
Self-rewarding correction for mathematical reasoning

Wei Xiong^{*1} Hanning Zhang^{*1} Chenlu Ye^{*1} Lichang Chen² Nan Jiang¹ Tong Zhang¹

Abstract

We study self-rewarding reasoning large language models (LLMs), which can simultaneously generate step-by-step reasoning and evaluate the correctness of their outputs during the inference time *without external feedback*. This integrated approach allows a single model to independently guide its reasoning process, offering computational advantages for model deployment.

We particularly focus on the representative task of self-correction, where models autonomously detect errors in their responses, revise outputs, and decide when to terminate iterative refinement loops. To enable this, we propose a two-staged algorithmic framework for constructing self-rewarding reasoning models using only self-generated data. In the first stage, we employ sequential rejection sampling to synthesize long chain-of-thought trajectories that incorporate both self-rewarding and self-correction mechanisms. Fine-tuning models on these curated data allows them to learn the patterns of self-rewarding and self-correction. In the second stage, we further enhance the models’ ability to assess response accuracy and refine outputs through *reinforcement learning* with rule-based signals. Experiments with Llama-3 and Qwen-2.5 demonstrate that our approach surpasses intrinsic self-correction capabilities and achieves performance comparable to systems that rely on external reward models.

1. Introduction

Large language models (LLMs) have demonstrated remarkable capabilities in reasoning-related tasks such as mathematics and coding. Notable examples include ChatGPT (OpenAI, 2023), Claude (Anthropic, 2023), and Gemini (Team et al., 2023). Following the release of GPT4-o1, LLMs with strong reasoning abilities have attracted even

more attention, along with inference methods that enhance reasoning. A particularly desirable property of such models is their ability to detect inconsistencies and errors in self-generated responses—based on feedback to their prior outputs—and correct these errors to produce improved responses. This process is often referred to as *self-correction* in the literature (Welleck et al., 2022; Madaan et al., 2024; Kim et al., 2024).

When an external ground-truth reward model is available, studies (Kim et al., 2024; Qu et al., 2024; Shinn et al., 2024) have shown that LLMs can refine their initial responses based on external gold reward feedback and determine when to terminate the self-correction loop. These approaches have proven effective for both mathematical reasoning and general agent tasks. Moreover, even when relying on imperfect proxy rewards, models can still achieve higher accuracy in revised responses by leveraging feedback from an outcome-based reward model (see Section 5 for empirical results). However, since these reward models are often themselves LLMs, deploying them requires running multiple models during inference, which increases computational costs and deployment complexity. In contrast, without external reward feedback, current LLMs struggle to refine their initial responses solely based on their intrinsic capabilities—a limitation known as intrinsic self-correction (Huang et al., 2023).

While reward models are traditionally trained with an additional scalar head for general-purpose chat (Ouyang et al., 2022; Bai et al., 2022; Touvron et al., 2023) and reasoning tasks (Cobbe et al., 2021a; Lightman et al., 2023), recent work suggests that LLMs themselves can generate reward signals in a generative way. For example, the LLM-as-a-judge approach (Zheng et al., 2023; Dubois et al., 2023) prompts the LLM to evaluate text outputs, effectively serving as a surrogate for human feedback. Another emerging direction explores generative reward models (Zhao et al., 2023; Dong et al., 2024; Zhang et al., 2024b; Mahan et al., 2024; Zhang et al., 2024a), which formulate evaluation tasks as instruction-following problems, using the probability of generating specific tokens as the reward value. These methods leverage LLMs’ next-token prediction capabilities, integrate the generation and evaluation into a unified framework.

^{*}Equal contribution ¹University of Illinois Urbana-Champaign
²University of Maryland, College Park. Correspondence to: Wei Xiong <wx13@illinois.edu>.

Building on these insights, this work investigates self-rewarding reasoning models that can incorporate three abilities within a single LLM: (i) generating step-by-step reasoning paths for given prompts, (ii) evaluating the correctness of generated responses, and (iii) revising and enhancing previous responses based on self-rewarding signals. Our key contributions are as follows:

1. **Self-rewarding reasoning framework.** We introduce a self-rewarding reasoning framework for LLMs, which integrates the generator and reward model into a single LLM, enabling autonomous reasoning, evaluation, and correction. This unification simplifies the model’s decision-making process and reduces computational overhead compared to external reward-based approaches.
2. **Algorithmic framework for self-correction.** We focus on the self-correction in mathematical reasoning and propose a two-stage framework that relies only on self-generated data. In the first stage, we use sequential rejection sampling to construct long chain-of-thought (CoT) trajectories that encode both self-rewarding and self-correction behaviors. Fine-tuning models on these trajectories enables them to detect the error in the self-generated responses and revise the previous attempts. In the second stage, we further enhance these patterns through reinforcement learning with rule-based signals.
3. **Empirical validation and analysis.** Through extensive experiments, we show that self-rewarding correction significantly outperforms intrinsic self-correction. Additionally, we conduct ablation studies to investigate the learning dynamics of the proposed framework, providing deeper insights into its behavior and effectiveness. The training codes and datasets are publicly available on GitHub¹.

2. Related Work

We review the works that are mostly related to our project in this section.

Self-rewarding alignment. Our work aligns with research on self-rewarding alignment (Yuan et al., 2024b; Prasad et al., 2024), where both of our project and their methods share similar spirits that we can unify the generation ability and evaluation ability into a single LLM. These methods leverage iterative DPO-type algorithms, where the model labels its own generated responses to provide training signals for subsequent iterations, enabling *self-improvement*. In

contrast, our approach does not focus on self-improvement during training. Instead, we rely on an external ground-truth reward model to provide learning signals in training. Our study emphasizes inference-time alignment for reasoning-focused LLMs, where self-rewarding signals are employed solely to guide inference rather than training.

Self-correction. Our work is closely related to self-correction in LLMs. We refer interested readers to the survey (Pan et al., 2023) for a more comprehensive review and only review some representative approaches that are mostly related to our project. Li et al. (2024) demonstrated that incorporating teacher model reflections into SFT data enhances students’ self-reflection abilities in general-purpose conversation tasks. However, for reasoning tasks, Huang et al. (2023) found that current LLMs—without additional training—fail to self-correct purely through intrinsic reasoning (i.e., prompting). This observation is also validated in Qu et al. (2024); Tyen et al. (2023); Zheng et al. (2024). A more in-depth analysis shows that most prior successful studies in this domain depend on **external** (ground-truth) reward models to determine when to initiate and terminate self-correction (Kim et al., 2024; Qu et al., 2024; Shinn et al., 2024; Madaan et al., 2024). Currently, there is no major work demonstrating that intrinsic self-correction (via prompting or fine-tuning) is reliably effective. Furthermore, because external reward models are typically LLM-based, these methods introduce additional computational overhead by requiring a multi-agent system for inference.

Recognizing this challenge, our study explores how LLMs can autonomously evaluate response quality and correct errors without external reward models. Specifically, we introduce a self-rewarding reasoning framework that enables a single LLM to perform error detection and self-correction effectively. Among the works in self-correction, the most relevant work is the recent Kumar et al. (2024), which employed a multi-turn deep RL approach to train self-correcting models. In comparison, this work introduces a new and general self-rewarding formulation for reasoning-focused LLMs, with self-correction as a representative application. Compared to the intrinsic correction and the framework in Kumar et al. (2024), one major difference is that our framework equips models with self-rewarding ability, enabling our models to intelligently scale inference compute by *selectively* revising the first attempts, which helps to reduce computational overhead by avoiding unnecessary iterations. We will also design experiments to illustrate this idea.

Algorithmically, our approach also differs from Kumar et al. (2024). We first use sequential rejection sampling to construct long CoT trajectories with both self-rewarding and self-correction patterns, which serve as warm-up fine-tuning data. We then enhance these behaviors through reinforcement learning (using either DPO-type algorithms or PPO)

¹<https://github.com/RLHFlow/Self-rewarding-reasoning-LLM>

with rule-based signals. In contrast, Kumar et al. (2024) employed RLOO (Ahmadian et al., 2024) with a specialized reward function for a two-turn self-correction task. While their no-public models (Gemini) and implementation details (parameters, codes) do not enable comparison, we believe that the multi-turn RL methods proposed by Kumar et al. (2024) could also complement the proposed self-rewarding framework, and achieve better reasoning performance compared to the standard reasoning models.

Rule-based RL for LLMs mathematical reasoning.

Rule-based reinforcement learning has received significant attention following the success of DeepSeek-R1 (DeepSeek-AI et al., 2025). Open-source efforts have since attempted to replicate its performance using Qwen models (Yang et al., 2024), including works such as Zeng et al. (2025); Cui et al. (2025); Zhang et al. (2025). These methods train LLMs using only the correctness score (whether the final answer is correct or not) and a format score (whether the final answer is output in a pre-determined format), in contrast to the previous works with the neural network-based reward model (Cobbe et al., 2021a; Lightman et al., 2023; Zhang et al., 2024a). In particular, DeepSeek-AI et al. (2025) observed that self-correction naturally emerges during RL training (referred to as an AHA moment in their report). However, our preliminary experiments, along with open-source replications using Qwen-2.5-Math (Liu et al., 2025; Zhang et al., 2025; Cheng et al., 2025), suggest that (i) the base models already exhibit some self-correction ability, though it is quite sparse. (ii) vanilla rule-based RL **cannot** consistently enhance self-correction without additional design.

Interestingly, even when using the same algorithms and data, similar improvements in mathematical reasoning are not observed in models such as Llama (Meta, 2024; Touvron et al., 2023). We hypothesize that Qwen-2.5-Math and DeepSeek-R1 benefit from extensive pre-training on high-quality mathematical corpora (e.g., 1T tokens for Qwen-2.5-Math (Yang et al., 2024)), and that the AHA moment may stem from carefully curated data containing self-correction patterns in pre-training or a cool-down stage. Since these datasets are non-public, the exact details remain unknown.

In contrast, our study shows that a warm-up stage using a carefully curated SFT dataset (collected via sequential rejection sampling) enables models to learn self-correction patterns more reliably. This foundation allows rule-based RL to further enhance these behaviors in a stable manner. We also remark that our two-stage framework and most of the associated experiments are performed prior to the release of DeepSeek-R1.

3. Self-rewarding Reasoning Language Models

We formulate the self-rewarding reasoning process as a multi-turn Markov Decision Process (MDP). After observ-

ing an initial prompt $s^1 = x \in \mathcal{X}$ from some distribution d_0 , an LLM, denoted as π , will generate an initial reasoning attempt $a^1 \sim \pi^1(\cdot|s^1)$ from the action space \mathcal{A} . The LLM then self-rewards its response by generating an evaluation:

$$y^1 \sim \pi^1(\cdot|s^1, a^1).$$

If the model assesses its answer as correct ($y^1 = [\text{VERIFY}]$ correct, details provided later), the generation stops. Otherwise, the LLM proceeds to the next step, generating a refined response and evaluation:

$$(a^2, y^2) \sim \pi^2(\cdot|s^2),$$

where the generation is conditioned on the updated state $s^2 = (s^1, a^1, y^1)$. The self-refinement process continues until the model produces a self-evaluation y^h that assesses the answer as correct.

We assume that we have access to the ground-truth verifier $r^* : \mathcal{X} \times \mathcal{A} \rightarrow \{0, 1\}$, which determines whether a response is correct. Throughout this study, we use the ToRA verification script (Gou et al., 2023), built on the Python library SymPy for symbolic mathematics. We also present a representative Example 1 to illustrate the process.

Two-stage training framework. Following standard post-training practices for LLMs, we adopt a two-stage approach:

1. **Self-rewarding instruction-following fine-tuning (IFT).** Starting with an initial LLM π_0 (e.g., a general-purpose chatbot), we collect demonstration data by a sequential rejection sampling process and fine-tune π_0 to get an improved model π_{ref} , which integrates self-rewarding reasoning abilities.
2. **Reinforcement learning (RL) optimization.** We further refine π_{ref} using RL, leveraging it as the reference model. This stage can further enhance the model’s ability to assess correctness and refine previous responses.

3.1. Self-rewarding Instruction-following Fine-tuning

Self-rewarding by token prediction. To train the LLMs to evaluate the reasoning steps, we formulate this task as an instruction-following task, following prior works (Zhao et al., 2023; Dong et al., 2024; Liu et al., 2023; Ye et al., 2024; Wang et al., 2024; Zhang et al., 2024b). Specifically, we allow models to include reasoning in their evaluations while requiring them to output specific tokens to indicate their evaluation results. We experimented with different token choices, such as: (i) a prompt “Is the most recent final answer correct (Yes or No)?” with “Yes” and “No” as the response tokens, as used in (Xie et al., 2023; Zhang et al., 2024b); (ii) explicit markers such as “[VERIFY] correct” and “[VERIFY] wrong”. Our experiments show no

User: A particular star has an absolute magnitude $M = -7$. If this star is observed in a galaxy that is at a distance of 3Mpc, what will its apparent magnitude be?

Please reason step by step, and put your final answer within \square . (prompt x)

Assistant: To determine the apparent magnitude of the star, we can use the distance modulus formula : $m - M = 5 \log_{10}(d) - 5$ where: m is the apparent magnitude, M is the absolute magnitude, d is the distance to the star in parsecs.... The apparent magnitude of the star, when observed from a distance of 3 Mpc, is approximately 7.58. So the final answer is: $\square{7.58}$. (initial response a^1)

To verify the correctness of the solution, let's recheck the steps ... So the final answer is: $\square{20.39}$. The self-evaluation is: [VERIFY] wrong. (self-rewarding evaluation y^1)

Let's correct it step by step. Given: Absolute magnitude ... The correct apparent magnitude is approximately 20.39. So the final answer is: $\square{20.39}$.

