level math problems, while MATH (Hendrycks et al., 2021) contains more complex questions, spanning various mathematical branches such as algebra, counting and probability, geometry, number theory, and calculus. For each benchmark, we report scores according to their respective evaluation protocols.

C ALTERNATIVE REFINEMENT LOSS

C.1 THE DERIVATION OF THE REFINEMENT LOSS FUNCTION FROM THE BRADLEY-TERRY MODEL PERSPECTIVE

Alternatively, we can enhance the model's self-refinement capability by leveraging insights from the Bradley-Terry (BT) model theory. We define the objective function as follows:

$$\max_{\pi} \mathbb{E}_{y_2 \sim \pi(\cdot|x,y_1,z)} \Big[r(y_2|x, y_1, z) - \beta D_{\mathrm{KL}}(\pi || \pi_{\mathrm{ref}}|x, y_1, z) \Big] \tag{23}$$

The solution process is analogous to that of Appendix A.1, allowing us to obtain the optimal solution:

$$\pi^*(y_2|x, y_1, z) = \frac{\pi_{\mathrm{ref}}(y_2|x, y_1, z) \exp\left(\frac{r(y_2|x,y_1,z)}{\beta}\right)}{Z(x, y_1, z)}, \tag{24}$$

where $Z(x, y_1, z)$ is the partition function. Reorganizing the above equation, we obtain:

$$r(y_2|x, y_1, z) = \beta \log \frac{\pi(y_2|x, y_1, z)}{\pi_{\mathrm{ref}}(y_2|x, y_1, z)} + \beta \log Z(x, y_1, z) \tag{25}$$

The standard expression of the BT model is:

$$p^*_{\mathrm{BT}}(y_2 \succ y_1|x) = \sigma(r^*(y_2|x) - r^*(y_1|x)) \tag{26}$$

Here, to enhance the self-refinement capability of the language model, we make a slight modification. Given the problem input x for the BT model, we also provide an arbitrary response y_{opt} along with a refinement template z, which serves as guidance for the model to generate better responses:

$$p^*_{\mathrm{BT}}(y_2 \succ y_1|x, y_{\mathrm{opt}}, z) = \sigma(r^*(y_2|x, y_{\mathrm{opt}}, z) - r^*(y_1|x, y_{\mathrm{opt}}, z)). \tag{27}$$

Then we define the refinement preference function:

$$p^*_{\mathrm{BT\_refine}}(y_2 \succ y_1|x, z) = p^*_{\mathrm{BT}}(y_2 \succ y_1|x, y_1, z) p^*_{\mathrm{BT}}(y_2 \succ y_1|x, y_2, z) \tag{28}$$

$$= \sigma(r^*(y_2|x, y_1, z) - r^*(y_1|x, y_1, z))\sigma(r^*(y_2|x, y_2, z) - r^*(y_1|x, y_2, z)) \tag{29}$$

$$\begin{aligned} &= \sigma\left(\beta \log \frac{\pi^*(y_2|x, y_1, z)}{\pi_{\mathrm{ref}}(y_2|x, y_1, z)} - \beta \log \frac{\pi^*(y_1|x, y_1, z)}{\pi_{\mathrm{ref}}(y_1|x, y_1, z)}\right) \\ &\times \sigma\left(\beta \log \frac{\pi^*(y_2|x, y_2, z)}{\pi_{\mathrm{ref}}(y_2|x, y_2, z)} - \beta \log \frac{\pi^*(y_1|x, y_2, z)}{\pi_{\mathrm{ref}}(y_1|x, y_2, z)}\right) \end{aligned} \tag{30}$$

Assuming access to a well-curated preference dataset $\mathcal{D} = \{(x^{(i)}, y_w^{(i)}, y_l^{(i)})\}_{i=1}^N$, we aim to leverage this dataset to activate the self-refinement capability of language models, thereby gradually steering the models toward generating better responses during the inference phase. To achieve this, we parametrize the policy model π_θ and estimate its parameters through maximum likelihood estimation. By treating the problem as a binary classification task, we have the negative log-likelihood loss:

$$\begin{aligned} \mathcal{L}_{\mathrm{BT\_refine}}(\pi) = &- \mathop{\mathbb{E}}_{(x,y_w,y_l)\sim\mathcal{D}} \left[\log \sigma \left(\beta \log \frac{\pi(y_w|x, y_l, z)}{\pi_{\mathrm{ref}}(y_w|x, y_l, z)} - \beta \log \frac{\pi(y_l|x, y_l, z)}{\pi_{\mathrm{ref}}(y_l|x, y_l, z)} \right) \right] \\ &- \mathop{\mathbb{E}}_{(x,y_w,y_l)\sim\mathcal{D}} \left[\log \sigma \left(\beta \log \frac{\pi(y_l|x, y_w, z)}{\pi_{\mathrm{ref}}(y_l|x, y_w, z)} - \beta \log \frac{\pi(y_w|x, y_w, z)}{\pi_{\mathrm{ref}}(y_w|x, y_w, z)} \right) \right] \end{aligned} \tag{31}$$

