Unlocking Recursive Thinking of LLMs: Alignment via Refinement

Haoke Zhang[♣,♠], **Xiaobo Liang**[♣,♠], **Cunxiang Wang**[♦,♥], **Juntao Li**[♣,♠ *], **Min Zhang**[♣,♠]
[♣]Soochow University, [♦]Zhipu AI, [♥]Tsinghua University
[♠] Key Laboratory of Data Intelligence and Advanced Computing, Soochow University

hkzhangnlp@stu.suda.edu.cn, {xbliang, ljt}@suda.edu.cn

Abstract

The OpenAI o1-series models have demonstrated that leveraging long-form Chain of Thought (CoT) can substantially enhance performance. However, the recursive thinking capabilities of Large Language Models (LLMs) remain limited, particularly in the absence of expert-curated data for distillation. In this paper, we propose **AvR**: **Alignment via Refinement**, a novel method aimed at unlocking the potential of LLMs for recursive reasoning through long-form CoT. AvR introduces a refinement process that integrates criticism and improvement actions, guided by differentiable learning techniques to optimize **refinement-aware rewards**. As a result, the synthesized multi-round data can be organized as a long refinement thought, further enabling test-time scaling. Experimental results show that AvR significantly outperforms conventional preference optimization methods. Notably, with only 3k synthetic samples, our method boosts the performance of the LLaMA-3-8B-Instruct model by over 20% in win rate on AlpacaEval 2.0. Our code is available at Github [1].

1 Introduction

Long-form CoT (Wei et al., 2022) plays a crucial role in test-time scaling (Snell et al., 2024; Muennighoff et al., 2025), as it enables recursive reasoning akin to human cognitive processes when addressing query intent (Simon and Newell, 1971; Schön, 1979; Bereiter and Scardamalia, 2013). However, most existing LLMs lack sequential revision capability, making it difficult to iteratively refine response quality through extended reasoning (Chen et al., 2024; Wang et al., 2025). Traditional alignment methods optimize LLM outputs solely based on final preference rewards, overlooking critical processes such as reflection and refinement of previously generated content. Although

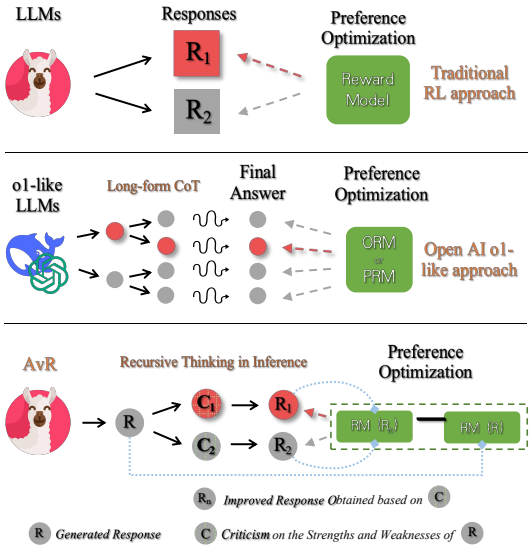

Figure 1: Reward assignment comparison between traditional RL in NLP, o1-like methods, and our AvR.

recent approaches, such as process supervision and reinforcement learning (RL) without supervised fine-tuning (Team et al., 2025; DeepSeekTeam, 2024), have shown notable gains in reasoning performance, the challenge of achieving effective test-time scaling remains an open research question.

We begin by comparing how **reward assignment** is handled across various Reinforcement Learning from Human Feedback (RLHF) algorithms, as illustrated in Figure 1. Traditional preference optimization methods fine-tune LLM behavior to directly maximize a reward function based on the Bradley–Terry preference model (Schulman et al., 2017; Rafailov et al., 2024). However, despite its efficiency, this approach fails to capture distinctions among generated responses, often leading to similar errors being repeated during parallel sampling. In contrast, o1-like models incorporate long-form CoT reasoning to align models with outcome-level or process-level rewards (Lightman et al., 2023; Wang et al., 2024b).

* Corresponding Author
[1] https://github.com/Banner-Z/AvR.git

Findings of the Association for Computational Linguistics: ACL 2025, pages 11169–11182

As a bonus, by leveraging RL algorithms and test-time computation, LLMs naturally learn to engage in recursive thinking behavior, incorporating self-verification, reflection. However, activating such capabilities incurs significant computational overhead (DeepSeekTeam, 2024), including the requirement for powerful backbone models, extensive sampling, and training costs. ***A central challenge is whether their complementary strengths can be integrated to enhance response quality with lower computational cost.***

In this work, we introduce a refinement-aware reward to unlock recursive thiking capability, which is both effective and efficient for alignment. Specifically, our method aims to maximize the preference reward of refinement, incorporating *criticism* and *improvement* actions (Kim et al., 2023). This approach draws inspiration from differential learning (Sutton, 1988), which ensures that the decision-making process can be effectively improved by optimizing the reward between different refinements.

In implementation, we design a two-stage framework to synthesize long-form recursive thinking data and optimize LLMs to align with refinement-driven behavior. ***In Stage 1***, we leverage LLMs to perform criticism and improvement actions to refine initial responses via *parallel sampling*, then rank the refined outputs to generate paired preference data. Then, we apply DPO (Rafailov et al., 2024) to train LLMs to maximize the refinement-aware reward, enabling the model to learn a policy that favors progressively improved outputs. ***In Stage 2***, we use the Stage 1 model to synthesize high-quality CoT trajectories via *sequential revision*, which serve as guidance to promote long-term refinement behavior, effectively encouraging models to optimize for long-horizon rewards. We demonstrate that our proposed AvR model, trained on only 10k examples, significantly outperforms the baseline (LLaMA-3-8B-Instruct) and even surpasses RL-based methods trained on 60k samples, as evaluated on AlpacaEval 2 and Arena-Hard v0.1. Our contributions are as follows:

- We introduce a novel **refinement-aware reward** that enables LLMs to optimize their policy and unlock recursive thinking capabilities, achieving both effectiveness and efficiency.
- Experimental results demonstrate that AvR outperforms current DPO algorithms on Alpaca Eval 2.0, achieving a **51.0% Win Rate** and a **51.4% LC Win Rate**.
- AvR presents a fundamentally different approach from online RL methods for **synthesizing high-quality long-form CoT data**, offering fresh insights into test-time scaling.

2 Related Work

Test-time compute, which leverages inference time resources to refine model outputs, has shown promise in enhancing reasoning quality. To better understand how models can acquire and benefit from recursive thinking capabilities, we provide a brief review of recent approaches.