(revised response a^2).

Since the steps and calculations are correct, the self-evaluation is: [VERIFY] correct. (self-rewarding evaluation y^2)

Table 1. An example of the self-rewarding reasoning path. We omit the detailed reasoning path for a clear presentation. The full trajectory is available at Table 13 in Appendix.

significant performance differences between these choices. During inference, rather than using the likelihood of “Yes” as a reward (as in (Zhao et al., 2023; Dong et al., 2024; Zhang et al., 2024b)), we sample the evaluation token from the distribution. This allows us to use a standard inference pipeline without any specific adjustment. See Table 1 for an example.

Remark 3.1. We choose these specific tokens primarily for research simplicity. However, we expect that similar results can be achieved even if these special tokens are replaced with more natural language expressions, such as “wait”, “aha”, or “let me re-check the answer”, where one can also leverage the LLMs to complete this paraphrasing process.

Data collection by sequential rejection sampling. We employ a rejection sampling approach, similar to STaR (Zelikman et al., 2022) and RAFT (Dong et al., 2023), where we generate a large amount of self-correction trajectories and only preserve the desired trajectories. The major difference is that since the self-correction behavior is sparse in base models and self-rewarding pattern is missing, it is unlikely to collect the desired trajectory directly. In view of this, we sequentially prompt the base model and generate different steps separately. Then, we combine them into long CoT trajectories that incorporate both self-rewarding and self-correction patterns.

Our data collection process consists of the following steps:

1. **Generating initial reasoning responses:** training prompts from datasets such as MATH (Hendrycks et al., 2021) and GSM8K (Cobbe et al., 2021a) and sample $N_1 = 50$ initial responses a^1 per prompt as our base

trajectories (see Section 5 for details of experiment setups).

2. **Self-rewarding signal sampling:** For each prompt and initial response, we further sample $N_2 = 8$ self-evaluations and keep only one evaluation result that is the same as the ground truth. Then, we split them into G^{correct} and G^{wrong} using the ground-truth verifier r^* .
3. **Correction sampling:** For each prompt and initial response in G^{wrong} , we sample $M_1 = 8$ completions by providing the feedback that the initial response was wrong to collect trajectories that successfully revise incorrect responses. For each prompt and initial response in G^{correct} , however, we also tell the model that the response was incorrect and collect $M_2 = 4$ completions. By doing so, we want to additionally collect “correct-to-correct” trajectories in the face of wrong judgment.

Eventually, we collect $8 \times |G^{\text{wrong}}| + 4 \times |G^{\text{correct}}|$ full trajectories. Then, we filter the dataset and only keep the following types of data:

- $\mathcal{D}_1^{\text{IFT}}$: wrong a^1 , $y^1 = [\text{VERIFY}]$ wrong, correct a^2 ;
- $\mathcal{D}_2^{\text{IFT}}$: correct a^1 , $y^1 = [\text{VERIFY}]$ wrong, correct a^2 ;
- $\mathcal{D}_3^{\text{IFT}}$: correct a^1 , $y^1 = [\text{VERIFY}]$ correct.

We provide an example of data collection process in Table 2. We limit the horizon to two iterations due to resource constraint, and preserve at most one trajectory per base sample

System: You are a mathematical reasoning assistant. For each problem, follow these steps strictly: 1. Solve the problem using step-by-step reasoning and output the final answer within \square . Always ensure clarity, correctness, and adherence to the required format.

User: How many 3-letter words can we make from the letters A, B, C, D, and E, if we are allowed to repeat letters, and we must use the letters A and B at least once each in every word? (Here, a word is an arbitrary sequence of letters.) Let's think step by step and output the final answer within \square .

Assistant: Here's how we can break it down: 1. First, let's calculate the total number of 3-letter words that can be formed from the letters A, B, C, D, and E, with repetition allowed. ... 7. Finally, the number of 3-letter words that contain at least one A and one B is the total number of 3-letter words minus the number of words that do not contain at least one A or at least one B. The number of 3-letter words that can be made from the letters A, B, C, D, and E, where each word must contain at least one A and one B, is $\boxed{38}$.

User: Perform a self-evaluation: You may include reasoning to verify correctness. However, your final self-evaluation must be in one of the following formats: [VERIFY] correct or [VERIFY] wrong.

Assistant: Let's perform a self-evaluation of the solution. We first calculate ... This is $125 - 101 = 24$. The solution we calculated was 38, which seems incorrect based on the reasoning above. The correct answer should be 24. **Let's output the self-evaluation: [VERIFY] wrong.**

User: please identify the mistake in your previous reasoning, revise your reasoning path and output a corrected final answer within \square

Assistant: Let's correct the mistake in the previous reasoning. The correct steps should be as follows: 1. Calculate ... 6. Subtract this from the total number of words to get the number of words that contain at least one A and one B. This is $125 - 101 = 24$. The correct answer is indeed $\boxed{24}$.

Table 2. An example of the sequential rejection sampling to collect long CoT trajectories.

to control dataset size. Then we fine-tune the LLMs using standard SFT pipeline to maximize:

$$\begin{aligned} & \sum_{\mathcal{D}_1^{\text{IFT}}} [\log P(y^1|x, a^1) + \log P(a^2|x, a^1, y^1)] \\ & + \sum_{\mathcal{D}_2^{\text{IFT}}} \log P(a^2|x, a^1, y^1) + \sum_{\mathcal{D}_3^{\text{IFT}}} \log P(y^1|x, a^1). \end{aligned} \quad (1)$$

In practice, however, we observe that the multi-task training can lead to stability issue and can slightly hurt the first-round performance. To mitigate this issue, we also train on the correct attempt a^1 for the samples in $\mathcal{D}_3^{\text{IFT}}$.

3.2. KL-regularized Reinforcement Learning

In this stage, we aim to further enhance the self-rewarding IFT models using reinforcement learning. We consider both deep RL methods (Schulman et al., 2017) and direct alignment algorithms (Zhao et al., 2023; Rafailov et al., 2023; Azar et al., 2023; Liu et al., 2023).

Learning signal. To facilitate the reinforcement learning stage, we assume there exists a trajectory-wise reward function $u^*(\tau)$ for trajectory

$$\tau = (x, a^1, y^1, \dots, a^H, y^H).$$

However, instead of learning a proxy reward from data like the BT model in RLHF (Ouyang et al., 2022) or outcome-supervised reward (ORM) in previous mathematical reasoning literature (Lightman et al., 2023), we primarily use the oracle reward

$$u^*(\tau) = r^*(x, a^H),$$

i.e., whether the final result is correct or not. The main advantage is that the oracle reward can largely mitigate the risk of reward hacking. This is also referred to as the *rule-based RL* in the very recent literature (DeepSeek-AI et al., 2025). We will also study the additional rule designs for either reward value assignment (PPO training) or data ranking (DPO training), where an implicit u^* is determined by the set of rules we use.

Following standard RLHF methodologies (Ouyang et al., 2022; Bai et al., 2022), we optimize the following KL-regularized objective:

$$\begin{aligned} & \max_{\pi \in \Pi} \mathbb{E}_{x \sim d_0, a^1 \sim \pi^0(\cdot|x)} \mathbb{E}_{\tau \sim \pi(\cdot|x, a^1)} \left[\right. \\ & \quad \left. u^*(\tau) - \eta \sum_{h=1}^H D_{\text{KL}}(\pi^h(\cdot|s^h), \pi_{\text{ref}}^h(\cdot|s^h)) \right]. \end{aligned} \quad (2)$$

The optimal policy, as well as its associated optimal value satisfies the following optimality condition (Xiong et al.,

2024a; Xie et al., 2024a; Zhong et al., 2024).

Proposition 3.2. *We can recursively define the following optimal value functions and optimal policies for a KL-regularized MDP with horizon H and deterministic external observation. For Q value, we have*

$$Q_h^*(s^h, a^h, y^h) = \begin{cases} u^*(s^H, a^H, y^H), & \text{if } h = H, \\ V_{h+1}^*(s_{h+1}), & \text{if } h \leq H - 1. \end{cases} \quad (3)$$

Also, for all $h \in [H]$, we have:

$$V_\star^h(s^h) = \underbrace{\eta \log \mathbb{E}_{a^h, y^h \sim \pi_{\text{ref}}^h(\cdot | s^h)} \exp \left(\frac{Q_\star^h(s^h, a^h, y^h)}{\eta} \right)}_{=: Z^h(s^h)},$$

$$\pi_\star^h(a^h, y^h | s^h) = \frac{\pi_{\text{ref}}^h(a^h, y^h | s^h)}{Z^h(s^h)} \cdot \exp \left(\frac{Q_\star^h(s^h, a^h, y^h)}{\eta} \right). \quad (4)$$

We remark that one advantage of the proposition is that it allows deterministic external message (e.g. instruction prompts) in the state update, which will be useful when we consider a simplified research framework in Section 5.

We also adopt Direct Preference Optimization (DPO) (Rafailov et al., 2023; Azar et al., 2023; Zhao et al., 2023; Ethayarajh et al., 2024) to solve Equation 2, primarily due to computational constraints. In particular, we use the multi-turn DPO (M-DPO) framework from Xiong et al. (2024a), since it allows deterministic external observation in the state transition. To facilitate direct preference learning and bypass explicit reward training, we impose the following trajectory-level Bradley-Terry (BT) preference structure (Bradley & Terry, 1952). Specifically, given two trajectories τ^1, τ^2 , the probability of τ^1 being preferred than τ^2 , denoted as $\tau^1 \succ \tau^2$, is

$$\mathbb{P}(\tau^1 \succ \tau^2 | \tau^1, \tau^2) = \sigma(u^*(\tau^1) - u^*(\tau^2)),$$

where $\sigma(z) = 1/(1 + \exp(-z))$ is the sigmoid function. Following Xiong et al. (2024a), we take log on both sides of (4), and connect a utility function u_θ with associated policy π_θ and value V_θ :

$$\log \frac{\pi_\theta^h(y^1 | s^1)}{\pi_{\text{ref}}^1(y^1 | s^1)} = V_\theta^2(s^2) - V_\theta^1(s^1),$$

$$\log \frac{\pi_\theta^h(a^h, y^h | s^h)}{\pi_{\text{ref}}^h(a^h, y^h | s^h)} = Q_\theta^h(s^h, a^h, y^h) - V_\theta^h(s^h).$$

For a pair of trajectories τ^w, τ^l where $\tau^w \succ \tau^l$, we have

$$u_\theta(\tau^w) - u_\theta(\tau^l) = \log \frac{\pi_\theta^1(y_w^1 | x, a^1)}{\pi_{\text{ref}}^1(y_w^1 | x, a^1)} - \log \frac{\pi_\theta^1(y_l | x, a^1)}{\pi_{\text{ref}}^1(y_l | x, a^1)} + \sum_{h=1}^H \log \frac{\pi_\theta^h(a_w^h, y_w^h | s_w^h)}{\pi_{\text{ref}}^h(a_w^h | s_w^h)} - \log \frac{\pi_\theta^h(a_l^h, y_l^h | s_l^h)}{\pi_{\text{ref}}^h(a_l^h | s_l^h)}.$$

Taking this reward difference parameterization into the log-likelihood of the BT model $\sum_{(\tau^w, \tau^l) \in \mathcal{D}} \log \sigma(u_\theta(\tau^w) - u_\theta(\tau^l))$, we obtain the loss function $\mathcal{L}_{\text{M-DPO}}(\theta)$:

$$- \sum_{(\tau^w, \tau^l) \in \mathcal{D}} \log \sigma \left(\eta \left[\log \frac{\pi_\theta^1(y_w^1 | x, a^1)}{\pi_{\text{ref}}^1(y_w^1 | x, a^1)} - \log \frac{\pi_\theta^1(y_l | x, a^1)}{\pi_{\text{ref}}^1(y_l | x, a^1)} + \sum_{h=1}^H \log \frac{\pi_\theta^h(a_w^h, y_w^h | s_w^h)}{\pi_{\text{ref}}^h(a_w^h | s_w^h)} - \log \frac{\pi_\theta^h(a_l^h, y_l^h | s_l^h)}{\pi_{\text{ref}}^h(a_l^h | s_l^h)} \right] \right). \quad (5)$$

4. Experiment Results

Task, datasets, and data format. We evaluate models' mathematical reasoning abilities using standard benchmarks, including MATH500 (Hendrycks et al., 2020), Olympiad-Bench (He et al., 2024), and Minerva Math (Lewkowycz et al., 2022). These datasets provide a moderate size for reliable and efficient model evaluation, covering topics such as algebra, geometry, probability, number theory, and calculus. For training, we mainly use the prompts in NumiaMath-CoT dataset (Beeching et al., 2024). Specifically, we use a 50K subset for the self-rewarding IFT stage, a 10K subset for validation and model selection, and the remaining data for RL training. During inference, the model generates up to 4096 tokens, with VLLM 0.5.4 (Kwon et al., 2023) accelerating the process.

Evaluation metrics. We employ two categories of metrics to evaluate our models: (1) mathematical reasoning and self-correction and (2) reward model accuracy. First, we follow Kumar et al. (2024) to consider the following metrics to evaluate the models' ability of mathematical reasoning and self-correction.

1. **Turn 1:** accuracy of the first attempt;
2. **Final accuracy:** accuracy of the final answer;
3. $\Delta(t_1, t_2)$: improvement in accuracy from the first attempt to the final answer;
4. $\Delta^{i \rightarrow c}(t_1, t_2)$: fraction of problems changed from incorrect to correct;
5. $\Delta^{c \rightarrow i}(t_1, t_2)$: fraction of problems changed from correct to incorrect.