Finally, we integrate the DPO loss with the self-refinement loss derived from the BT model perspective to obtain the ARIES loss function from the BT model viewpoint:

$$\mathcal{L}_{\mathrm{BT\_ARIES}}(\pi_\theta; \pi_{\mathrm{ref}}) = (1 - \alpha)\mathcal{L}_{\mathrm{DPO}}(\pi_\theta; \pi_{\mathrm{ref}}) + \alpha \mathcal{L}_{\mathrm{BT\_refine}}(\pi_\theta; \pi_{\mathrm{ref}}). \tag{32}$$

C.2 Training Details

During the SFT phase, BT_ARIES follows the same procedure as ARIES. The primary distinction between BT_ARIES and ARIES arises in the preference training phase, where we set $\beta_{\text{refine}} = 0.05$ in BT_ARIES.

C.3 Experiment Results

We compared BT_ARIES and ARIES across various benchmarks, with experimental results presented in Table 4. Both BT_ARIES and ARIES demonstrated nearly identical performance across all benchmarks, underscoring that the strength of our approach lies not in the algorithm itself, but in the foundational principles it embodies. Specifically, it is the concept of refinement that drives the effectiveness of our method and framework, enabling them to deliver impressive results.

Table 4: Comparison of experimental results between BT_ARIES and ARIES on AlpacaEval 2 (Li et al., 2023), Arena-Hard (Li et al., 2024), and MT-Bench (Zheng et al., 2023) under the **Direct** and **SR** settings. LC and WR represent length-controlled win rate and raw win rate, respectively.

Method	**Direct**					**Self-Refinement (SR)**		
	AlpacaEval 2		**Arena-Hard**	**MT-Bench**		**AlpacaEval 2**		**Arena-Hard**
	LC (%)	WR (%)	WR (%)	GPT-4 Turbo	GPT-4	LC (%)	WR (%)	WR (%)
BT_ARIES *offline*	19.8	19.3	20.0	7.0	7.5	27.8	25.7	24.8
BT_ARIES+RM *iter*1	31.9	34.5	31.1	7.1	7.5	50.6	51.8	41.0
BT_ARIES+RM *iter*2	**45.2**	**47.7**	**39.5**	7.4	7.7	**66.2**	**66.6**	49.9
ARIES *offline*	19.1	18.6	17.4	7.0	7.6	28.8	27.1	23.5
ARIES+RM *iter*1	32.7	33.5	31.9	7.3	7.7	50.2	49.9	37.5
ARIES+RM *iter*2	45.0	46.8	38.0	**7.7**	**8.1**	62.3	63.3	**50.3**

Table 5: Accuracy Comparison of BT_ARIES and ARIES on GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021) tasks using **Direct** and **SR** generation strategies.

Method	**GSM8K**(%)		**MATH**(%)	
	Direct	SR	Direct	SR
BT_ARIES *offline*	67.6	71.7	37.7	44.1
BT_ARIES+RM *iter*1	70.3	73.2	46.5	47.4
BT_ARIES+RM *iter*2	70.1	71.6	**50.2**	**52.0**
ARIES *offline*	68.3	72.2	38.3	42.4
ARIES+RM *iter*1	70.2	71.7	46.1	47.8
ARIES+RM *iter*2	**71.9**	**73.6**	48.7	50.1

D Details of Ablation Studies

The specific experimental results for Figure 4 and Figure 5 are presented in Table 6 below.

Table 6: Ablation study on AlpacaEval 2 and Arena-Hard. LC and WR represent length-controlled win rate and raw win rate, respectively.