Self-Correction While this process allows models to refine their outputs through feedback and can be further enhanced by CoT synthesis, recent work has highlighted a key limitation: LLMs often struggle to correct their own errors in the absence of an external reward function (Kamoi et al., 2024; Zhang et al., 2024a; Jiang et al., 2024). However, obtaining high-quality CoT trajectories and reliable reward functions remains challenging in open-domain tasks. To mitigate the reliance on external reward functions in multi-turn self-improvement, Qu et al. (2024) proposed a majority voting mechanism, enabling the model to select improved responses through self-comparison. Other approaches such as self-rewarding(Yuan et al., 2024) and meta-rewarding(Wu et al., 2024) utilize LLMs-as-a-judge to generate internal reward signals, facilitating iterative self-improvement. However, these methods primarily focus on enhancing LLMs' ability to evaluate or generate responses in multiple rounds, rather than enabling models to naturally engage in response refinement.

RL-based Methods Recently, o1-like LLMs have demonstrated remarkable performance on complex reasoning tasks, particularly in mathematics and code generation (OpenAI, 2024; Qwen-Team, 2024; DeepSeekTeam, 2024). Monte Carlo Tree Search (MCTS) has proven to be an effective strategy for synthesizing high-quality CoT data across math, code, and general generation tasks (Zhang et al., 2024b; Guan et al., 2025; Zhao et al., 2024). Qin et al. (2024) proposed a journey learning paradigm that encourages models to go beyond shortcut solutions, promoting complete exploration including trial and error, reflection, and backtracking. Similarly, Min et al. (2024) introduced an imitate, explore, and self-improve framework to reproduce slow-thinking reasoning systems. Wang

et al. (2024a) presented DRT-o1, which applies a long-form CoT process to machine translation and demonstrates strong performance in literature translation. Kumar et al. (2024) proposed SCoRe, an online reinforcement learning method that significantly enhances LLMs' self-correction abilities on math and code benchmarks, further showcasing the potential of recursive reasoning. Despite the effectiveness of these approaches, they often incur substantial computational costs. In contrast, our method provides an efficient alternative, achieving significant performance improvements with only a small amount of refinement data, thereby offering a more accessible path to unlocking recursive thinking capabilities in LLMs.

3 Problem Formulation

Unlike the traditional definition of token-level Markov Decision Processes (MDPs) in language generation tasks, we define both the query and each response from LLMs as independent actions. We introduce a new action, *refinement*, which can be viewed as a combination of self-judgment and self-correction performed by the LLMs. Formally, we define the multi-step MDP as a tuple $\{\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma\}$, where $\mathcal{S}$ represents the state space, with the initial state s_0 sampled from the initial prompt distribution ρ_0. The action space $\mathcal{A}$ consists of the possible sequences sampled by LLMs policy π, including both direct responses and *refinement*. The transition function $\mathcal{T}$ is typically deterministic for LLMs, modeled as $\mathcal{P}(s_{t+1}) = [s_t : a_t]$, where the next state is obtained by concatenating the previous state and the selected action. The reward function $\mathcal{R}$ provides immediate reward, while the discount factor γ balances short-term returns and long-term returns.

Most importantly, traditional token-level MDPs are optimized to maximize rewards by directly generating the final response, with explicit reward signals available only at the final state. In contrast, our proposed MDP aims to maximize the cumulative reward over multiple steps:

$$\max_{\pi} \mathbb{E}_{s_{t+1}\sim[s_t:a_t], a_t\sim\pi(\cdot|s_t)} \left[\sum_{t=0}^{|T|} \gamma^t \mathcal{R}(s_{t+1}, s_t) \right],$$

where explicit reward signals $\mathcal{R}(s_{t+1}, s_t)$ are provided at every time step, rather than being delayed until the final state.

Recursive thinking aims to encourage models to receive positive reward signals (i.e., $\mathcal{R} > 0$) at each step of an iterative reasoning process. For an initial response s_0 and a sequence of refinements $\{s_0, s_1, \ldots, s_T\}$, the refinement process can be naturally decomposed into two levels of optimization: 1) Single-step optimization, where the goal is to ensure that a refinement action yields a higher reward than the original response, i.e., $\mathcal{R}(s_1, s_0) > \mathcal{R}(s_0)$. 2) Multi-step optimization, which imposes a stronger constraint: each subsequent refinement step must produce a response with a progressively higher reward than the previous step, i.e., $\mathcal{R}(s_{t+1}, s_t) > \mathcal{R}(s_t, s_{t-1})$. An algorithm that satisfies these conditions would enable LLMs to acquire recursive thinking capabilities, empowering them to iteratively refine their outputs in a self-improving manner.

In particular, we define a **Refinement-aware Reward** that enforces s_t satisfy: Each refinement must be better than the initial response s_0; Each refinement must also be an improvement over the previous step. During optimization, we perform rejection sampling to discard any refinement trajectories that violate this condition:

$$\begin{aligned} \text{Accept}(s_t) = 1 \quad & \text{if } \mathcal{R}(s_{t+1}, s_t) > 0 \\ & \text{and } \mathcal{R}(s_{t+1}, s_t) > \mathcal{R}(s_0) \\ = 0 \quad & \text{otherwise.} \end{aligned}$$

This ensures that the model learns from only effective and cumulatively beneficial refinement steps.

4 Method

In this section, we propose a two-stage framework to unlock the recursive thinking capabilities of LLMs. As shown on the left side of Figure 2, the first stage explicitly guides the model to refine a previously generated response using carefully designed prompts. In contrast, the right side illustrates the second stage, where the model transitions to autonomous recursive thinking, performing multi-step refinement without relying on external instructions. The model constructs a recursive thinking trajectory by generating internal control statements such as "Now, let's try to criticize this answer," "Okay, let's improve the above answer based on the criticism," and "Now it's almost done." These intermediate reasoning steps are enclosed within special tokens to distinguish them from the final response. Ultimately, the model decides on its own when the refinement process is complete and proceeds to generate the final answer.