Due to the nature of the self-rewarding reasoning framework, we additionally include the metrics to measure the accuracy as a reward model. We also defer a more comprehensive understanding of the proposed framework with a slightly simplified template to next section, where we will additionally compute the ratio of modifying a correct answer to incorrect when facing a misleading reward.

1. **RM Accuracy** (a, b) : class-dependent accuracy for

Table 3. Main results of experiments with Qwen2.5-Math-7B-base. The single-turn baselines are used to train a regular CoT reasoning model. The baselines with [†] perform self-correction under the external prompt, where training may apply to enhance this ability. We use greedy decoding following the convention of the recent open-source projects on mathematical reasoning.

Benchmark	Method	Turn 1	Final Accuracy	$\Delta(t_1, t_2)$	$\Delta^{i \rightarrow c}(t_1, t_2)$	$\Delta^{c \rightarrow i}(t_1, t_2)$
MATH500	Single-turn STaR/RAFT	77.0	77.0	-	-	-
	Single-turn DPO	76.8	76.8	-	-	-
	Single-turn PPO	79.4	79.4	-	-	-
	Prompt with Gold RM [†]	65.4	66.8	1.4	1.4	0.0
	Intrinsic self-correction [†]	65.4	51.4	-14.0	1.4	15.4
	STaR/RAFT for self-correction [†]	71.6	70.4	-1.2	5.0	6.2
	STaR/RAFT+ for self-correction [†]	72.0	71.2	-0.8	3.0	3.8
	Self-rewarding IFT	72.6	77.2	4.6	5.0	0.4
	Self-rewarding IFT + DPO w correctness	72.8	78.6	5.8	6.0	0.2
	Self-rewarding IFT + PPO w correctness	75.8	80.2	4.4	4.8	0.4
OlympiadBench	Single-turn STaR/RAFT	40.1	40.1	-	-	-
	Single-turn DPO	39.0	39.0	-	-	-
	Single-turn PPO	39.5	39.5	-	-	-
	Prompt with Gold RM [†]	23.4	25.6	2.2	2.2	0
	Intrinsic self-correction [†]	23.4	18.1	-5.3	2.2	7.5
	STaR/RAFT for self-correction [†]	36.5	32.5	-4.0	7.2	11.2
	STaR/RAFT+ for self-correction [†]	35.7	35.5	-0.2	3.2	3.4
	Self-rewarding IFT	35.4	39.4	4.0	4.7	0.7
	Self-rewarding IFT + DPO w correctness	37.6	40.1	2.5	3.5	1.0
	Self-rewarding IFT + PPO w correctness	41.0	43.4	2.4	2.8	0.4
Minerva Math	Single-turn STaR/RAFT	32.0	32.0	-	-	-
	Single-turn DPO	31.6	31.6	-	-	-
	Single-turn PPO	33.1	33.1	-	-	-
	Prompt with Gold RM [†]	9.9	11.7	1.8	1.8	0
	Intrinsic self-correction [†]	9.9	8.4	-1.5	1.8	3.3
	STaR/RAFT for self-correction [†]	28.7	29.4	0.7	1.8	1.1
	STaR/RAFT+ for self-correction [†]	25.7	25.3	-0.4	0.8	1.2
	Self-rewarding IFT	23.2	28.7	5.5	7.3	1.8
	Self-rewarding IFT + DPO w correctness	26.8	34.6	7.8	9.6	1.8
	Self-rewarding IFT + PPO w correctness	34.0	38.4	4.4	5.1	0.7

correct and incorrect trajectories. In other words, a is the true positive rate and b is the true negative rate;

2. **Ratio** $p^{c \rightarrow i}(t_1, t_2)$: probability of modifying a correct answer to incorrect when facing a misleading reward.

For all evaluations, we use zero-shot CoT prompting and greedy decoding following the convention of recent projects with Qwen-2.5-Math models.

Experiment setup of self-rewarding IFT. We use Qwen2.5-Math-7B-base as the base model, which is continuously pre-trained on extensive mathematical and instruction-following data. Sequential rejection sampling (introduced in Section 3.1) is used for data collection, resulting in a dataset of 32K trajectories, where we roughly balance between correct and incorrect first attempts. In fine-tuning, samples are packed into 8192-token blocks and we use a learning rate of $1e-5$, a cosine scheduler, and a 0.05 warm-up ratio. Global batch size is set to be 32. We train the models for three

epochs and eventually select the one at the end of the first epoch.

Experiment setup of reinforcement learning. For iterative DPO training, we adopt setups from Xiong et al. (2024a) with a learning rate of 2×10^{-7} , a cosine scheduler, and a batch size of 32. We tune $\eta \in \{0.1, 0.5\}$ and also train with and without an NLL loss in the DPO objective (Pang et al., 2024; Xie et al., 2024a; Liu et al., 2024). For each iteration, we use 20K prompts and collect 8 responses per prompt. Then, we extract the comparison pairs using the correctness score. If all responses admit the same score, we skip the prompt. A 10K validation set from NuminaMath-CoT is used for model selection. The primary metric for model selection is accuracy at turn 2. When models achieve comparable turn-2 accuracy, we choose the models with higher $\Delta(t_1, t_2)$ improvement. The best model of these training setups is used as the representative model. For PPO training, we mainly follow a public example script of verL

(Sheng et al., 2024), which is publicly available².

Baseline: improving the self-correction ability. We consider several baseline methods in the self-correction literature, including training-free approaches and fine-tuning. For training-free methods, we evaluate intrinsic self-correction (Huang et al., 2023), where models rely solely on prompting to perform correction, and self-correction with external ground-truth rewards (Qu et al., 2024). The prompts used for these methods are provided in Appendix B. We also include STaR and RAFT approaches (Zelikman et al., 2022; Dong et al., 2023), which are inspired by expert iteration in reinforcement learning (Anthony et al., 2017). These methods generate numerous trajectories with the base model, filter out failed attempts, and fine-tune on successfully revised responses. Following Kumar et al. (2024), we study a variant, STaR/RAFT+, which augments the training set with a set of correct-to-correct trajectories. To ensure a fair comparison, the total number of training samples for STaR/RAFT(+) is kept the same as in our self-rewarding IFT stage.

Baseline: improving the single-turn reasoning ability. In addition, we also consider several baselines that improve the models’ single-turn reasoning ability without self-correction. These methods include the STaR/RAFT (Zelikman et al., 2022; Dong et al., 2023), iterative DPO (Xiong et al., 2023) with the correctness score to rank data, and PPO with the correctness score. In particular, we adopt the iterative algorithms in the implementations of the STaR/RAFT and DPO because we observe that they achieve much better performance to serve as competitive baselines. We start from Qwen-2.5-Math-7B and train with only self-generated data for a fair comparison. We remark that the Qwen-2.5-Math-7B has been trained on many instruction-following data in the pre-training stage and the recent open-source projects also show that it can be used as the starting checkpoint without distillation from larger LLMs or human instructions (Zeng et al., 2025; Zhang et al., 2025).

4.1. Main Results

We report the main results in Table 3. Note that there can be an error of 0.1 due to rounding.

Intrinsic self-correction with prompting fails in general.

We first observe that intrinsic self-correction without explicit reward signals typically reduces final test accuracy. Upon analyzing the outputs, we find that models tend to modify their initial responses regardless of its correctness, as they lack a mechanism to determine when to refine their answers versus when to terminate the correction

process. Moreover, even when given ground-truth rewards, base models with prompting alone achieve only marginal improvement in incorrect-to-correct transitions $\Delta^{i \rightarrow c}(t_1, t_2)$. For example, on MATH-500 benchmark, prompting with gold reward only leads to $\Delta^{i \rightarrow c}(t_1, t_2) = 1.4\%$.

We also notice that the STaR/RAFT method, which fine-tunes models on revised incorrect attempts, fails to significantly improve performance. It increases $\Delta^{i \rightarrow c}(t_1, t_2)$ (incorrect-to-correct transitions) on MATH500 from 1.4% to 5.0%, but still suffers from a $\Delta^{c \rightarrow i}(t_1, t_2)$ (correct-to-incorrect transitions) of 6.2%. Additionally, the STaR/RAFT+ variant, which includes correct-to-correct trajectories, becomes more conservative in modifying the initial attempt. While this reduces incorrect corrections ($\Delta^{c \rightarrow i}(t_1, t_2)$), it also lower $\Delta^{i \rightarrow c}(t_1, t_2)$, ultimately degrading test accuracy. These findings align with prior studies, and highlight the limitations of intrinsic self-correction, even with training (Huang et al., 2023; Kumar et al., 2024).

Self-rewarding reasoning models significantly outperform existing baselines of self-correction.

Across all tasks, self-rewarding reasoning models consistently improve final accuracy with higher $\Delta(t_1, t_2)$ compared to baseline methods. We notice that fine-tuning on the synthetic trajectories with self-correction behavior yields models with much higher $\Delta^{i \rightarrow c}(t_1, t_2)$, suggesting that the models are more good at correcting the error in the self-generated responses. Distinct from the STaR/RAFT, models trained with self-rewarding IFT also exhibit significantly lower $\Delta^{c \rightarrow i}(t_1, t_2)$, indicating they are better at recognizing when to stop due to the additional self-rewarding signals. For instance, on MATH500,

- self-rewarding IFT achieves $\Delta^{i \rightarrow c} = 5.0\%$ (vs. 1.4% for intrinsic self-correction);
- self-rewarding IFT achieves $\Delta^{c \rightarrow i} = 0.4\%$ (vs. 15.4% for intrinsic self-correction and 3.8% for STaR/RAFT+);

Since STaR/RAFT(+) and self-rewarding IFT use the same data synthesis approach (rejection sampling) but under different self-correction frameworks, these results highlight the advantage of our self-rewarding reasoning framework.

Self-rewarding reasoning models improve the final accuracy compared to the single-turn baselines.

We also compare the self-rewarding reasoning models with RL training against their single-turn counterparts. For both the PPO and DPO, the self-rewarding reasoning models achieve higher final test accuracy due to the additional correction step. For instance, the self-rewarding IFT + PPO yields a model with 43.4% final accuracy on OlympiadBench, and 38.4% on Minerva Math, compared to the 39.5% and 33.1%

²https://github.com/RLHFlow/Online-DPO-RL/blob/main/ppo_training/ver1_example.sh

Table 4. The results of reward modeling accuracy (%). We report the accuracy of self-rewarding signals for the three benchmarks in two separate classes. For instance, MATH-500 C is the accuracy of recognizing a correct trajectory, while MATH-500 W is the accuracy of recognizing a wrong trajectory. The model highlighted by (*) is selected as the final model.

Method	MATH-500 C	MATH-500 W	OlympiadBench C	OlympiadBench W	Minerva Math C	Minerva Math W
Self-rewarding IFT	93.0	47.7	89.6	45.9	91.7	36.1
PPO Step 100	97.5	56.4	98.1	33.5	87.4	29.7
PPO Step 220 (*)	98.6	47.6	97.8	39.3	94.2	32.4
DPO Iter 2	91.3	56.2	81.9	51.8	86.7	36.2
DPO Iter 5 (*)	92.0	50.6	88.2	44.5	92.4	37.4

of the single-turn counterpart. Similarly, with the DPO, the self-rewarding reasoning models achieve a 78.6% on MATH500, a 40.1% on OlympiadBench, and 34.6% on Minerva Math, while the single-turn DPO model admits 76.8%, 39.0%, 31.6%, respectively.

However, self-rewarding models use more tokens at inference due to the additional correction step. For a fair comparison, we will also study the behavior of self-rewarding correction under scaled test-time compute budgets in Section 5.

Deep RL algorithm outperforms the direct alignment algorithms. We observe that PPO outperforms iterative DPO by a large margin. For example, the PPO-trained model achieves a 43.4% final accuracy on OlympiadBench, compared to the 40.1% of the DPO method. This suggests that when absolute reward signals are available, enforcing a preference structure (Bradley-Terry model) is unnecessary and may degrade performance. Another possible reason is the limited data utilization in DPO. We notice that, with our setup, we can collect comparison pairs for only 40% to 60% prompts. For the remaining prompts, models either generate no correct trajectories or all trajectories are correct. As a result, DPO utilizes less training data than PPO, which may contribute to its lower accuracy.

Reward model (RM) accuracy. Since our self-rewarding framework unifies the generator and reward model, we evaluate the accuracy of our models as a reward model. We observe that the Qwen2.5-Math-7B-base can fail to strictly follow the format by omitting the self-evaluation step or not generating the evaluation result in the pre-determined format possibly because the model is not instruction-following fine-tuned. However, this happens in less than 10% of the cases so we focus on the samples with the evaluation step and also further involve human supervision to summarize the statistics. We report the result in Table 4. We observe that the self-rewarding IFT model is much more good at recognizing the correct trajectories, as the accuracy is generally higher than 90%, even though we balance the two types of trajectories in the training set. This directly leads to the

small $\Delta^{c \rightarrow i}(t_1, t_2)$ we observe in the main table.

We also notice that the RL training (both PPO and DPO) does not consistently improve the reward modeling accuracy. Analysis of PPO checkpoints (initial model, Step 100 and Step 220) clearly shows a trade-off between correct and incorrect classification accuracy. The PPO training explores different trade-off between them, with the goal of maximizing the final accuracy. Similar observation also applies to the DPO training. Moreover, the best model of PPO training tends to prioritize recognizing correct trajectories, at the cost of lower accuracy in identifying incorrect responses, which aligns with the lower $\Delta^{c \rightarrow i}(t_1, t_2)$ and also lower $\Delta^{i \rightarrow c}(t_1, t_2)$. This may be because correcting an incorrect answer is generally more challenging than maintaining a correct initial response. We defer a more detailed study of the impact of data composition on reward modeling accuracy to the next section.