Method	**Direct**			**Self-Refinement (SR)**		
	AlpacaEval 2		**Arena-Hard**	**AlpacaEval 2**		**Arena-Hard**
	LC (%)	WR (%)	WR (%)	LC (%)	WR (%)	WR (%)
ARIES+RM *iter*1 using PS	28.7	29.7	29.8	41.0	42.5	35.5
ARIES+RM *iter*2 using PS	40.7	42.7	37.1	59.4	60.6	49.6
ARIES *offline* w/o DPO	19.3	19.2	18.3	25.9	24.8	25.0
ARIES+RM *iter*1 w/o DPO	31.4	33.0	29.6	47.4	49.2	39.5
ARIES+RM *iter*2 w/o DPO	38.3	41.3	36.0	54.9	56.3	48.6
ARIES *offline*	19.1	18.6	17.4	28.8	27.1	23.5
ARIES+RM *iter*1	32.7	33.5	31.9	50.2	49.9	37.5
ARIES+RM *iter*2	**45.0**	**46.8**	**38.0**	**62.3**	**63.3**	**50.3**

E EVALUATION OF SELF-REFINEMENT CAPABILITIES IN OPEN-SOURCE MODELS

In this section, we evaluate the self-refinement capabilities of several prominent open-source models. To provide a comprehensive assessment, we examine the performance of the Llama-3.1-8B-Instruct, Qwen2.5-7B-Instruct, and Gemma-2-9B-Instruct models across three different refinement templates. The templates employed in this evaluation are as follows:

Refinement Template 1

Below is a QUESTION from a user and an EXAMPLE RESPONSE.
Please provide a more helpful RESPONSE, improving the EXAMPLE RESPONSE by making the content even clearer, more accurate, and concise. Focus on addressing the human's QUESTION without including irrelevant sentences.
Your RESPONSE should not only be well-written, logical, and easy-to-follow, but also demonstrate expert-level insight, engaging the reader with the most relevant information.

QUESTION:
{Question}

EXAMPLE RESPONSE:
{Example_Response}

Now, refine and improve the RESPONSE further. You can consider two approaches:
1. REFINEMENT: If the EXAMPLE RESPONSE is sufficient and addresses most of the QUESTION's concerns, enhance clarity, accuracy, or conciseness as needed.
2. NEW RESPONSE: If the EXAMPLE RESPONSE lacks clarity or relevance to the QUESTION, craft a more effective RESPONSE that thoroughly resolves the QUESTION.

Do not include analysis-just give the improved RESPONSE.

RESPONSE:

Refinement Template 2

Below is a QUESTION from a user and an EXAMPLE RESPONSE.
Please provide a more helpful RESPONSE, improving the EXAMPLE RESPONSE by making the content even clearer, more accurate, and concise. Focus on addressing the human's QUESTION without including irrelevant sentences.
Your RESPONSE should not only be well-written, logical, and easy-to-follow, but also demonstrate expert-level insight, engaging the reader with the most relevant information.

QUESTION:
{Question}

EXAMPLE RESPONSE:
{Example_Response}

Now, refine and improve the RESPONSE further. You can consider two approaches:
1. REFINEMENT: If the EXAMPLE RESPONSE is sufficient and addresses most of the QUESTION's concerns, enhance clarity, accuracy, or conciseness as needed.
2. NEW RESPONSE: If the EXAMPLE RESPONSE lacks clarity or relevance to the QUESTION, craft a more effective RESPONSE that thoroughly resolves the QUESTION.

Format your answer as follows:
ANALYSIS: <Analyze the strengths and shortcomings of the EXAMPLE RESPONSE>
RESPONSE: <Provide an improved response>

Refinement Template 3

Below is a QUESTION from a user and an EXAMPLE RESPONSE.
Please provide a better RESPONSE.

QUESTION:
{Question}

EXAMPLE RESPONSE:
{Example_Response}

RESPONSE:

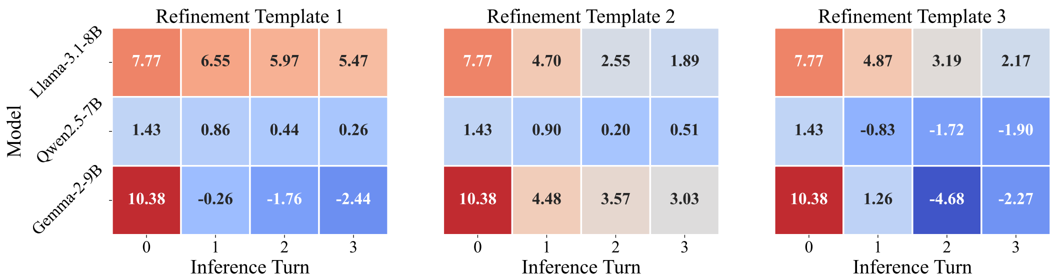

Figure 6: Self-refinement Capability Evaluation of Llama-3.1-8B-Instruct, Qwen2.5-7B-Instruct, and Gemma-2-9B-Instruct across 3 kinds of refinement template. The score is assigned by the reward model Skywork/Skywork-Reward-Llama-3.1-8B-v0.2 (Liu et al., 2024).