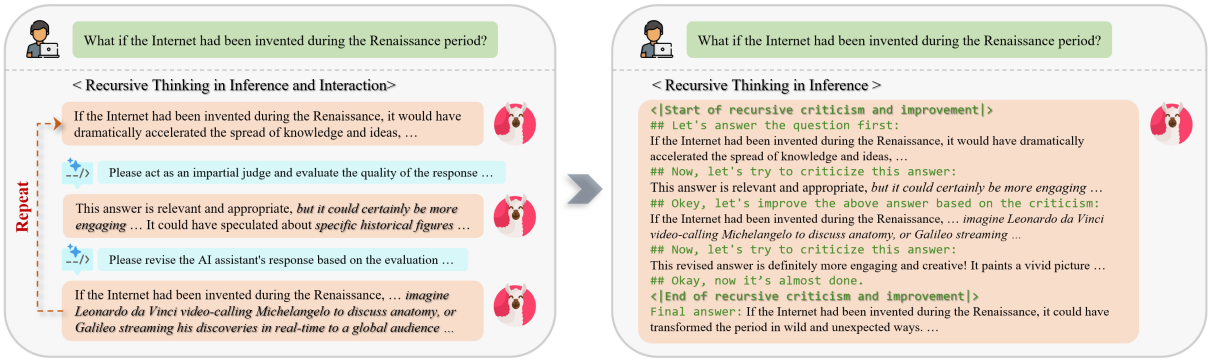

Figure 2: Illustration of two distinct reasoning paradigms. The left side depicts a **multi-step reasoning process**, where each step iteratively refines the previous response through explicit improvement. The right side illustrates a **single-step reasoning process**, wherein the model leverages recursive thinking to complete the user's query.

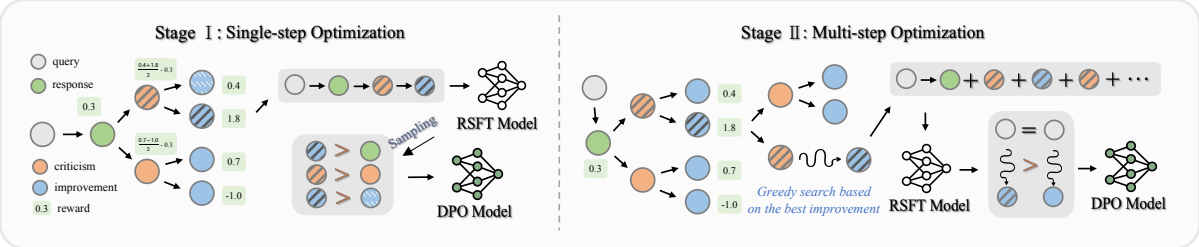

Figure 3: Illustration of our framework: on the left is the first stage, enabling recursive thinking in multi-step inference and interaction; on the right is the second stage, unlocking recursive thinking in inference autonomously.

4.1 Stage I: Single-step optimization

To enable recursive thinking in multi-step inference for LLMs, we begin by guiding the model to obtain positive reward signals through single-step optimization on each instance. In particular, we construct a refinement tree composed of *criticism* and *improvement* nodes, which allows the model to explore suitable refinement trajectories. As illustrated in the left part of Figure 3, we score both the original response and each refined response using a Bradley-Terry reward model. Our goal is to ensure that, along any trajectory within the refinement tree, the model can modify previous content and consistently receive positive reward feedback.

We begin by applying Reject-Sampling Supervised Fine-Tuning (RSFT) on the backone model, using the best improvement trajectory determined by the scores of refined responses. The RSFT training data consists of multi-turn refinement dialogues, as illustrated in the left part of Figure 2. Next, we perform joint optimization of the three core reasoning behaviors: *generation*, *criticism*, and *improvement* using Direct Preference Optimization (DPO). The pairwise training data are constructed from the refinement tree as follows: For the *generation* step, we pair the best improved response with the original response; samples are discarded if no improvement yields a higher reward; For the *criticism* step, we select a pair of criticism based on their preference scores; For the *improvement* step, we choose a pair of candidate improvements under the same criticism node. An example loss function for the generation step is given below:

$$\mathcal{L}_{\texttt{DPO}}(\pi_\theta; \pi_{\texttt{ref}}) = -\,\mathbb{E}_{(q,r^+,r^-)\sim\mathcal{D}} \left[\log \sigma\left(\beta \log \frac{\pi_\theta(r^+|q)}{\pi_{\texttt{ref}}(r^+|q)} - \beta \log \frac{\pi_\theta(r^-|q)}{\pi_{\texttt{ref}}(r^-|q)}\right)\right],$$

where q, r is the query and response, $\pi_{\texttt{ref}}$ is typically the RSFT model, β is the hyperparameter that controls the proximity of the policy π_θ to the original policy $\pi_{\texttt{ref}}$.

4.2 Stage II: Multi-step optimization

To ensure that the model can benefit from test-time compute scaling, we integrate recursive thinking into a long-form CoT reasoning framework, thereby eliminating the reliance on explicit prompts and step-specific supervision. Building upon Stage I, we adopt a greedy search strategy to synthesize recursive thinking trajectories, as illustrated on the right side of Figure 3. Specifically, given an initial response or the best refinement from the previous iteration—the model generates x criticisms, each followed by y improvements (step = 2 in our implementation). Among the resulting candidates, the

best improvement is selected using a score computed by the Bradley-Terry reward model, and this serves as the input for the next iteration. The recursive process continues until no further improvement surpasses the best response from the previous step. Finally, we train an RSFT model using these automatically selected recursive thinking trajectories, which correspond to the highest-ranked responses under the learned preference model.

In addition, we observe that the reward model tends to favor longer outputs, which often leads the RSFT model to generate unnecessarily verbose responses during inference. To mitigate this issue and enable the model to produce concise yet high-quality outputs when appropriate, we introduce a length-controlled DPO fine-tuning stage on top of the RSFT model.

Concretely, we perform multi-sample inference using the RSFT model (five samples per input in practice), and identify the highest-scoring and lowest-scoring responses according to the reward model. We then filter the samples to retain only those where the highest-scoring response is shorter than the lowest-scoring one. This subset is subsequently used to perform DPO training, thereby encouraging the model to prefer concise outputs when they are also more preferable under the reward model.

5 Experiments

5.1 Experimental Settings

Implementation Details LLaMA-Factory[2] is employed in our training. To fetch a broad range of baselines, our experiments are conducted on Meta-Llama-3-8B-Instruct[3] (AI@Meta, 2024). We use Skywork-Reward-Gemma-2-27B-v0.2[4] (Liu et al., 2024) as the Bradley-Terry Model throughout the work. We utilize llama3-ultrafeedback armorm[5] dataset, which uses 60k prompts from Ultrafeedback (Cui et al., 2023) and regenerate the chosen and rejected response pairs with Meta-Llama-3-8B-Instruct. To enhance the process, we leverage Qwen2.5-32B-Instruct-GPTQ-Int8[6] (Team, 2024) as a corrector to produce criticisms and improvements for the training of RSFT model in **AvR Stage I model**. All data synthesis following the completion of this RSFT model is performed using our own models, without the introduction of any external models. We set the temperature to 0.7 and top_p to 0.8 for all training data generation and testing. The learning rate is 5e-6 for SFT and 5e-7 for DPO. The value of β is set to 0.01. The batch size is 64, and the number of training epochs is 1.0. The cutoff length is set to 2048 for **AvR Stage I model** and 8192 for **AvR Stage II model**. We utilize the DeepSpeed (Rasley et al., 2020) library, Zero Redundancy Optimizer (ZeRO) (Rajbhandari et al., 2020) Stage 3, and FlashAttention (Dao, 2023), along with a mixed precision computation approach using bfloat16 (BF16) and tfloat32 (TF32), across 8 NVIDIA A100 GPUs.