Learning dynamic of the RL stage. While the RL training improves final accuracy, the final test accuracy is determined by both the turn-1 accuracy and $\Delta(t_1, t_2)$. In particular, we notice that the final accuracy gains come primarily from the higher turn-1 accuracy, as the models after the RL training usually admit a much higher turn-1 accuracy, but also a lower $\Delta^{i \rightarrow c}(t_1, t_2)$. To understand the learning dynamic of the RL training, we plot test accuracy on three benchmarks in terms of the RL training steps in Figure 1. We observe that in the early stage of the RL training, both the turn-1 accuracy and the final accuracy increase, and their gap $\Delta(t_1, t_2)$ is also increased or maintained as a stable level. This indicates that the models learn to use their knowledge better in the first round and improve or maintain a comparable level of correction ability. Around training step 100, however, the increase of the final accuracy is mainly from the higher turn-1 accuracy and their gap is narrowed, indicating less reliance on self-correction.

We also plot the average generation length in the first figure. Initially, the length decreases because the Qwen2.5-Math-7B-base model tends to generate many python codes, resulting in lengthy responses. We observe that the code

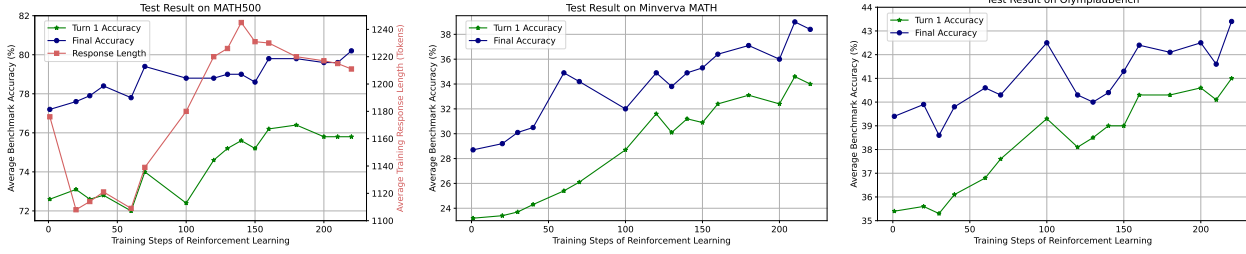


Figure 1. The learning dynamic of the PPO training, initialized from the self-rewarding IFT model. We also plot the average generation length during the training in the first figure.

usually takes many tokens and can lead to incomplete reasoning path and it is discouraged by the reward signal. This observation is consistent with Zeng et al. (2025). Then, the length increases in the next stage, indicating that the reflection and self-correction abilities are also encouraged by the RL training. Finally, the length decreases again, along with a higher turn-1 accuracy and a smaller $\Delta(t_1, t_2)$, indicating that the models learn to provide a correct answer in their first attempt and also, the self-correction pattern is discouraged. This is also supported by the reward model accuracy, where the RL-trained models tend to be more conservative and evaluate the attempt as correct.

5. More Experiment Results with a Two-turn Conversation Framework and Llama Models

In this section, we continue to investigate the self-rewarding reasoning framework.

5.1. Data Format: Simplified Two-turn Framework

Previously, we combined multiple reasoning steps into a single long CoT trajectory, which aligns with common practice. However, this approach poses significant challenges for our study, as models—particularly Qwen2.5-Math-7B-base—often fail to strictly follow instructions for evaluating or revising responses based on their history. For instance, models sometimes will also generate the evaluation results using \square or not to correct the responses even though the self-evaluation result is “[VERIFY] wrong”. Additionally, models can perform multiple rounds of self-evaluation and correction, but these steps are tightly coupled and cannot be easily decoupled into separate stages.

To address these issues, we adopt a simplified two-turn conversation framework, where the user provides explicit instructions between different steps. Specifically, after receiving the mathematical problem, the model will first generate the CoT reasoning a^1 and self-evaluation y . Then, the user provide a deterministic instruction o based on the self-evaluation y :

1. Since your initial response is self-evaluated as incorrect, there might be an error in the solution above because of lack of understanding of the question. Please correct the error, if any, and rewrite the solution. Put your final answer within \square ;
2. Since your initial response is self-evaluated as correct, confirm it and provide no further modifications. Put your final answer within \square .

Meanwhile, when collecting the data, the self-rewarding signal is determined directly by the ground-truth oracle reward with the template designed in Zhang et al. (2024b), without additional reasoning. While this simplification may reduce reward modeling accuracy (Zhang et al., 2024b), it facilitates controlled experimentation by allowing modifications to the self-rewarding signal. Similar frameworks—without the self-rewarding component—have been explored in previous works (Huang et al., 2023; Kumar et al., 2024). See Table 6 for an illustrative example.

5.2. Experiment Setup

Base model, task, and datasets. Qwen2.5-Math-7B-base serves as a strong and specialized base model, which is pre-trained on a large mathematical corpus. To ensure generality and a more comprehensive evaluation, we experiment with the Llama model series. Specifically, our base models include Llama-3-8B-it and Llama-3-SFT, the latter being fine-tuned on Open-MathInstruct2-1M (Toshniwal et al., 2024a). While both models are generally weaker than Qwen2.5-Math-7B-base, Llama-3-SFT is stronger than Llama-3-8B-it.

In this section, we evaluate the models’ mathematical reasoning abilities using the MATH and GSM8K benchmarks, which are well-suited to their capacities. For MATH, we use 7.5K training problems during the self-rewarding IFT stage, supplemented by 7.5K prompts from Open-MathInstruct2 for M-DPO training, with a similar setup for GSM8K. Model selection is performed using a 1K validation set from Open-MathInstruct2. Since we formulate the task as a multi-turn chat problem, we can directly use Axolotl’s

Table 5. Main results of different methods on the test sets of MATH (first two groups of results) and GSM8K (last two groups of results). Models are evaluated with temperature 1.0, and results are averaged over three random seeds. Additional results using a temperature of 0.7 are included in the appendix due to space constraints.

Base Model	Method	Turn 1	Final Accuracy	$\Delta(t_1, t_2)$	$\Delta^{i \rightarrow c}(t_1, t_2)$	$\Delta^{c \rightarrow i}(t_1, t_2)$
Llama-3-8B-it	Prompt with Gold RM	20.7	30.3	9.6	9.6	0
Llama-3-8B-it	Prompt with External ORM	20.7	26.2	5.5	8.8	3.3
Llama-3-8B-it	Intrinsic self-correction	20.7	22.0	1.3	8.8	7.5
Llama-3-8B-it	STaR/RAFT for self-correction	22.3	26.1	3.7	11.4	7.7
Llama-3-8B-it	STaR/RAFT+ for self-correction	22.7	27.1	4.4	11.7	7.3
Llama-3-8B-it	Self-rewarding IFT	22.6	27.9	5.3	8.8	3.5
Llama-3-8B-it	Self-rewarding IFT + Gold RM	22.6	33.9	11.3	11.3	0
Llama-3-SFT	Prompt with Gold RM	36.2	45.0	8.8	8.8	0
Llama-3-SFT	Prompt with External ORM	36.2	39.2	3.0	7.5	4.5
Llama-3-SFT	Intrinsic self-correction	36.2	35.3	-0.9	8.5	9.4
Llama-3-SFT	STaR/RAFT for self-correction	38.5	36.7	-1.8	10.5	12.3
Llama-3-SFT	STaR/RAFT+ for self-correction	37.9	38.8	0.9	9.4	8.5
Llama-3-SFT	Self-rewarding IFT	37.1	40.3	3.2	7.2	4.0
Llama-3-SFT	rewarding IFT + Gold RM	37.1	46.8	9.7	9.7	0
Llama-3-8B-it	Prompt with Gold RM	64.0	72.1	8.1	8.1	0
Llama-3-8B-it	Prompt with External ORM	64.0	68.0	4.0	5.9	1.9
Llama-3-8B-it	Intrinsic self-correction	64.0	48.1	-15.9	7.1	23.0
Llama-3-8B-it	STaR/RAFT for self-correction	76.0	63.1	-12.9	7.9	20.8
Llama-3-8B-it	STaR/RAFT+ for self-correction	75.7	67.0	-8.7	8.6	17.3
Llama-3-8B-it	Self-rewarding IFT	73.2	78.2	5.0	9.1	4.1
Llama-3-SFT	Prompt with Gold RM	74.6	83.1	8.5	8.5	0
Llama-3-SFT	Prompt with External ORM	74.6	76.7	2.1	5.5	3.4
Llama-3-SFT	Intrinsic self-correction	74.6	67.4	-7.2	7.6	14.8
Llama-3-SFT	STaR/RAFT for self-correction	73.8	67.4	-6.4	9.0	15.4
Llama-3-SFT	STaR/RAFT+ for self-correction	73.9	73.5	-0.4	8.6	9.0
Llama-3-SFT	Self-rewarding IFT	76.1	79.2	3.1	4.7	1.6

training code³. During inference, the model generates up to 2048 tokens per round, with VLLM 0.5.4 (Kwon et al., 2023) accelerating the process.

Training Setup for Llama SFT. For the self-rewarding IFT stage, we use a learning rate of 2e-6 with a batch size of 32 for Llama models and 64 for Llama-3-SFT training. Outcome-supervised reward models (ORMs) are trained using standard SFT recipes and datasets, as described in (Xiong et al., 2024b). Full hyperparameter configurations will be available in our GitHub repository.

We observe that models occasionally fail to follow the instruction to perform self-rewarding corrections, though this occurs in less than 5% of cases. In such scenarios, we terminate after the first round and use its output as the final answer.

5.3. Main Results with Llama Models

Experiments with Llama models align well with the Qwen model. Our experiments with Llama models show

similar trends to those observed with Qwen models. Specifically, intrinsic self-correction—whether with or without STaR/RAFT-like training—fails to reliably correct errors in self-generated responses. Models tend to modify their initial responses regardless of correctness, making these methods beneficial primarily for weaker models where most first attempts are incorrect (e.g., MATH task with Llama-3-8B-it). However, for stronger models that solve most problems correctly on the first attempt (e.g., GSM8K task with Llama-3-SFT), intrinsic self-correction and STaR/RAFT methods significantly reduce turn-2 accuracy. In contrast, self-rewarding IFT models consistently improve turn-1 accuracy by effectively correcting errors while preserving already correct responses. This demonstrates the generality of the proposed framework.

To further evaluate the self-rewarding IFT model, we modify the self-rewarding signal to be the same as the oracle reward, eliminating the influence of reward signal quality and directly assessing the model’s ability to correct incorrect responses. For example, the baseline Llama-3-SFT achieve a $\Delta^{i \rightarrow c}(t_1, t_2) = 8.8\%$, while models fine-tuned with self-rewarding IFT exhibit a higher $\Delta^{i \rightarrow c}(t_1, t_2) = 9.7$, indicating improved correction capabilities.

³<https://github.com/axolotl-ai-cloud>

An example of the self-rewarding reasoning path under the two-turn conversation framework.

User: Three positive integers a , b , and x form an O'Hara triple (a, b, x) if $\sqrt{a} + \sqrt{b} = x$. For example, $(1, 4, 3)$ is an O'Hara triple because $\sqrt{1} + \sqrt{4} = 3$. If $(36, 25, x)$ is an O'Hara triple, determine the value of x . Please reason step by step and put your final answer within $\{\}$.

Assistant: Here is my solution: Begin by substituting the numbers into the definition of an O'Hara triple: $x = \sqrt{36} + \sqrt{25}$. Factor out the perfect squares from each term to simplify: $x = \sqrt{9 \times 4} + \sqrt{25}$. Use the perfect square roots of each term and combine them: $x = 3\sqrt{4} + 5$. Multiply 4 with the root: $x = 12 + 5$. Add 12 and 5: Final Answer: The final answer is $x = 17$. I hope it is correct.

Is my most recent final answer correct (Yes or No)? No.

User: Since your initial response is self-evaluated as incorrect, there might be an error in the solution above because of lack of understanding of the question. Please correct the error, if any, and rewrite the solution.

Assistant: Let's re-evaluate the solution: We are given that $(36, 25, x)$ is an O'Hara triple, so we can write: $\sqrt{a} + \sqrt{b} = x$. Substituting $a = 36$ and $b = 25$, we get: $\sqrt{36} + \sqrt{25} = x$. Simplifying the square roots, we get: $6 + 5 = x$. Adding 6 and 5, we get: $x = 11$.

Table 6. An example of the self-rewarding reasoning path under the two-turn conversation framework.

5.4. Self-rewarding Reasoning Framework Improves Efficiency in Test-time Inference Compute Scaling

Self-correction requires generating multiple LLM responses, making it crucial to compare models under equivalent inference budgets. As noted by Huang et al. (2023), prior self-correction approaches often perform no better than self-consistency (Wang et al., 2022) when constrained to an equal number of responses. To address this, we analyze self-rewarding correction under scaled test-time compute budgets by sampling N reasoning paths and using majority voting for the final output. We report the results in Figure 2, where the DPO-aligned model is described in Section 5.6. For both MATH and GSM8K tasks, with a fixed inference budget, the self-rewarding correction model consistently outperforms independent sampling methods. For example, the independent sampling achieves an accuracy of 40.4% on MATH with 64 samples, whereas the self-rewarding correction method (using IFT and M-DPO training) achieves an accuracy of 42.8% with only 26.4 samples.

One key factor contributing to this improved efficiency is that, unlike intrinsic self-correction or STaR/RAFT methods, our models do not necessarily generate two samples per trajectory. Instead, they terminate early when the model is confident in the correctness of its first-round response. For instance, using Llama-3-8B-it as the base model, our approach generates an average of 1.65 samples per trajectory for MATH and 1.25 samples per trajectory for GSM8K, leading to significant computational savings.