The experimental results are presented in Figure 6. Our experiments reveal that, despite their widespread popularity, current open-source models often struggle to effectively refine their own responses, which can even lead to a degradation in performance.

F Qualitative Examples

Prompt:
What type of soil is suitable for cactus?

ARIES+RM Direct Response:
Cacti, being succulent plants adapted to arid environments, thrive in well-draining soils that replicate their natural desert conditions. The ideal soil for cacti should meet these characteristics:
1. **Well-Draining**: Cacti cannot tolerate waterlogged conditions. The soil should have excellent drainage to prevent root rot.
2. **Low Nutrient Content**: Cacti prefer soils low in organic matter and rich in minerals, especially calcium and magnesium. Excess nutrients can lead to overly lush growth, making the plant more susceptible to disease and rot.
3. **Sand-Like Texture**: The soil should be composed of a mix of coarse sand, perlite, or pumice, which helps with drainage and aeration.
4. **Coarse-Grained**: Fine, compacted soils can retain water, which is detrimental to cacti. A gritty, well-aerated soil is essential.

Here's a typical cactus soil mix:

- **Coarse Sand**: About 40-50% (to ensure excellent drainage)
- **Pumice or Perlite**: About 30-40% (to improve aeration and drainage)
- **Compressed Coco Coir**: About 10-20% (to hold some moisture and provide a gentle nutrient base)
- **Calcitic Lime**: Optional (to raise the pH and provide calcium, essential for healthy growth)

You can purchase a pre-mixed cactus soil or create your own by combining these components. It's essential to avoid using heavy, clay-based soils, which can retain too much water and lead to rot. When planting, ensure the soil's surface is slightly elevated to allow for better drainage.

Always ensure the soil remains slightly dry between waterings to mimic the natural desert environment. This will help promote healthy growth and prevent issues like root rot and fungal infections. Happy cactus growing!

ARIES+RM Self-Refinement Response:
The **ideal soil type** for cacti is well-draining, nutrient-poor, and slightly acidic to neutral (pH 6.5-7.5). Here's a detailed breakdown:
1. Composition:

- **Primary Ingredients:**
  - **Coarse grit or sand (40-50%)**: Provides excellent drainage, preventing waterlogging which can lead to root rot.
  - **Perlite or pumice (20-30%)**: Improves aeration, retains some moisture, and enhances drainage further.
  - **Loam or vermiculite (10-20%)**: Adds a bit of fertility and retains some moisture, though still minimal.
- **Optional Additives:**
  - **Calcined clay (1-5%)**: Helps maintain soil structure and provide slight nutrient content.
  - **Activated charcoal (1-2%)**: Improves drainage, reduces odors, and filters out harmful substances.

2. Characteristics:

- **Well-draining:** Absolutely crucial. Cacti cannot tolerate waterlogged conditions.
- **Nutrient-poor:** Too much fertilizer can lead to excessive growth, making the cactus weak and susceptible to disease.
- **Acidic to slightly alkaline:** Most cacti prefer a neutral pH, but slightly acidic (around 6.5-7.0) is often ideal.
- **Loose and airy:** Allows roots to breathe and develop properly.

Instead of creating your own mix, it's highly recommended to use a commercially available cactus-specific potting mix.

These blends are formulated with the perfect proportions of the above ingredients and are readily available at garden centers and nurseries.

By using this well-draining, nutrient-poor soil, you'll create an optimal environment for your cacti to thrive, just like their natural desert home.

Table 7: A generated example on AlpacaEval 2 shows that both Direct Response Generation and Self-Refinement provide highly valuable responses for cactus planting.

Prompt:
Mick pays his teacher $800 for 40 lessons worth 2 hours each. If this will be all he is going to pay for his lessons, how much did he receive?

ARIES+RM Direct Response:
Based on the information provided, Mick paid $800 for 40 lessons, with each lesson lasting 2 hours. The calculation for the total time spent in lessons is as follows:

$$\text{Total hours} = \text{Number of lessons} \times \text{Duration per lesson} = 40 \times 2 = 80 \text{ hours}$$

Since Mick is only paying $800 for these lessons and there's no additional payment mentioned, the amount he "received" (which seems like a bit of a misinterpretation in this context, as he's paying for the lessons, not receiving a sum) is the same as what he paid, which is $800.