Evaluation We evaluate our models on two open-ended conversation benchmarks: Alpaca Eval 2 (Li et al., 2023), and Arena-Hard v0.1 (Li et al., 2024). Alpaca Eval 2 is an LLM-based automatic evaluation with 805 questions from 5 datasets. The model's responses are compared with those of GPT-4-Turbo. The win rate represents the likelihood that the auto-evaluator favors the evaluated model's responses. To mitigate the length bias in the auto-evaluator, the Length-controlled Win Rate is utilized. Arena-Hard v0.1 contains 500 challenging user queries sourced from Chatbot Arena, comparing the models' responses against a baseline model (GPT-4-0314). Following the standard experimental setup, GPT-4 Turbo (corresponding to GPT-4-Preview-1106) is used as the auto-evaluator in three benchmarks.

Baselines In our experiments, we primarily introduce three types of baselines. The first consists of well-pretrained and aligned dialogue models, such as Qwen2.5-32B-Instruct-GPTQ-Int8, Llama-3.1-405B-Instruct[7], and GPT-4-0613. The second includes models trained with hybrid reinforcement learning methods, including DPO (Rafailov et al., 2023), KTO (Ethayarajh et al., 2024), ORPO (Hong et al., 2024), R-DPO (Park et al., 2024), and SimPO (Meng et al., 2025). The third category encompasses self-improvement approaches, including Self-Rewarding LLM (Yuan et al., 2024) and Meta-Rewarding LLM (Wu et al., 2024).

[2] https://github.com/hiyouga/LLaMA-Factory.git

[3] https://huggingface.co/meta-llama/Meta-Llama-3-8B-Instruct

[4] https://huggingface.co/Skywork/Skywork-Reward-Gemma-2-27B-v0.2

[5] https://huggingface.co/datasets/princeton-nlp/llama3-ultrafeedback-armorm

[6] https://huggingface.co/Qwen/Qwen2.5-32B-Instruct-GPTQ-Int8

[7] https://huggingface.co/meta-llama/Llama-3.1-405B-Instruct

Method	Init Model	Data scale	Win Rate	Len.-control. Win Rate	Length
SEED	Llama-3-8B-Ins	-	25.0%	25.0%	1956
+*refine*	Llama-3-8B-Ins	-	22.4%$_{\downarrow 2.6\%}$	21.4%$_{\downarrow 3.6\%}$	1952
+*refine*	Qwen2.5-32B-Ins	-	33.3%$_{\uparrow 8.3\%}$	34.7%$_{\uparrow 9.7\%}$	1943
gpt-4-0613	gpt-4-0613	-	15.8%	30.2%	-
Llama-3.1-405B-Ins	Llama-3.1-405B-Ins	-	39.1%	39.3%	-
RL Methods					
DPO	Llama-3-8B-Ins	60k	37.9%$_{\uparrow 12.9\%}$	40.3%$_{\uparrow 15.3\%}$	-
KTO	Llama-3-8B-Ins	60k	31.8%$_{\uparrow 6.8\%}$	33.1%$_{\uparrow 8.1\%}$	-
ORPO	Llama-3-8B-Ins	60k	37.8%$_{\uparrow 12.8\%}$	41.1%$_{\uparrow 16.1\%}$	-
SimPO	Llama-3-8B-Ins	60k	40.5%$_{\uparrow 15.5\%}$	<u>44.7%</u>$_{\uparrow 19.7\%}$	1825
Meta-Rewarding LLM					
Iteration 1	Llama-3-8B-Ins	5k	27.6%$_{\uparrow 2.6\%}$	27.9%$_{\uparrow 2.9\%}$	1949
Iteration 2	Llama-3-8B-Ins	5k (10k)	33.3%$_{\uparrow 8.3\%}$	32.7%$_{\uparrow 7.7\%}$	2001
Iteration 3	Llama-3-8B-Ins	5k (15k)	37.2%$_{\uparrow 12.2\%}$	35.5%$_{\uparrow 10.5\%}$	2064
Iteration 4	Llama-3-8B-Ins	5k (20k)	39.5%$_{\uparrow 14.5\%}$	39.4%$_{\uparrow 14.4\%}$	2003
AvR Stage I					
RSFT	Llama-3-8B-Ins	10k	21.1%$_{\downarrow 3.9\%}$	22.1%$_{\downarrow 2.9\%}$	1925
+*refine round 1*	Llama-3-8B-Ins	-	25.7%$_{\uparrow 0.7\%}$	22.8%$_{\downarrow 2.2\%}$	2234
+*refine round 2*	Llama-3-8B-Ins	-	27.2%$_{\uparrow 2.2\%}$	22.6%$_{\downarrow 2.4\%}$	2457
DPO	Llama-3-8B-Ins	10k (20k)	37.0%$_{\uparrow 12.2\%}$	36.2%$_{\uparrow 11.2\%}$	2179
+*refine round 1*	Llama-3-8B-Ins	-	49.2%$_{\uparrow 24.2\%}$	39.1%$_{\uparrow 14.1\%}$	2562
+*refine round 2*	Llama-3-8B-Ins	-	<u>50.8%</u>$_{\uparrow 25.8\%}$	35.5%$_{\uparrow 10.5\%}$	2963
AvR Stage II					
RSFT	Llama-3-8B-Ins	10k	**51.0%**$_{\uparrow 26.0\%}$	42.5%$_{\uparrow 17.5\%}$	2687
+*length control*	Llama-3-8B-Ins	4k (14k)	49.0%$_{\uparrow 24.0\%}$	**51.4%**$_{\uparrow 26.4\%}$	1989

Table 1: The experimental results on Alpaca Eval 2, with **bold numbers** indicating the best performance and <u>underlined numbers</u> representing the second-best performance.