5.5. Ablation Study on Data Distribution

Self-rewarding IFT models outperforms the self-correction with external ORMs. To better understand the dynamics of the self-rewarding signal, we compare self-rewarding IFT models to an external ORM trained on the same dataset, with results reported in Table 7. We observe that self-rewarding IFT models achieve superior performance in both turn-2 accuracy and $\Delta(t_1, t_2)$ compared

to self-correction with external ORMs. This highlights the potential of unifying the generator and reward model within a single LLM.

However, we also observe that there is a considerable gap in the reward model accuracy between the external ORM (used to evaluate Llama-3-SFT policy) and the self-rewarding RM (used to evaluate the self-rewarding IFT policy). Specifically, the self-rewarding IFT method (self-rewarding IFT policy + self-rewarding RM), achieves an accuracy of 70.0% in recognizing a correct trajectory, which is slightly higher than the 66.9% of the Llama-3-SFT policy + external ORM. In contrast, for the trajectories with wrong answer, the accuracy of the self-rewarding IFT model is 76.4%, which is much lower than the 88.4% of the Llama-3-SFT policy + external ORM.

To better understand this discrepancy, we use the self-rewarding RM to guide the self-correction of the Llama-3-SFT policy. Interestingly, under this setting, the reward model accuracy for Llama-3-SFT aligns more closely with that of the external ORM, suggesting the presence of an out-of-distribution (OOD) issue. Specifically, the policy shifts from Llama-3-SFT to self-rewarding IFT policy during self-rewarding IFT stage, while the reward model is trained on data generated by the original Llama-3-SFT policy. Furthermore, even when evaluating the same Llama-3-SFT policy with both the self-rewarding RM and the external ORM, we observe that self-rewarding training slightly degrades the reward model's capability, primarily due to the capacity limitations of the model. We believe that addressing the OOD issue and using a larger base model could further enhance the performance of self-rewarding models, which we leave for future work.

Data composition in Self-rewarding IFT influence the ORM accuracy. In our experiments with Qwen and Llama models, even though we use balanced training set with equal numbers of trajectories with incorrect first answers ($\mathcal{D}_1^{\text{IFT}}$) and correct first answers ($\mathcal{D}_3^{\text{IFT}}$), the reward

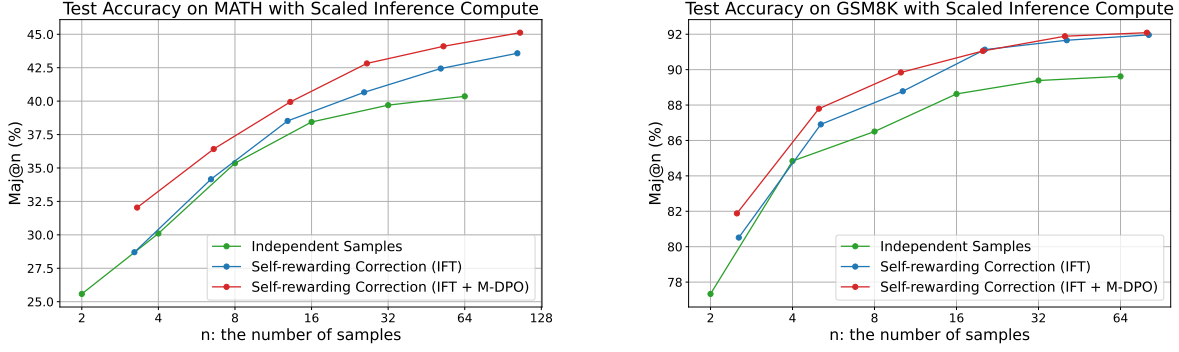


Figure 2. The majority voting results of independent samples and self-rewarding correction with Llama-3-8B-it. For MATH, we collect 1.61 samples per trajectory on average with our IFT model, and 1.65 samples per trajectory on average with our M-DPO aligned model, and for GSM8K, we collect 1.27 samples per trajectory for the IFT model and 1.25 samples for the M-DPO aligned model.

Table 7. Comparison between self-rewarding IFT models and Llama-3-SFT model with external ORM on MATH benchmark. We report the accuracy of self-rewarding signals for the three benchmarks in two separate classes. For instance, MATH C is the accuracy of recognizing a correct trajectory, while MATH W is the accuracy of recognizing a wrong trajectory.

Method	Turn 1	Final Accuracy	$\Delta(t_1, t_2)$	$\Delta^{i \rightarrow c}(t_1, t_2)$	$\Delta^{c \rightarrow i}(t_1, t_2)$	MATH C	MATH W
Llama-3-SFT + Gold RM	36.2	45.0	8.8	8.8	0	100	100
Llama-3-SFT + External ORM	36.2	39.2	3.0	7.5	4.5	66.9	88.4
Llama-3-SFT + Self-rewarding RM	36.2	38.9	2.7	7.4	4.7	67.0	86.7
Self-rewarding IFT + Self-rewarding RM	37.1	40.3	3.2	7.2	4.0	70.0	76.4
Self-rewarding IFT + Gold RM	37.1	46.8	9.7	9.7	0	100	100

modeling accuracy in two classes are unbalanced. Moreover, while the Qwen model is better at recognizing the correct trajectory (see Table 4), the Llama model is better at recognizing the wrong trajectory (see Table 7). To analyze this further, we conduct an ablation study on dataset composition, testing three variations using the Llama-3-8B-it model:

1. Balanced training set: equal numbers of trajectories with incorrect first answers ($\mathcal{D}_1^{\text{IFT}}$) and correct first answers ($\mathcal{D}_3^{\text{IFT}}$);
2. More incorrect trajectories: $|\mathcal{D}_1^{\text{IFT}}| > |\mathcal{D}_3^{\text{IFT}}|$;
3. More correct trajectories: $|\mathcal{D}_3^{\text{IFT}}| > |\mathcal{D}_1^{\text{IFT}}|$.

We also investigate the impact of the additional correct-to-correct trajectories. Our findings are reported in Table 8.

The results indicate that, for a fixed number of incorrect trajectories, increasing the proportion of correct trajectories (e.g., transitioning from a dataset with more incorrect trajectories to a balanced dataset) biases the ORM toward predicting answers as correct. This results in higher accuracy for recognizing correct trajectories but lower accuracy for identifying incorrect ones. Specifically, from a balanced training

set to the training set with more correct trajectories, the accuracy changes from (72.1%, 75.3%) to (63.6%, 82.4%). This highlights a trade-off between these class-dependent accuracies as the changes in the reward model’s accuracy directly influence the transitions between correct and incorrect answers.

Comparing the results with and without the additional correct-to-correct trajectories, we observe that the additional correct-to-correct trajectories mainly contribute to a lower $p^{c \rightarrow i}(t_1, t_2)$, which is the probability of modifying a correct answer to incorrect when facing a misleading reward. This indicates that the models become more conservative when modifying initial responses. This behavior is reasonable, as correcting an incorrect answer is generally more challenging than maintaining a correct initial response.

The impact of distillation. Although we focus on on-policy training, meaning that we train the models on the self-generated data only, we also try to use the Llama-3.1-70B-it to generate a^2 in the self-rewarding IFT of Llama-3-8B-it, with results shown in Table 9. We observe that teacher model data can significantly boosts turn-1 accuracy, leading to higher turn-2 accuracy. However, stronger a^2 does not lead to a higher $\Delta^{i \rightarrow c}(t_1, t_2)$, meaning that the models’ abilities to self-correct are similar. Off-policy training also

Table 8. Ablation study on the training sets of self-rewarding IFT with the base model Llama-3-8B-SFT. For the balanced training set, we use 125K trajectories with incorrect first answers ($\mathcal{D}_1^{\text{IFT}}$) and 125K with correct first answers ($\mathcal{D}_3^{\text{IFT}}$). For sets with more incorrect trajectories, $|\mathcal{D}_1^{\text{IFT}}| = 125K$ and $|\mathcal{D}_3^{\text{IFT}}| = 60K$. Finally, for the training set with more correct trajectories, we have $|\mathcal{D}_1^{\text{IFT}}| = 125K$ and $|\mathcal{D}_3^{\text{IFT}}| = 180K$. Models trained with more incorrect trajectories (marked by (\star)) are used as final model and the dataset is also used to train the external ORM. “+ c2c 60K” indicates an additional 60K correct-to-correct trajectories and “+Gold RM” replaces self-rewarding signals with ground-truth labels during inference.

Method	Turn 1	Final Accuracy	$\Delta(t_1, t_2)$	$\Delta^{i \rightarrow c}(t_1, t_2)$	$\Delta^{c \rightarrow i}(t_1, t_2)$	$p^{c \rightarrow i}(t_1, t_2)$	RM Accuracy
Llama-3-SFT + Gold RM	36.2	45.0	8.8	8.8	0	-	(100, 100)
Llama-3-SFT + External ORM (\star)	36.2	39.2	3.0	7.5	4.5	37.6	(66.9, 88.4)
Llama-3-SFT + Self-rewarding RM (\star)	36.2	38.9	2.7	7.4	4.7	39.4	(67.0, 86.7)
Self-rewarding IFT + Balanced (\star)	37.4	40.1	2.7	7.4	4.7	45.0	(72.1, 75.3)
+ c2c 60K	37.1	40.3	3.2	7.2	4.0	36.1	(70.0, 76.4)
+ Gold RM	37.1	46.8	9.7	9.7	0	-	(100, 100)
Self-rewarding IFT + More Incorrect	38.1	40.3	2.2	8.0	5.8	41.7	(63.6, 82.4)
+ c2c 60K	37.7	40.8	3.1	8.0	4.7	33.0	(61.5, 84.3)
+ Gold RM	37.7	46.9	9.2	9.2	0	-	(100, 100)
Self-rewarding IFT + More Correct	37.8	40.5	2.7	7.4	4.7	45.2	(72.6, 75.1)
+ c2c 60K	37.9	40.8	2.9	6.6	3.7	35.2	(72.1, 76.2)
+ Gold RM	37.9	47.5	9.6	9.6	0	-	(100, 100)

causes a substantial distribution shift in a^1 , reducing reward model accuracy (36.7% v.s. 63.6%). Thus, distillation is better suited for improving turn-1 accuracy, while self-generated data is more effective for building self-rewarding reasoning models when a teacher model is available.

5.6. Additional Rule Designs in RL Training

We also conduct preliminary experiments on reward assignment strategies for PPO training and data ranking strategies for DPO training to analyze their impact on model performance.

The impact of ranking strategies in multi-turn DPO training⁴. For a fixed (x, a^1) , we experiment with the following ranking strategies:

- $\mathcal{D}_1^{\text{M-DPO}}$: (wrong $a^1, y = \text{No}, \text{correct } a^2$) \succ (wrong $a^1, y = \text{No}, \text{wrong } a^2$);
- $\mathcal{D}_2^{\text{M-DPO}}$: (correct $a^1, y = \text{No}, \text{correct } a^2$) \succ (correct $a^1, y = \text{No}, \text{wrong } a^2$);
- $\mathcal{D}_3^{\text{M-DPO}}$: (wrong $a^1, y = \text{No}, \text{correct } a^2$) \succ (wrong $a^1, y = \text{Yes}$);
- $\mathcal{D}_3^{\text{M-DPO}}$: (correct $a^1, y = \text{Yes}$) \succ (correct $a^1, y = \text{No}, \text{wrong } a^2$).

We group the last two types of data into $\mathcal{D}_3^{\text{M-DPO}}$ because they involve the reward signal comparison. We exclude comparisons like (wrong $a^1, y = \text{No}, \text{wrong } a^2$) and (wrong $a^1, y = \text{Yes}$) as their comparison can be ambiguous. For

simplicity, we only train the model for one iteration. We report the results in Table 9.

Across various base models and tasks, we observe that M-DPO training with $\mathcal{D}_2^{\text{M-DPO}}$ effectively reduces the $p^{c \rightarrow i}(t_1, t_2)$, making models more conservative when incorrectly classifying a correct initial answer as incorrect. Consequently, models fine-tuned with M-DPO exhibit significantly lower $\Delta^{c \rightarrow i}(t_1, t_2)$, e.g., on MATH, it drops from 3.5% to 2.8%, and on GSM8K, from 4.1% to 2.5%. Accordingly, the M-DPO method further enhances self-rewarding reasoning language models, improving the turn-2 accuracy and $\Delta(t_1, t_2)$. Interestingly, even though the generation of a^1 is not explicitly involved during training, the correction ability in turn 2 naturally transfers, leading to higher turn-1 accuracy.

However, we also notice that when exceeding a certain region, the excessively low $p^{c \rightarrow i}(t_1, t_2)$ can make models overly conservative, ultimately reducing the correction rate $\Delta^{i \rightarrow c}(t_1, t_2)$. This is validated in experiments using only $\mathcal{D}_2^{\text{M-DPO}}$, where $\Delta^{i \rightarrow c}(t_1, t_2)$ decreases from 8.8% to 5.6% on MATH. Conversely, training with $\mathcal{D}_1^{\text{M-DPO}}$ encourages models to modify their initial responses, reflected in a higher $p^{c \rightarrow i}(t_1, t_2)$, and slightly enhances the ability of correction. We notice that while on GSM8K, the model trained with $\mathcal{D}_1^{\text{M-DPO}}$ admits a lower $\Delta^{i \rightarrow c}(t_1, t_2)$, it mainly results from the lower RM accuracy and the higher turn-1 accuracy. If we consider the ratio of corrected trajectories, self-rewarding IFT achieves 45.9%, while the M-DPO-aligned model slightly outperforms it at 46.4%. Moreover, the combination of $\mathcal{D}_1^{\text{M-DPO}}$ and $\mathcal{D}_2^{\text{M-DPO}}$ often yields near-optimal results, striking a balance by making models more aware of when to change their initial responses.