Therefore, Mick received (in the sense of what he paid) $800.

If the question were framed differently, ensuring clarity on what "received" exactly means (like "the total value of the lessons he paid for"), then the answer would still be $800. But considering the literal receipt of cash, it aligns with $800 he paid.

ARIES+RM Self-Refinement Response:
Mick paid a total of **$800** for his lessons. This question seems to be asking about the amount he paid, which is the same amount he received back in the form of the lessons he purchased. Therefore, Mick received **$800**.

The question mixes concepts of payment and "receiving back" (which is usually more about value received), but since all the money was paid upfront, it logically means Mick received the full value of what he paid for (40 lessons at $20 each = $800). Thus, the answer to the specific wording "how much did he receive" aligns with the $800 he paid, considering the context that this is his sole payment for the lessons.

To summarize, Mick received **$800**.

GPT4-1106-Preview Response:
To determine the amount Mick received per lesson, we need to divide the total amount paid by the number of lessons he received.

Mick paid $800 for 40 lessons.

So, $800 $\div$ 40 lessons = $20 per lesson.

Each lesson is 2 hours long, so to find out how much Mick received per hour, we divide the amount per lesson by the number of hours per lesson:

$20 per lesson $\div$ 2 hours per lesson = $10 per hour.

Thus, Mick received $10 per hour for his lessons.

Table 8: A case study on AlpacaEval 2 demonstrates that our approach, ARIES, allows the model to understand questions more deeply and comprehensively than the baseline GPT-4-1106-Preview. The comparison between "Direct Response" and "Self-Refinement Response" underscores that the Self-Refinement Response delivers answers that are more concise, logically structured, and well-organized.

Prompt:
if it takes 8 hours to fly to Singapore, stop over for an hour and a half and then fly 14 hours to London then what is the total travel time if the plane is delayed in Singapore

ARIES+RM Direct Response:
The total travel time, **including the stopover**, is **20 hours**.

Here's the breakdown:

1. **Direct flight to Singapore:** 8 hours
2. **Stopover:** 1.5 hours
3. **Direct flight to London:** 14 hours

Total: 8 hours + 1.5 hours + 14 hours = **23.5 hours**

However, you mentioned a **delay** in Singapore. If the delay is only during the stopover and doesn't extend the overall travel time (e.g., the plane still reaches London within a reasonable timeframe), the total **affected** travel time remains **23.5 hours**.

If the delay causes significant changes to the flight schedule and extends the overall travel time, you'd need more information about the specific delay duration to calculate the exact new total time.

Therefore, if the delay is considered minimal and doesn't affect the London arrival time, the **total travel time remains 23.5 hours**.

ARIES+RM Self-Refinement Response:
Here's how to calculate the total travel time, considering a delay in Singapore:
1. **Calculate the base travel time without delay:**
   - Singapore to London: 8 hours (flight) + 1.5 hours (stopover) + 14 hours (second flight) = **23.5 hours**
2. **Account for the delay in Singapore:**
   - Let the delay be represented as 'x' hours.
3. **Calculate the new total travel time:**
   - New Singapore to London travel time: 8 hours (flight) + 1.5 hours (stopover) + x hours (delay) + 14 hours (second flight) = **23.5 + x hours**

Therefore, the total travel time with the delay is **23.5 + x hours**.

To get a specific number, you need to know the duration of the delay (x). For example, if the delay is 2 hours:

- Total travel time: **23.5 + 2 = 25.5 hours**

GPT4-1106-Preview Response:
To calculate the total travel time, we need to add up all the segments of the journey along with any delays.

Here's the breakdown:

- Flight to Singapore: 8 hours
- Stopover in Singapore: 1.5 hours
- Flight to London: 14 hours

Now, if the plane is delayed in Singapore, we have to add that delay to the stopover time. However, you haven't specified the duration of the delay.

Let's assume the delay is D hours. The total travel time would then be:

Total travel time = 8 hours (first flight) + 1.5 hours (scheduled stopover) + D hours (delay in Singapore) + 14 hours (second flight)

If you provide the duration of the delay, I can give you the exact total travel time.

Table 9: A case study on AlpacaEval 2 reveals that the Self-Refinement Response exhibits a more coherent problem-solving logic, comparable to GPT-4-1106-Preview, whereas the Direct Response demonstrates more fragmented and disorganized reasoning. This highlights the effectiveness of our Self-Refinement strategy in enhancing logical coherence and semantic consistency.