5.2 Main Results

Table 1 presents the main results of our experiments on Alpaca Eval 2. We report both the win rate and the length-controlled win rate to assess the performance of the methods. For each score, we indicate the difference between the results before and after applying the method. Additionally, we provide the average string length of the model outputs and the scale of training prompts for each model[8] (The numbers in brackets represent the cumulative amount of training prompts used if the model was trained iteratively.). We analyze the performance from the following perspectives:

AvR Stage II model shows significant improvements by constructing high-quality training data through inference scaling. While the Llama-3-8B-Instruct model struggles with self-correction on Alpaca Eval 2, the win rate on the first generation improves by 12% after the RSFT and DPO training on only 20k data and the self-correction ability improves even further, resulting in an additional 12.2% improvement in the quality of the first responses.

LLMs are highly effective at recursive thinking in inference. With a well-trained AvR Stage I model, we find that just 10k prompts can lead to a 26% improvement in the win rate and a 17% improvement in the length-controlled win rate by constructing long-form CoT data, which encourages recursive criticism and improvement during the inference process. In comparison to meta-rewarding approaches or hybrid reinforcement learning methods, which search for a better response to guide LLMs in generating improved outputs, our method—focused on teaching LLMs the iterative process—yields a greater overall im-

[8]To mitigate the substantial decline in generation capabilities caused by training with a fixed-format prompting dataset during the AvR Stage I, we incorporate all 60k responses generated by Meta-Llama-3-8B-Instruct into the RSFT training corpus, thereby preserving its generative abilities.

Method	Score	95% CI	Length
gpt-3.5-turbo-0125	23.3%	(-2.2, 1.9)	-
gpt-4-0613	37.9%	(-2.8, 2.4)	-
Llama-3-8B-Instruct (Seed)	20.6%	(-2.0, 1.8)	2485
RL Method			
DPO	32.6%	-	-
ORPO	25.8%	-	-
R-DPO	33.1%	-	-
SimPO	33.8%	-	-
Self-Rewarding LLM			
Iteration 1	23.2%	(-1.7, 1.9)	2438
Iteration 2	26.3%	(-2.1, 2.3)	2427
Iteration 3	28.2%	(-2.0, 1.9)	2413
Iteration 4	27.3%	(-2.0, 2.2)	2448
Meta-Rewarding LLM			
Iteration 1	25.1%	(-1.9, 1.8)	2395
Iteration 2	27.4%	(-2.0, 2.0)	2416
Iteration 3	27.6%	(-2.3, 2.6)	2501
Iteration 4	29.1%	(-2.3, 2.1)	2422
AvR Stage II	**34.5%**	(-2.5, 2.3)	3144

Table 2: The experimental results on Arena Hard v0.1.

provement in final output with fewer training data and lower training costs.

Constructing preference pairs by maximizing the difference before and after refinement significantly enhances the self-iterative ability of LLMs. When comparing the results of RSFT with those of DPO in our AvR Stage I model, the improvement after one round of refinement is only 4.6% for RSFT, whereas Qwen2.5-32B-Instruct achieves an 8.3% improvement, as it generates the training data for RSFT. The DPO model reverses this trend, even though it only uses the data generated by the RSFT model. We believe the key advantage of our DPO approach lies in allowing the LLMs to learn how to maximize the difference between the outputs before and after refinement.

The AvR Stage II model requires only a slight DPO to learn length control, relying on self-generated preference data. Our AvR Stage II model tends to increase the output length in each refinement round, which results in a lower score on the length-controlled win rate. However, our experiments demonstrate that the AvR Stage II model can efficiently learn length control by training with just 4k preference pairs. This method achieves an 8.9% improvement in the length-controlled win rate and reduces the length of the final outputs, bringing them closer to the seed model, with only a 2.0% decline in the win rate.

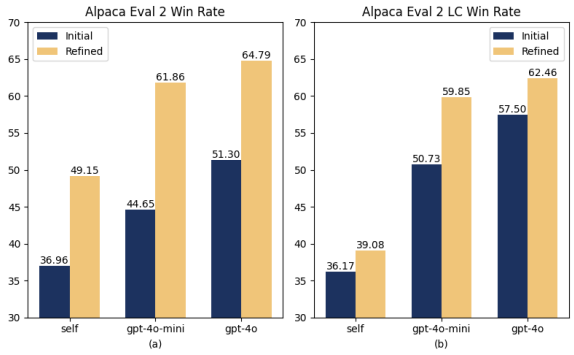

Figure 4: The results demonstrate our model's ability to correct the outputs of more powerful LLMs.

5.3 Results on Arena Hard v0.1

To further validate the effectiveness of our proposed method, we conduct experiments on the Arena Hard v0.1 benchmark using our AvR Stage II model. Among the methods trained based on Llama-3-8B-Instruct, AvR Stage II model achieves the best performance, surpassing that of GPT-3.5-turbo-0125. The 95% confidence interval (95% CI) for our model's performance ranges from (-2.5, 2.3). In comparison to approaches such as Self-Rewarding LLM and Meta-Rewarding LLM, which require multiple iterations of training, and methods like DPO and SimPO, which rely on extensive preference data for reinforcement learning, our framework demonstrates superior efficiency. Notably, our approach achieves optimal performance with only 10k data points during SFT highlighting its effectiveness and simplicity.

5.4 Analysis of Refinement Capability

To further investigate the refinement capability of our AvR Stage I model, we employ it to refine the outputs of stronger LLMs and evaluate their performance on Alpaca Eval 2. Specifically, we use the original outputs of GPT-4o and GPT-4o-mini[9], as provided by the benchmark, and apply our model to refine these outputs. As shown in Figure 4, our model significantly improves the performance of both models, even though the win rate and length-controlled win rate of our self-refinement results are notably lower than the original outputs of GPT-4o. When combined with the results in Table 1, it is evident that surpassing the generation quality of powerful models like GPT-4o and GPT-4o-mini is challenging through reinforcement learning or iterative training focused solely on model generation

[9]The GPT-4o-0513 and GPT-4o-mini-2024-07-18 models are utilized in this work.

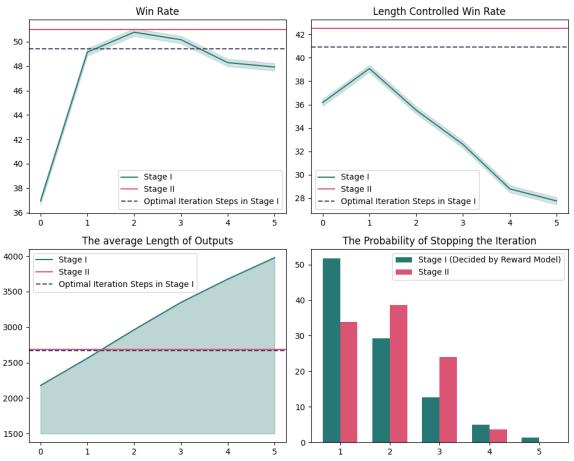

Figure 5: Analysis of iteration capability by comparing the models of the two stages.