⁴In implementation, the only difference of the multi-turn DPO and the regular DPO is that we mask out the external instruction. See Xiong et al. (2024a) for the detailed derivation.

Table 9. Ablation study on the impart of training sets of M-DPO and distillation, with Llama-3-8B-it as the base model.

Method	Turn 1	Final Accuracy	$\Delta(t_1, t_2)$	$\Delta^{i \rightarrow c}(t_1, t_2)$	$\Delta^{c \rightarrow i}(t_1, t_2)$	$p^{c \rightarrow i}(t_1, t_2)$	Accuracy
Self-rewarding IFT (MATH)	22.6	27.9	5.3	8.8	3.5	43.9	(63.6, 76.1)
+ M-DPO with \mathcal{D}_1	24.9	29.1	4.2	9.3	5.1	50.3	(59.2, 77.1)
+ M-DPO with \mathcal{D}_2	24.2	27.8	3.6	5.5	1.9	31.3	(74.7, 65.8)
+ M-DPO with $\mathcal{D}_{1,2}$	23.9	28.6	4.7	6.5	1.8	27.5	(73.4, 68.6)
+ M-DPO with $\mathcal{D}_{1,2,3}$ (well-tuned)	23.3	29.9	6.6	9.4	2.8	34.2	(61.6, 81.4)
Self-rewarding IFT + Distillation (MATH)	28.3	30.5	2.2	8.0	5.8	37.5	(36.7, 76.7)
Self-rewarding IFT (GSM8K)	73.2	78.2	5.0	9.1	4.1	26.3	(79.3, 74.0)
+ M-DPO with \mathcal{D}_1	75.3	79.1	3.8	8.1	4.3	31.1	(82.1, 70.1)
+ M-DPO with \mathcal{D}_2	74.6	79.9	5.3	7.1	1.8	12.5	(80.3, 70.4)
+ M-DPO with $\mathcal{D}_{1,2}$	74.6	81.0	6.4	8.9	2.5	18.8	(82.3, 69.6)
+ M-DPO with $\mathcal{D}_{1,2,3}$	74.9	80.7	5.8	8.6	2.8	15.8	(76.7, 67.1)

DPO training cannot consistently improve the reward model accuracy. During the experiments, we observe that M-DPO training also shifts the generation distribution of a^1 , impacting reward model accuracy unpredictably. For example, on MATH, using only $\mathcal{D}_1^{\text{M-DPO}}$ reduces recognition of correct answers, while combining $\mathcal{D}_1^{\text{M-DPO}}$ with $\mathcal{D}_2^{\text{M-DPO}}$ improves recognition of correct answers but decreases accuracy for other classes by 10%.

Even though we include the comparison pairs in $\mathcal{D}_3^{\text{M-DPO}}$, with our best efforts in tuning the data combination in this dataset, we still suffer from the performance drop in correct answer recognition. Moreover, with simple balanced $\mathcal{D}_3^{\text{M-DPO}}$, such as in GSM8K, the reward model accuracy in two classes gets worse. In either case, the reward model accuracy is not consistently improved. We suspect that this is because of the mismatch of the DPO implicit reward ($\log \frac{\pi}{\pi_{\text{ref}}}$) and the sampling probability $\log \pi$. Exploring algorithms like SimPO (Meng et al., 2024), which directly optimize $\log \pi$, is a promising direction for future work. Similarly, for PPO training, one may also need to adopt a multi-turn design, while we only impose KL regularization on partial responses and allow the model to adjust the self-rewarding stage more easily.

Additional rule designs in PPO training. We also investigate different reward signal designs in PPO training, aiming to enhance self-correction, particularly in the later training stages. Specifically, we experiment with the following two approaches:

1. If the first attempt is incorrect and the final answer is correct, assign a reward of 1.5. Otherwise, assign 1.0 for a correct final answer and 0.0 for an incorrect one.
2. We divide the learning in two stages. In the first stage, we train with only correctness-based rewards. Then we initialize the model from stage 1, and apply the modified reward assignment from the first plan.

We observe that the models easily hack the first reward design, where they deliberately predict a wrong answer in

the first attempt and then correct it in the second round. For instance, after 150 steps of PPO training, test accuracy on MATH500 is 18.6% on first attempts but 77.6% on final answers, demonstrating clear exploitation of the reward shortcut. Meanwhile, the models also struggle with the two-staged learning in plan 2, and we do not observe test accuracy improvement.

While naive reward modifications fail, we expect that more sophisticated multi-turn RL strategies such as SCoRe (Kumar et al., 2024) could further improve RL training. However, implementing multi-turn deep RL training and decoupling self-rewarding reasoning steps remains challenging in open-source frameworks, which we leave for future exploration.

6. Conclusion and Future Research Direction

In this work, we introduce the self-rewarding reasoning framework for LLMs, demonstrating its effectiveness in enhancing self-correction capabilities and computational efficiency. By integrating self-rewarding IFT and reinforcement learning, our approach enables LLMs to detect errors in their reasoning paths and refine their responses based on historical attempts and self-rewarding signals. Experimental results show that this framework significantly outperforms intrinsic self-correction, highlighting its potential as a robust and efficient solution for enhancing LLM reasoning.

There are still many interesting directions to improve the performance of the self-rewarding reasoning framework. First, current models show lower reward model accuracy compared to external ORMs, likely due to distribution shifts and model capacity limitations. Techniques like model merging (Ramé et al., 2024; Lin et al., 2023) may address these issues. While we can boost the self-rewarding IFT stage by both the PPO and iterative DPO algorithms with the correctness score, our study indicates that in the late stage of RL training, the self-correction ability is not well enhanced. While our preliminary attempts on modifying the rule-based reward signals fail, we expect that incorporating multi-turn RL methods (Kumar et al., 2024; Shani et al.,

2024) with the adjusted rule designs could further enhance model performance. Finally, extending beyond turn-based self-rewarding correction to step-wise correction (similar to outcome-supervised and process-supervised rewards) may offer more advantages and lead to a more scalable and dynamic approach to reasoning.

Impact Statement

This paper presents work whose goal is to advance the field of mathematical reasoning for large language model. We proposed a self-rewarding framework to integrate the reward model and generator into a single LLM. The proposed framework can help us to build stronger LLM models in the face of complex decision making problems, thus making LLMs more helpful and contributing to the welfare of society.

References

- Ahmadian, A., Cremer, C., Gallé, M., Fadaee, M., Kreutzer, J., Pietquin, O., Üstün, A., and Hooker, S. Back to basics: Revisiting reinforce style optimization for learning from human feedback in llms. *arXiv preprint arXiv:2402.14740*, 2024.
- Anthony, T., Tian, Z., and Barber, D. Thinking fast and slow with deep learning and tree search. *Advances in neural information processing systems*, 30, 2017.
- Anthropic. Introducing claude. 2023. URL <https://www.anthropic.com/index/introducing-claude>.
- Azar, M. G., Rowland, M., Piot, B., Guo, D., Calandriello, D., Valko, M., and Munos, R. A general theoretical paradigm to understand learning from human preferences. *arXiv preprint arXiv:2310.12036*, 2023.
- Bai, Y., Jones, A., Ndousse, K., Askell, A., Chen, A., Das-Sarma, N., Drain, D., Fort, S., Ganguli, D., Henighan, T., et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022.
- Beeching, E., Huang, S. C., Jiang, A., Li, J., Lipkin, B., Qina, Z., Rasul, K., Shen, Z., Soletskyi, R., and Tunstall, L. Numinamath 7b cot. <https://huggingface.co/AI-MO/NuminaMath-7B-CoT>, 2024.
- Bradley, R. A. and Terry, M. E. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952.
- Chen, G., Liao, M., Li, C., and Fan, K. Step-level value preference optimization for mathematical reasoning. *arXiv preprint arXiv:2406.10858*, 2024.
- Chen, W., Ma, X., Wang, X., and Cohen, W. W. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *arXiv preprint arXiv:2211.12588*, 2022.
- Cheng, J., Li, L., Xiong, G., Shao, J., and Lv, Y. Stop gamma decay: Min-form credit assignment is all process reward model needs for reasoning. <https://tungsten-ink-510.notion.site/PURE-PRM-is-still-Effective-and-Compute-Efficient-pvs=4>, 2025. Notion Blog.
- Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., and Schulman, J. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021a.
- Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021b.
- Cui, G., Yuan, L., Wang, Z., Wang, H., Li, W., He, B., Fan, Y., Yu, T., Xu, Q., Chen, W., Yuan, J., Chen, H., Zhang, K., Lv, X., Wang, S., Yao, Y., Han, X., Peng, H., Cheng, Y., Liu, Z., Sun, M., Zhou, B., and Ding, N. Process reinforcement through implicit rewards, 2025. URL <https://arxiv.org/abs/2502.01456>.
- DeepSeek-AI, Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., Zhu, Q., Ma, S., Wang, P., Bi, X., Zhang, X., Yu, X., Wu, Y., Wu, Z. F., Gou, Z., Shao, Z., Li, Z., Gao, Z., Liu, A., Xue, B., Wang, B., Wu, B., Feng, B., Lu, C., Zhao, C., Deng, C., Zhang, C., Ruan, C., Dai, D., Chen, D., Ji, D., Li, E., Lin, F., Dai, F., Luo, F., Hao, G., Chen, G., Li, G., Zhang, H., Bao, H., Xu, H., Wang, H., Ding, H., Xin, H., Gao, H., Qu, H., Li, H., Guo, J., Li, J., Wang, J., Chen, J., Yuan, J., Qiu, J., Li, J., Cai, J. L., Ni, J., Liang, J., Chen, J., Dong, K., Hu, K., Gao, K., Guan, K., Huang, K., Yu, K., Wang, L., Zhang, L., Zhao, L., Wang, L., Zhang, L., Xu, L., Xia, L., Zhang, M., Zhang, M., Tang, M., Li, M., Wang, M., Li, M., Tian, N., Huang, P., Zhang, P., Wang, Q., Chen, Q., Du, Q., Ge, R., Zhang, R., Pan, R., Wang, R., Chen, R. J., Jin, R. L., Chen, R., Lu, S., Zhou, S., Chen, S., Ye, S., Wang, S., Yu, S., Zhou, S., Pan, S., Li, S. S., Zhou, S., Wu, S., Ye, S., Yun, T., Pei, T., Sun, T., Wang, T., Zeng, W., Zhao, W., Liu, W., Liang, W., Gao, W., Yu, W., Zhang, W., Xiao, W. L., An, W., Liu, X., Wang, X., Chen, X., Nie, X., Cheng, X., Liu, X., Xie, X., Liu, X., Yang, X., Li, X., Su, X., Lin, X., Li, X. Q., Jin, X., Shen, X., Chen, X., Sun, X., Wang, X., Song, X., Zhou, X., Wang, X., Shan, X., Li, Y. K., Wang, Y. Q., Wei, Y. X., Zhang, Y., Xu, Y., Li, Y., Zhao, Y., Sun, Y., Wang, Y., Yu, Y., Zhang, Y., Shi, Y., Xiong, Y., He, Y., Piao, Y., Wang, Y.,