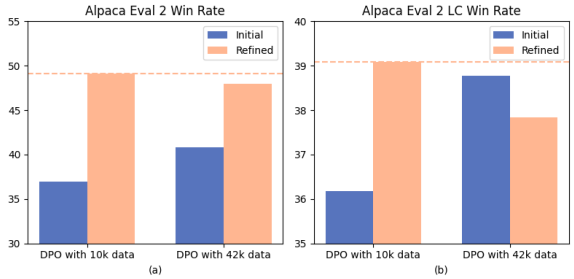

Figure 6: Experimental results on the scaling of DPO training data in the first stage.

Data scale	Win Rate	LC Win Rate	Length
Init Model	24.99%	24.96%	1956
3k	45.22%	38.69%	2683
6k	47.71%	39.06%	2719
10k	51.02%	42.46%	2687
20k	**51.07%**	**42.69%**	2559

Table 3: Experimental results on the scaling of SFT training data in the second stage.

capabilities. However, by enhancing the refinement ability of the model through inference scaling and difference maximization training, we can obtain improved responses from these powerful models. This also suggests promising directions for the future development of model capabilities.

5.5 Analysis of Iteration Capability

In Figure 5, we illustrate the performance variations across multiple iterations, comparing the AvR Stage I model with the AvR Stage II model. In our experiments, the AvR Stage I model is used to generate criticisms and improved responses iteratively, while the Bradley-Terry Model (Skywork-Reward-Gemma-2-27B-v0.2) acts as a verifier, determining when to stop the iteration. Specifically, the AvR Stage I model halts iteration when the Bradley-Terry Model detects no improvement in performance compared to the current iteration, and the previous response is then selected as the final output. The four smaller subfigures in the main figure represent the following from top to bottom and left to right: the change in win rate score, the change in length-controlled win rate score, the change in length, and the distribution of the best iteration round. One key observation is that the AvR Stage I model fails to prevent the continuous increase in length during the iteration process, although the overall quality does not improve significantly after the first two iterations. This trend is also reflected in the length-controlled win rate and the distribution of optimal iteration rounds. The dotted line in the figure indicates the result of selecting the best iteration round at the instance level. It can be seen that the AvR Stage II model (represented by the red solid line) consistently outperforms this configuration on AvR Stage I model, further demonstrating the effectiveness and importance of unlocking the recursive thinking of LLMs. Additionally, we observe changes in the distribution of iteration rounds. The best iteration round typically falls between the first and third rounds, with the AvR Stage II model showing a stronger preference for the second round.

5.6 Scaling on Training Data

In our experiment, we also analyze the scale of the training data, focusing primarily on the DPO training during the Stage I and the SFT training during the Stage II. Figure 6 presents the results of the scaling experiment during the DPO training. It is evident that the model's generation ability significantly improved when trained with 42k prompts compared to the 10k prompts setting. This observation is also reflected in Table 1—training with 60k prompts resulted in better generation performance than our DPO model. However, the model trained with 42k prompts demonstrates significantly lower correction performance than the model trained with 10k prompts, even leading to a decrease in the length-controlled win rate during the refinement phase. Overall, after one round of refinement, the 10k model outperforms the 42k model. Since our experiment places greater emphasis on the desirable property of self-refinement (as discussed in Section 5.4), we use the model trained with 10k

prompts for all subsequent experiments. Table 3 presents the experimental results on data scaling during the SFT training in the Stage II. We found that training with just 3k long CoT data resulted in a performance improvement of over 20%, which was comparable to the performance of the Meta-Rewarding LLM trained on 20k data and the reinforcement learning method trained on 60k data. Up to 10k training data, data scaling consistently led to significant improvements in model performance. However, the improvement from 10k to 20k training data is marginal. We attribute this phenomenon to a combination of model capacity limitations, data quality constraints (partly due to the misalignment of the reward model), and the training algorithm. In all other experiments, we used the 10k training version.

6 Conclusion

In this paper, we propose a novel framework to unlock the recursive thinking capabilities of LLMs. By leveraging inference-time scaling and maximizing a refinement-aware reward, our method synthesizes high-quality data that supports long CoT reasoning and enables self-improvement within the model itself. Crucially, our two-stage training and inference framework significantly reduces the GPU memory footprint and training cost compared to RL-based methods. Experimental results demonstrate that our approach substantially enhances both the generative and corrective abilities of the base model. Remarkably, with only a small amount of training data, our method empowers an 8B model to dynamically refine its outputs during inference. Furthermore, we observe that our model can effectively refine responses generated by more powerful models, offering new perspectives for improving LLM performance through recursive self-improvement.

Limitations

Since the evaluation of dialogue tasks often has problems such as inconsistency and uncertainty, the ability and preference of the reward model often affect the quality of synthetic data and thus the effect of the model. Better reward modeling, such as using LLM-as-a-judge for fine-grained evaluation to provide better criticism for refinement (Wang et al., 2023; Liang et al., 2024; Ankner et al., 2024), or other better forms of feedback, will bring more expansion and improvement to our work. In addition, this work does not explore more reinforcement learning methods. We have preliminarily discovered in the experiment that reinforcement learning has significantly enhanced this inference method. We believe this is a direction worthy of further expansion in the future.

Acknowledgments

We want to thank all the anonymous reviewers for their valuable comments. This work was supported by the National Science Foundation of China (NSFC No. 62206194), the Natural Science Foundation of Jiangsu Province, China (Grant No. BK20220488), the Young Elite Scientists Sponsorship Program by CAST (2023QNRC001). We also acknowledge MetaStone Tech. Co. for providing us with the software, optimisation on high performance computing and computational resources required by this work.

References

AI@Meta. 2024. Llama 3 model card.

Zachary Ankner, Mansheej Paul, Brandon Cui, Jonathan D. Chang, and Prithviraj Ammanabrolu. 2024. Critique-out-loud reward models. *Preprint*, arXiv:2408.11791.

Carl Bereiter and Marlene Scardamalia. 2013. *The psychology of written composition*. Routledge.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, et al. 2024. Do not think that much for 2+ 3=? on the overthinking of o1-like llms. *arXiv preprint arXiv:2412.21187*.