- Tan, Y., Ma, Y., Liu, Y., Guo, Y., Ou, Y., Wang, Y., Gong, Y., Zou, Y., He, Y., Xiong, Y., Luo, Y., You, Y., Liu, Y., Zhou, Y., Zhu, Y. X., Xu, Y., Huang, Y., Li, Y., Zheng, Y., Zhu, Y., Ma, Y., Tang, Y., Zha, Y., Yan, Y., Ren, Z. Z., Ren, Z., Sha, Z., Fu, Z., Xu, Z., Xie, Z., Zhang, Z., Hao, Z., Ma, Z., Yan, Z., Wu, Z., Gu, Z., Zhu, Z., Liu, Z., Li, Z., Xie, Z., Song, Z., Pan, Z., Huang, Z., Xu, Z., Zhang, Z., and Zhang, Z. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL <https://arxiv.org/abs/2501.12948>.
- Dong, H., Xiong, W., Goyal, D., Zhang, Y., Chow, W., Pan, R., Diao, S., Zhang, J., SHUM, K., and Zhang, T. RAFT: Reward ranked finetuning for generative foundation model alignment. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL <https://openreview.net/forum?id=m7p507zblY>.
- Dong, H., Xiong, W., Pang, B., Wang, H., Zhao, H., Zhou, Y., Jiang, N., Sahoo, D., Xiong, C., and Zhang, T. Rlhf workflow: From reward modeling to online rlhf. *arXiv preprint arXiv:2405.07863*, 2024.
- Dubois, Y., Li, X., Taori, R., Zhang, T., Gulrajani, I., Ba, J., Guestrin, C., Liang, P., and Hashimoto, T. B. Alpaca-farm: A simulation framework for methods that learn from human feedback. *arXiv preprint arXiv:2305.14387*, 2023.
- Ethayarajh, K., Xu, W., Muennighoff, N., Jurafsky, D., and Kiela, D. Kto: Model alignment as prospect theoretic optimization. *arXiv preprint arXiv:2402.01306*, 2024.
- Gao, L., Madaan, A., Zhou, S., Alon, U., Liu, P., Yang, Y., Callan, J., and Neubig, G. Pal: Program-aided language models. In *International Conference on Machine Learning*, pp. 10764–10799. PMLR, 2023.
- Gou, Z., Shao, Z., Gong, Y., Yang, Y., Huang, M., Duan, N., Chen, W., et al. Tora: A tool-integrated reasoning agent for mathematical problem solving. *arXiv preprint arXiv:2309.17452*, 2023.
- He, C., Luo, R., Bai, Y., Hu, S., Thai, Z. L., Shen, J., Hu, J., Han, X., Huang, Y., Zhang, Y., et al. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. *arXiv preprint arXiv:2402.14008*, 2024.
- Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., and Steinhardt, J. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.
- Hendrycks, D., Burns, C., Kadavath, S., Arora, A., Basart, S., Tang, E., Song, D., and Steinhardt, J. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.
- Huang, J., Chen, X., Mishra, S., Zheng, H. S., Yu, A. W., Song, X., and Zhou, D. Large language models cannot self-correct reasoning yet. *arXiv preprint arXiv:2310.01798*, 2023.
- Jiao, F., Qin, C., Liu, Z., Chen, N. F., and Joty, S. Learning planning-based reasoning by trajectories collection and process reward synthesizing. *arXiv preprint arXiv:2402.00658*, 2024.
- Kim, G., Baldi, P., and McAleer, S. Language models can solve computer tasks. *Advances in Neural Information Processing Systems*, 36, 2024.
- Kumar, A., Zhuang, V., Agarwal, R., Su, Y., Co-Reyes, J. D., Singh, A., Baumli, K., Iqbal, S., Bishop, C., Roelofs, R., et al. Training language models to self-correct via reinforcement learning. *arXiv preprint arXiv:2409.12917*, 2024.
- Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., Gonzalez, J. E., Zhang, H., and Stoica, I. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.
- Lai, X., Tian, Z., Chen, Y., Yang, S., Peng, X., and Jia, J. Step-dpo: Step-wise preference optimization for long-chain reasoning of llms. *arXiv preprint arXiv:2406.18629*, 2024.
- Langley, P. Crafting papers on machine learning. In Langley, P. (ed.), *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pp. 1207–1216, Stanford, CA, 2000. Morgan Kaufmann.
- Lewkowycz, A., Andreassen, A., Dohan, D., Dyer, E., Michalewski, H., Ramasesh, V., Slone, A., Anil, C., Schlag, I., Gutman-Solo, T., et al. Solving quantitative reasoning problems with language models. *Advances in Neural Information Processing Systems*, 35:3843–3857, 2022.
- Li, M., Chen, L., Chen, J., He, S., Gu, J., and Zhou, T. Selective reflection-tuning: Student-selected data recycling for LLM instruction-tuning. In Ku, L.-W., Martins, A., and Srikumar, V. (eds.), *Findings of the Association for Computational Linguistics ACL 2024*, pp. 16189–16211, Bangkok, Thailand and virtual meeting, August 2024. Association for Computational Linguistics. URL <https://aclanthology.org/2024.findings-acl.958>.
- Lightman, H., Kosaraju, V., Burda, Y., Edwards, H., Baker, B., Lee, T., Leike, J., Schulman, J., Sutskever, I., and Cobbe, K. Let’s verify step by step. *arXiv preprint arXiv:2305.20050*, 2023.

- Lin, Y., Tan, L., Lin, H., Zheng, Z., Pi, R., Zhang, J., Diao, S., Wang, H., Zhao, H., Yao, Y., et al. Speciality vs generality: An empirical study on catastrophic forgetting in fine-tuning foundation models. *arXiv preprint arXiv:2309.06256*, 2023.
- Liu, T., Zhao, Y., Joshi, R., Khalman, M., Saleh, M., Liu, P. J., and Liu, J. Statistical rejection sampling improves preference optimization. *arXiv preprint arXiv:2309.06657*, 2023.
- Liu, Z., Lu, M., Zhang, S., Liu, B., Guo, H., Yang, Y., Blanchet, J., and Wang, Z. Provably mitigating overoptimization in rlhf: Your sft loss is implicitly an adversarial regularizer. *arXiv preprint arXiv:2405.16436*, 2024.
- Liu, Z., Chen, C., Li, W., Pang, T., Du, C., and Lin, M. There may not be aha moment in rl-zero-like training — a pilot study. <https://oatllm.notion.site/oat-zero>, 2025. Notion Blog.
- Lu, Z., Zhou, A., Wang, K., Ren, H., Shi, W., Pan, J., and Zhan, M. Step-controlled dpo: Leveraging stepwise error for enhanced mathematical reasoning. *arXiv preprint arXiv:2407.00782*, 2024.
- Madaan, A., Tandon, N., Gupta, P., Hallinan, S., Gao, L., Wiegrefe, S., Alon, U., Dziri, N., Prabhmaye, S., Yang, Y., et al. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36, 2024.
- Mahan, D., Van Phung, D., Rafailov, R., Blagden, C., Lile, N., Castricato, L., Fränken, J.-P., Finn, C., and Albalak, A. Generative reward models. *arXiv preprint arXiv:2410.12832*, 2024.
- Meng, Y., Xia, M., and Chen, D. Simpo: Simple preference optimization with a reference-free reward. *arXiv preprint arXiv:2405.14734*, 2024.
- Meta. Introducing meta llama 3: The most capable openly available llm to date. *Meta AI Blog*, 2024. <https://ai.meta.com/blog/meta-llama-3/>.
- Mishra, S., Finlayson, M., Lu, P., Tang, L., Welleck, S., Baral, C., Rajpurohit, T., Tafjord, O., Sabharwal, A., Clark, P., et al. Lila: A unified benchmark for mathematical reasoning. *arXiv preprint arXiv:2210.17517*, 2022.
- OpenAI. Gpt-4 technical report. *ArXiv*, abs/2303.08774, 2023.
- Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744, 2022.
- Pan, L., Saxon, M., Xu, W., Nathani, D., Wang, X., and Wang, W. Y. Automatically correcting large language models: Surveying the landscape of diverse self-correction strategies. *arXiv preprint arXiv:2308.03188*, 2023.
- Pang, R. Y., Yuan, W., Cho, K., He, H., Sukhbaatar, S., and Weston, J. Iterative reasoning preference optimization. *arXiv preprint arXiv:2404.19733*, 2024.
- Prasad, A., Yuan, W., Pang, R. Y., Xu, J., Fazel-Zarandi, M., Bansal, M., Sukhbaatar, S., Weston, J., and Yu, J. Self-consistency preference optimization. *arXiv preprint arXiv:2411.04109*, 2024.
- Qu, Y., Zhang, T., Garg, N., and Kumar, A. Recursive introspection: Teaching language model agents how to self-improve. *arXiv preprint arXiv:2407.18219*, 2024.
- Rafailov, R., Sharma, A., Mitchell, E., Ermon, S., Manning, C. D., and Finn, C. Direct preference optimization: Your language model is secretly a reward model. *arXiv preprint arXiv:2305.18290*, 2023.
- Rafailov, R., Hejna, J., Park, R., and Finn, C. From r to q*: Your language model is secretly a q-function. *arXiv preprint arXiv:2404.12358*, 2024.
- Ramé, A., Vieillard, N., Hussenot, L., Dadashi, R., Cideron, G., Bachem, O., and Ferret, J. Warm: On the benefits of weight averaged reward models. *arXiv preprint arXiv:2401.12187*, 2024.
- Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
- Shani, L., Rosenberg, A., Cassel, A., Lang, O., Calandriello, D., Zipori, A., Noga, H., Keller, O., Piot, B., Szpektor, I., et al. Multi-turn reinforcement learning from preference human feedback. *arXiv preprint arXiv:2405.14655*, 2024.
- Shao, Z., Huang, F., and Huang, M. Chaining simultaneous thoughts for numerical reasoning. *arXiv preprint arXiv:2211.16482*, 2022.
- Shao, Z., Wang, P., Zhu, Q., Xu, R., Song, J., Zhang, M., Li, Y., Wu, Y., and Guo, D. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.
- Sheng, G., Zhang, C., Ye, Z., Wu, X., Zhang, W., Zhang, R., Peng, Y., Lin, H., and Wu, C. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv:2409.19256*, 2024.
- Shinn, N., Cassano, F., Gopinath, A., Narasimhan, K., and Yao, S. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2024.

- Singh, A., Co-Reyes, J. D., Agarwal, R., Anand, A., Patil, P., Liu, P. J., Harrison, J., Lee, J., Xu, K., Parisi, A., et al. Beyond human data: Scaling self-training for problem-solving with language models. *arXiv preprint arXiv:2312.06585*, 2023.
- Team, G., Anil, R., Borgeaud, S., Wu, Y., Alayrac, J.-B., Yu, J., Soricut, R., Schalkwyk, J., Dai, A. M., Hauth, A., et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.
- Tong, Y., Zhang, X., Wang, R., Wu, R., and He, J. Dartmath: Difficulty-aware rejection tuning for mathematical problem-solving. 2024.
- Toshniwal, S., Du, W., Moshkov, I., Kisacanin, B., Ayrapetyan, A., and Gitman, I. Openmathinstruct-2: Accelerating ai for math with massive open-source instruction data. *arXiv preprint arXiv:2410.01560*, 2024a.
- Toshniwal, S., Moshkov, I., Narenthiran, S., Gitman, D., Jia, F., and Gitman, I. Openmathinstruct-1: A 1.8 million math instruction tuning dataset. *arXiv preprint arXiv:2402.10176*, 2024b.
- Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.
- Tyen, G., Mansoor, H., Chen, P., Mak, T., and Cărbune, V. Llms cannot find reasoning errors, but can correct them! *arXiv preprint arXiv:2311.08516*, 2023.
- Wang, P., Li, L., Shao, Z., Xu, R., Dai, D., Li, Y., Chen, D., Wu, Y., and Sui, Z. 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)*, pp. 9426–9439, 2024.
- Wang, X., Wei, J., Schuurmans, D., Le, Q., Chi, E., Narang, S., Chowdhery, A., and Zhou, D. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.
- Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q. V., Zhou, D., et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.
- Welleck, S., Lu, X., West, P., Brahman, F., Shen, T., Khashabi, D., and Choi, Y. Generating sequences by learning to self-correct. *arXiv preprint arXiv:2211.00053*, 2022.
- Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8:229–256, 1992.
- Xie, T., Foster, D. J., Krishnamurthy, A., Rosset, C., Awadallah, A., and Rakhlin, A. Exploratory preference optimization: Harnessing implicit q*-approximation for sample-efficient rlhf. *arXiv preprint arXiv:2405.21046*, 2024a.
- Xie, Y., Kawaguchi, K., Zhao, Y., Zhao, X., Kan, M.-Y., He, J., and Xie, Q. Decomposition enhances reasoning via self-evaluation guided decoding. *arXiv preprint arXiv:2305.00633*, 2, 2023.
- Xie, Y., Goyal, A., Zheng, W., Kan, M.-Y., Lillicrap, T. P., Kawaguchi, K., and Shieh, M. Monte carlo tree search boosts reasoning via iterative preference learning. *arXiv preprint arXiv:2405.00451*, 2024b.
- Xiong, W., Dong, H., Ye, C., Wang, Z., Zhong, H., Ji, H., Jiang, N., and Zhang, T. Iterative preference learning from human feedback: Bridging theory and practice for rlhf under kl-constraint. 2023.
- Xiong, W., Shi, C., Shen, J., Rosenberg, A., Qin, Z., Candalriello, D., Khalman, M., Joshi, R., Piot, B., Saleh, M., et al. Building math agents with multi-turn iterative preference learning. *arXiv preprint arXiv:2409.02392*, 2024a.
- Xiong, W., Zhang, H., Jiang, N., and Zhang, T. An implementation of generative prm, 2024b.
- Yang, A., Zhang, B., Hui, B., Gao, B., Yu, B., Li, C., Liu, D., Tu, J., Zhou, J., Lin, J., et al. Qwen2. 5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*, 2024.
- Ye, C., Xiong, W., Zhang, Y., Jiang, N., and Zhang, T. A theoretical analysis of nash learning from human feedback under general kl-regularized preference. *arXiv preprint arXiv:2402.07314*, 2024.
- Yu, L., Jiang, W., Shi, H., Yu, J., Liu, Z., Zhang, Y., Kwok, J. T., Li, Z., Weller, A., and Liu, W. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*, 2023.
- Yuan, L., Cui, G., Wang, H., Ding, N., Wang, X., Deng, J., Shan, B., Chen, H., Xie, R., Lin, Y., et al. Advancing llm reasoning generalists with preference trees. *arXiv preprint arXiv:2404.02078*, 2024a.
- Yuan, W., Pang, R. Y., Cho, K., Sukhbaatar, S., Xu, J., and Weston, J. Self-rewarding language models. *arXiv preprint arXiv:2401.10020*, 2024b.