Ganqu Cui, Lifan Yuan, Ning Ding, Guanming Yao, Wei Zhu, Yuan Ni, Guotong Xie, Zhiyuan Liu, and Maosong Sun. 2023. Ultrafeedback: Boosting language models with high-quality feedback. *Preprint*, arXiv:2310.01377.

Tri Dao. 2023. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*.

DeepSeekTeam. 2024. Deepseek-r1-lite-preview is now live: unleashing supercharged reasoning power.

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

Xinyu Guan, Li Lyna Zhang, Yifei Liu, Ning Shang, Youran Sun, Yi Zhu, Fan Yang, and Mao Yang. 2025. rstar-math: Small llms can master math reasoning with self-evolved deep thinking. *arXiv preprint arXiv:2501.04519*.

Jiwoo Hong, Noah Lee, and James Thorne. 2024. Reference-free monolithic preference optimization with odds ratio. *arXiv e-prints*, pages arXiv–2403.

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*.

Ryo Kamoi, Yusen Zhang, Nan Zhang, Jiawei Han, and Rui Zhang. 2024. When can llms actually correct their own mistakes? a critical survey of self-correction of llms. *Transactions of the Association for Computational Linguistics*, 12:1417–1440.

Geunwoo Kim, Pierre Baldi, and Stephen McAleer. 2023. Language models can solve computer tasks. *Advances in Neural Information Processing Systems*, 36:39648–39677.

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

Tianle Li, Wei-Lin Chiang, Evan Frick, Lisa Dunlap, Tianhao Wu, Banghua Zhu, Joseph E Gonzalez, and Ion Stoica. 2024. From crowdsourced data to high-quality benchmarks: Arena-hard and benchbuilder pipeline. *arXiv preprint arXiv:2406.11939*.

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

Xiaobo Liang, Haoke Zhang, Juntao Li, Jun Xu, Min Zhang, et al. 2024. Fennec: Fine-grained language model evaluation and correction extended through branching and bridging. *arXiv preprint arXiv:2405.12163*.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. *arXiv preprint arXiv:2305.20050*.

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

Yu Meng, Mengzhou Xia, and Danqi Chen. 2025. Simpo: Simple preference optimization with a reference-free reward. *Advances in Neural Information Processing Systems*, 37:124198–124235.

Yingqian Min, Zhipeng Chen, Jinhao Jiang, Jie Chen, Jia Deng, Yiwen Hu, Yiru Tang, Jiapeng Wang, Xiaoxue Cheng, Huatong Song, et al. 2024. Imitate, explore, and self-improve: A reproduction report on slow-thinking reasoning systems. *arXiv preprint arXiv:2412.09413*.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. 2025. s1: Simple test-time scaling. *arXiv preprint arXiv:2501.19393*.

OpenAI. 2024. Learning to reason with llms. https://openai.com/index/learning-to-reason-with-llms/.

Ryan Park, Rafael Rafailov, Stefano Ermon, and Chelsea Finn. 2024. Disentangling length from quality in direct preference optimization. *arXiv preprint arXiv:2403.19159*.

Yiwei Qin, Xuefeng Li, Haoyang Zou, Yixiu Liu, Shijie Xia, Zhen Huang, Yixin Ye, Weizhe Yuan, Hector Liu, Yuanzhi Li, et al. 2024. O1 replication journey: A strategic progress report–part 1. *arXiv preprint arXiv:2410.18982*.

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

QwenTeam. 2024. Qwq: Reflect deeply on the boundaries of the unknown, november 2024. *URL https://qwenlm. github. io/blog/qwq-32b-preview*.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36:53728–53741.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. 2020. Zero: Memory optimizations toward training trillion parameter models. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–16. IEEE.

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 3505–3506.

Donald A Schön. 1979. The reflective practitioner. *New York*.

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

Herbert A Simon and Allen Newell. 1971. Human problem solving: The state of the theory in 1970. *American psychologist*, 26(2):145.

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

Richard S Sutton. 1988. Learning to predict by the methods of temporal differences. *Machine learning*, 3:9–44.

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, et al. 2025. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*.

Qwen Team. 2024. Qwen2.5: A party of foundation models.

Jiaan Wang, Fandong Meng, Yunlong Liang, and Jie Zhou. 2024a. Drt-o1: Optimized deep reasoning translation via long chain-of-thought. *arXiv preprint arXiv:2412.17498*.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. 2024b. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9426–9439.

Yidong Wang, Zhuohao Yu, Zhengran Zeng, Linyi Yang, Cunxiang Wang, Hao Chen, Chaoya Jiang, Rui Xie, Jindong Wang, Xing Xie, et al. 2023. Pandalm: An automatic evaluation benchmark for llm instruction tuning optimization. *arXiv preprint arXiv:2306.05087*.

Yue Wang, Qiuzhi Liu, Jiahao Xu, Tian Liang, Xingyu Chen, Zhiwei He, Linfeng Song, Dian Yu, Juntao Li, Zhuosheng Zhang, et al. 2025. Thoughts are all over the place: On the underthinking of o1-like llms. *arXiv preprint arXiv:2501.18585*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

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

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

Qingjie Zhang, Han Qiu, Di Wang, Haoting Qian, Yiming Li, Tianwei Zhang, and Minlie Huang. 2024a. Understanding the dark side of llms' intrinsic self-correction. *arXiv preprint arXiv:2412.14959*.

Yuxiang Zhang, Shangxi Wu, Yuqi Yang, Jiangming Shu, Jinlin Xiao, Chao Kong, and Jitao Sang. 2024b. o1-coder: an o1 replication for coding. *arXiv preprint arXiv:2412.00154*.

Yu Zhao, Huifeng Yin, Bo Zeng, Hao Wang, Tianqi Shi, Chenyang Lyu, Longyue Wang, Weihua Luo, and Kaifu Zhang. 2024. Marco-o1: Towards open reasoning models for open-ended solutions. *arXiv preprint arXiv:2411.14405*.

A Appendix

A.1 Comparison with LongCoT Model

To compare the o1-like LongCoT model with our method, we evaluate the DeepSeek-R1-Distill-Llama-8B[10] model on the Alpaca Eval 2.0 benchmark. As shown in Table 4, we observe that DeepSeek-R1-Distill-Llama-8B, trained on large-scale LongCoT data, performs poorly on this benchmark. Moreover, although the distilled DeepSeek R1 model can generate very long thinking, the thinking length of the DeepSeek-R1-Distill-Llama-8B model on Alpaca Eval 2.0 is only 2.8 times that of the final response, while our models can reach 3.5 times or even more than 4 times.

A.2 Experiment on Math Tasks

We conduct experiments on the Qwen2.5-7B-Instruct[11] model to evaluate performance on several math benchmarks. To refine the model's responses, we leverage DeepSeek-R1, effectively simulating a well-trained Stage I model. We sample 1,000 questions from OpenThoughts-114k and employ the Qwen2.5-Math-RM-72B[12] as our reward model. As shown in Table 5, our method achieves consistent improvements across various math benchmarks, demonstrating the effectiveness of the proposed approach for math tasks.

[10] https://huggingface.co/deepseek-ai/DeepSeek-R1-Distill-Llama-8B

[11] https://huggingface.co/Qwen/Qwen2.5-7B-Instruct

[12] https://huggingface.co/Qwen/Qwen2.5-Math-RM-72B

Model	Win Rate	Len.-control. Win Rate	Length of Final Response	Length of Thinking
Meta-Llama-3-8B-Instruct	25.0%	25.0%	1956	-
DeepSeek-R1-Distill-Llama-8B	22.1%	27.9%	1615	4472
AvR Stage II				
RSFT	**51.0%**	42.5%	2687	11781
+length control	49.0%	**51.4%**	1989	6947

Table 4: The experimental results of our methods and the common approach of using long CoT for reasoning enhancement on Alpaca Eval 2, with **bold numbers** indicating the best performance.

Method	Data Scale	AIME24	MATH 500	Olympiad Bench	Minerva Math	GSM8K
Qwen2.5-7B-Instruct	-	10.0	74.2	36.7	33.5	91.6
SFT with Our Stage II Data	1k	**16.7**	**74.8**	**38.5**	**34.9**	**92.4**

Table 5: Results on math benchmarks for the Qwen2.5-7B-Instruct model with and without our Stage II data.

A.3 Prompts and Cases

In this section, we present the prompts and formats used for model training and inference, and provide several cases of the inference process.

Prompts in AvR

User: {User Query}
Assistant: {Response}
User: Please act as an impartial judge and evaluate the quality of the response provided by an AI assistant to the user question displayed above. Your evaluation should consider factors such as the helpfulness, relevance, accuracy, depth, creativity, and level of detail of the response. Your evaluation should focus on the assistant's answer to the last user question. Begin your evaluation by providing a short explanation. Be as objective as possible. After providing your explanation, you must rate the response at the end of your answer on a scale of 1 to 10 by strictly following this format: "[[rating]]", for example: "Rating: [[5]]".
Assistant: {Judgement}
User: Please revise the AI assistant's response based on the evaluation provided above, addressing any shortcomings mentioned in the review. Your revision should focus solely on improving the assistant's answer to the last user question. Provide the revised response directly, without any additional commentary.
Assistant: {Refinement}

User: Who was the first lead vocal for the rock band Starship?

Assistant: The first lead vocalist for the rock band Starship was Marty Balin, who was a founding member of the band. He was the lead singer on many of their early hits, including "Don't Count Me Out," "We Built This City," and "Miracles."

User: Please act as an impartial judge and evaluate the quality of the response ...

Assistant: The AI assistant provided a clear and accurate response to the user's question about the first lead vocalist for Starship. The information given is relevant and directly addresses the user's query. The response is detailed enough to mention specific songs and the founding member's role, which adds depth to the answer without being overly verbose. The information provided is helpful for someone seeking to understand the early lead vocals of the band. The response is also creative in the sense that it includes specific examples of songs and emphasizes the founding member's role, which enhances the user's understanding of the context.
Rating: [[8]]
The rating reflects the high quality of the response, with a minor deduction due to the absence of additional context or historical context that might have further enriched the answer, but this was not necessary given the straightforward nature of the question. The information provided is comprehensive and directly answers the user's query effectively.

User: Please revise the AI assistant's response based on the evaluation provided above, ...

Assistant: The first lead vocalist for the rock band Starship was Marty Balin, a founding member who played a pivotal role in shaping the band's sound. During the early years, he was the primary lead singer on hits like "Don't Count Me Out" (1969) and "We Built This City" (1979). These songs showcased his distinctive vocals, setting the foundation for the band's later success. Balin's contributions as a lead vocalist are significant in understanding Starship's early years and development.

Table 6: Case on AvR Stage I Model.

User: Please give me a list of planets in our solar system. I am going to choose which one I want to know more.
Assistant: "**<|Start of recursive criticism and improvement|>**
Let's answer the question first:

Here is the list of planets in our solar system: 1. Mercury 2. Venus 3. Earth 4. Mars 5. Jupiter 6. Saturn 7. Uranus 8. Neptune
Which one would you like to know more about?

Now, let's try to criticize this answer:

Rating: [[8]] The AI assistant provided a concise and relevant list of planets in our solar system, directly addressing the user's request. The response is helpful, accurate, and detailed enough to give the user a clear choice. ...
The only aspect that prevents a perfect score is the absence of additional context or explanation about the solar system structure or the format of the response, which might have added depth without compromising clarity. Given the straightforward nature of the request, this minor omission does not detract significantly from the overall quality. Therefore, the rating reflects a high level of helpfulness and relevance, with a slight deduction for the potential depth that could have been included without compromising clarity.
Rating: [[8]] (The response is highly relevant, accurate, and helpful, with a minor deduction for the absence of additional context without compromising clarity.)

Okey, let's improve the above answer based on the criticism:

Here is the list of planets in our solar system, organized for easy selection: **1.** **Mercury**: Closest to the Sun, with extreme temperatures and a highly elliptical orbit. **2.** **Venus**: Hottest planet, shrouded in thick clouds, with surface temperatures reaching 462°C (863°F).
...
Which planet would you like to explore further? (Choose one for a detailed dive!)

Now, let's try to criticize this answer:

Rating: [[8]] The AI assistant provided a comprehensive and structured list of the planets in our solar system, catering to the user's request for a detailed exploration of a chosen planet. The response: 1. **Helpfulness**: The list is detailed and well-organized, offering a clear overview of each planet, making it easy for the user to select one for further exploration.
...
Overall, the response is highly effective in meeting the user's needs and setting the stage for a detailed exploration of their chosen planet.

Okay, now it's almost done.
<|End of recursive criticism and improvement|>

Final answer: Here is the list of planets in our solar system, organized for easy selection: **1.** **Mercury**: Closest to the Sun, with extreme temperatures and a highly elliptical orbit. **2.** **Venus**: Hottest planet, shrouded in thick clouds, with surface temperatures reaching 462°C (863°F). **3.** **Earth**: Our home planet, supporting life, with diverse ecosystems and a moderate climate.
...
Which planet would you like to explore further? (Choose one for a detailed dive!)"

Table 7: Case on AvR Stage II model.