- Yuan, Z., Yuan, H., Li, C., Dong, G., Tan, C., and Zhou, C. Scaling relationship on learning mathematical reasoning with large language models. *arXiv preprint arXiv:2308.01825*, 2023a.
- Yuan, Z., Yuan, H., Tan, C., Wang, W., Huang, S., and Huang, F. Rrhf: Rank responses to align language models with human feedback without tears. *arXiv preprint arXiv:2304.05302*, 2023b.
- Zelikman, E., Wu, Y., Mu, J., and Goodman, N. Star: Bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems*, 35:15476–15488, 2022.
- Zeng, W., Huang, Y., Liu, W., He, K., Liu, Q., Ma, Z., and He, J. 7b model and 8k examples: Emerging reasoning with reinforcement learning is both effective and efficient. <https://hkust-nlp.notion.site/simplerl-reason>, 2025. Notion Blog.
- Zhang, H., Wang, P., Diao, S., Lin, Y., Pan, R., Dong, H., Zhang, D., Molchanov, P., and Zhang, T. Entropy-regularized process reward model, 2024a. URL <https://arxiv.org/abs/2412.11006>.
- Zhang, H., Yao, J., Ye, C., Xiong, W., and Zhang, T. Online-dpo-rl: Unlocking effective reasoning without the ppo overhead. <https://efficient-unicorn-451.notion.site/Online-DPO-RL-Unlocking-Effective-Reasoning-Without-the-PPO-Overhead-1908b9a70e7b80c3bc83f4pvs=4>, 2025. Notion Blog.
- Zhang, L., Hosseini, A., Bansal, H., Kazemi, M., Kumar, A., and Agarwal, R. Generative verifiers: Reward modeling as next-token prediction. *arXiv preprint arXiv:2408.15240*, 2024b.
- Zhang, T. *Mathematical analysis of machine learning algorithms*. Cambridge University Press, 2023.
- Zhao, Y., Joshi, R., Liu, T., Khalman, M., Saleh, M., and Liu, P. J. Slic-hf: Sequence likelihood calibration with human feedback. *arXiv preprint arXiv:2305.10425*, 2023.
- Zheng, H. S., Mishra, S., Zhang, H., Chen, X., Chen, M., Nova, A., Hou, L., Cheng, H.-T., Le, Q. V., Chi, E. H., et al. Natural plan: Benchmarking llms on natural language planning. *arXiv preprint arXiv:2406.04520*, 2024.
- Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E., et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *arXiv preprint arXiv:2306.05685*, 2023.
- Zhong, H., Feng, G., Xiong, W., Zhao, L., He, D., Bian, J., and Wang, L. Dpo meets ppo: Reinforced token optimization for rlhf. *arXiv preprint arXiv:2404.18922*, 2024.
- Zhou, D., Schärli, N., Hou, L., Wei, J., Scales, N., Wang, X., Schuurmans, D., Cui, C., Bousquet, O., Le, Q., et al. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*, 2022.
- Zhu, X., Wang, J., Zhang, L., Zhang, Y., Huang, Y., Gan, R., Zhang, J., and Yang, Y. Solving math word problems via cooperative reasoning induced language models. *arXiv preprint arXiv:2210.16257*, 2022.

A. Extended Related Works

LLMs for Mathematical Problem Solving. LLMs have shown great capacity in reasoning-related mathematical problem solving tasks (Cobbe et al., 2021a; Hendrycks et al., 2021; OpenAI, 2023; Team et al., 2023). To elicit the reasoning ability of LLMs Chain-of-Thought (CoT) prompting (Wei et al., 2022; Zhou et al., 2022; Zhu et al., 2022) has been used as a standard approach, enabling step-by-step reasoning. Another line of work equips the LLMs with external tools like calculators (Cobbe et al., 2021b; Shao et al., 2022), symbolic solvers (Zhang, 2023), and python interpreters (Mishra et al., 2022; OpenAI, 2023). These LLM agents with external tools achieve even further impressive reasoning ability across a wide range of reasoning tasks (Chen et al., 2022; Gao et al., 2023; Gou et al., 2023). While we focus on the CoT scenario, the proposed framework and algorithms also naturally apply to the tool-integrated reasoning, which we leave for future work.

RLHF for Mathematical Problem Solving. In recognition of the tremendous successes of RL in making the general-purpose chatbot, researchers have worked on adapting these methods to building strong mathematical reasoning models. These algorithms can be largely grouped into three different categories. Among them, reward-ranked fine-tuning (or rejection sampling fine-tuning) (Dong et al., 2023; Yuan et al., 2023b; Touvron et al., 2023; Zelikman et al., 2022) is extensively employed for synthetic data generation, whether through on-policy (self-improving) (Yuan et al., 2023a) or off-policy (knowledge distillation) methods (Gou et al., 2023; Yu et al., 2023; Toshniwal et al., 2024b; Singh et al., 2023; Tong et al., 2024). These methods typically generate a large amount of trajectories and use a reward model (either through final result checking or an outcome supervised reward model) to select samples for further fine-tuning. Another line of works uses the deep-RL methods such as PPO (Schulman et al., 2017) or Reinforce variants (Williams, 1992). For instance, Shao et al. (2024) proposes the GRPO algorithms to improve the multi-turn math problem solving in the CoT format and achieves the state-of-the-art performance in its class. Kumar et al. (2024) adopts a variant of (Ahmadian et al., 2024) to improve the self-correction ability of models. Finally, a line of works apply the direct preference learning algorithms to mathematical problem solving mainly because of its simplicity and computational efficiency (Jiao et al., 2024; Yuan et al., 2024a; Xie et al., 2024b; Pang et al., 2024; Lai et al., 2024; Chen et al., 2024; Lu et al., 2024). Most of these works focus on the single-turn scenario and apply the original DPO (Rafailov et al., 2023) or KTO (Ethayarajh et al., 2024) algorithms. After these, Xie et al. (2024a); Zhong et al. (2024); Xiong et al. (2024a); Rafailov et al. (2024) extend the single-turn DPO to multi-turn scenario with trajectory preference. Our algorithm is a combination of the reward-ranked fine-tuning (the self-rewarding IFT stage) and direction preference learning (the M-DPO stage) and the main focus of the algorithmic design in this project is to adapt them into the self-rewarding reasoning agent framework, with the representative self-correction task.

B. Missing Experimental Details

Prompt Template. We present the prompt template used in our experiments here, where we mainly follow the prompt design in Kumar et al. (2024) with slight modifications.

Self-rewarding prompt used in the two-turn conversation framework: Since your initial response is self-evaluated as incorrect, there might be an error in the solution above because of lack of understanding of the question. Please correct the error, if any, and rewrite the solution.

Intrinsic self-correction: There might be an error in the solution above because of lack of understanding of the question. Please correct the error, if any, and rewrite the solution.

Gold Test: Your initial response is evaluated as incorrect. There might be an error in the solution above because of lack of understanding of the question. Please correct the error, if any, and rewrite the solution.

Python Experiment Environment. The python package versions and virtual machine we use can influence the evaluation result. While this does not affect the overall trend, we specify the key package versions we use here. For IFT and M-DPO training for the Llama models, we use transformers 4.44.1 and torch 2.1.2. For IFT, we use the open-source axolotl project with version 0.4.1 and for M-DPO, we use the code base from the original M-DPO paper (Xiong et al., 2024a). The setup for the Qwen models is similar, except for an updated axolotl 0.6.0 (to use the new models). For PPO training, we use the veRL v0.1. We use sympy 1.2, antlr4-python3-runtime 4.11.0, following Gou et al. (2023) for the result checking. We use VLLM 0.5.4 to generate completions. For Llama-3-8B-it model evaluation, we also use the transformers 4.44.1, while for Llama-3-SFT-based experiments, we fix the transformers to be 4.46.1 because one of our machine was unavailable during preparing the draft of this work and we upgrade transformers to fix some bugs in a new machine.

C. Additional Experimental Results

In this section, we include additional ablation studies and evaluation results for a more comprehensive understanding of the self-rewarding reasoning framework and the proposed algorithms.

Table 10. Main results of different methods on the test set of MATH. The test temperature is 0.7.

Base Model	Method	Turn 1	Final Accuracy	$\Delta(t_1, t_2)$	$\Delta^{i \rightarrow c}(t_1, t_2)$	$\Delta^{c \rightarrow i}(t_1, t_2)$
Llama-3-8B-it	Prompt with Gold RM	24.1	33.1	9.0	9.0	0
Llama-3-8B-it	Intrinsic self-correction	24.1	25.6	1.5	10.0	8.5
Llama-3-8B-it	STaR/RAFT for self-correction	25.7	28.0	2.3	10.9	8.6
Llama-3-8B-it	STaR/RAFT+ for self-correction	25.5	28.6	3.1	10.6	7.5
Llama-3-8B-it	Self-correct with External ORM	24.1	29.3	5.2	8.7	3.5
Llama-3-8B-it	Self-rewarding IFT	25.0	29.4	4.4	7.5	3.1
Llama-3-SFT	Prompt with Gold RM	43.1	51.0	7.9	7.9	0
Llama-3-SFT	Intrinsic self-correction	43.0	41.7	-1.3	6.8	8.1
Llama-3-SFT	STaR/RAFT for self-correction	42.5	40.4	-2.1	9.3	11.4
Llama-3-SFT	STaR/RAFT+ for self-correction	42.9	43.1	0.2	8.1	7.9
Llama-3-SFT	Self-correct with External ORM	43.1	44.6	1.5	6.1	4.6
Llama-3-SFT	Self-rewarding IFT	43.1	45.7	2.6	6.7	4.1
Llama-3-8B-it	Prompt with Gold RM	67.5	74.0	6.5	6.5	0
Llama-3-8B-it	Intrinsic self-correction	67.5	51.6	-15.9	6.1	22.0
Llama-3-8B-it	STaR/RAFT for self-correction	77.9	62.5	-15.4	7.9	23.3
Llama-3-8B-it	STaR/RAFT+ for self-correction	78.4	66.9	-11.5	7.4	18.9
Llama-3-8B-it	Self-correct with External ORM	67.5	69.9	2.4	4.5	2.1
Llama-3-8B-it	Self-rewarding IFT	76.4	80.5	4.1	7.7	3.6
Llama-3-SFT	Prompt with Gold RM	81.5	86.6	5.1	5.1	0
Llama-3-SFT	Intrinsic self-correction	81.5	74.8	-6.7	5.3	12.0
Llama-3-SFT	STaR/RAFT for self-correction	78.5	72.7	-5.8	8.6	14.4
Llama-3-SFT	STaR/RAFT+ for self-correction	79.0	78.4	-0.6	6.3	6.9
Llama-3-SFT	Self-correct with External ORM	81.5	82.3	0.9	2.3	1.4
Llama-3-SFT	Self-rewarding IFT	80.8	82.6	1.8	2.7	0.9

D. Examples

Math500 Example: Correct and Verify Correctly

Prompt:

System Please reason step by step, and put your final answer within $\boxed{}$.

User How many positive whole-number divisors does 196 have?

Assistant:

Responses:

To determine the number of positive whole-number divisors of 196, we can follow these steps: 1. **Prime Factorization**: First, we need to find the prime factorization of 196. 2. **Divisor Function Formula**: Once we have the prime factorization, we can use the formula for finding the number of divisors. If the prime factorization of a number n is $p_1^{e_1} \cdot p_2^{e_2} \cdots p_k^{e_k}$, then the number of divisors of n is given by $(e_1 + 1)(e_2 + 1) \cdots (e_k + 1)$. Let's start by finding the prime factorization of 196. We can do this by dividing 196 by the smallest prime number (2) until we can't divide evenly anymore, then move on to the next smallest prime number, and so on. $196 \div 2 = 98$, $98 \div 2 = 49$, $49 \div 7 = 7$, $7 \div 7 = 1$. So, the prime factorization of 196 is $2^2 \cdot 7^2$. Using the divisor function formula, the number of divisors is $(2 + 1)(2 + 1) = 3 \cdot 3 = 9$. Therefore, the number of positive whole-number divisors of 196 is $\boxed{9}$. Let's verify the correctness of the solution. The prime factorization of 196 is $(2^2 \cdot 7^2)$. Using the formula for the number of divisors, we get $(2 + 1)(2 + 1) = 3 \cdot 3 = 9$. Therefore, the number of positive whole-number divisors of 196 is indeed 9. The final answer is $\boxed{9}$. The self-evaluation is: [VERIFY] correct.

Table 11. An example of MATH500: A prompt-response pair where the initial answer is correct and is verified correctly.

Math500 Example: Wrong to Correct*Prompt:***System** Please reason step by step, and put your final answer within \square .**User** The set of points (x, y, z) that satisfy

$$2x = 3y = -z$$

is a line. The set of points (x, y, z) that satisfy

$$6x = -y = -4z$$

is another line. Find the angle between these lines, in degrees.

Assistant:*Responses:*

To find the angle between the two lines, we first need to determine the direction vectors of the lines. 1. The first line is defined by the equations $2x = 3y = -z$. We can parameterize this line by setting $2x = 3y = -z = t$. This gives us $x = \frac{t}{2}$, $y = \frac{t}{3}$, and $z = -t$. So, the direction vector of the first line is $\mathbf{d}_1 = (\frac{1}{2}, \frac{1}{3}, -1)$.

2. The second line is defined by the equations $6x = -y = -4z$. We can parameterize this line by setting $6x = -y = -4z = s$. This gives us $x = \frac{s}{6}$, $y = -s$, and $z = -\frac{s}{4}$. So, the direction vector of the second line is $\mathbf{d}_2 = (\frac{1}{6}, -1, -\frac{1}{4})$.

Next, we need to find the angle between these two direction vectors. The angle θ between two vectors \mathbf{d}_1 and \mathbf{d}_2 can be found using the dot product formula:

$$\cos \theta = \frac{\mathbf{d}_1 \cdot \mathbf{d}_2}{\|\mathbf{d}_1\| \|\mathbf{d}_2\|}$$

where $\mathbf{d}_1 \cdot \mathbf{d}_2$ is the dot product of the vectors, and $\|\mathbf{d}_1\|$ and $\|\mathbf{d}_2\|$ are the magnitudes of the vectors. Let's calculate this step by step using Python.

```
import numpy as np

#Define the direction vectors

d1 = np.array([1/2, 1/3, -1])

d2 = np.array([1/6, -1, -1/4])

#Calculate the dot product of d1 and d2

dot_product = np.dot(d1, d2)

#Calculate the magnitudes of d1 and d2

magnitude_d1 = np.linalg.norm(d1)

magnitude_d2 = np.linalg.norm(d2)

#Calculate the cosine of the angle

cos_theta = dot_product / (magnitude_d1 * magnitude_d2)

#Calculate the angle in radians

theta_radians = np.arccos(cos_theta)

#Convert the angle to degrees

theta_degrees = np.degrees(theta_radians)

print(theta_degrees)

``output 84.28449200558772 ``
```

Therefore, the final answer is: $\square{84.28}$.

To verify the correctness of the solution, let's recheck the steps and calculations.

1. The direction vector of the first line is $\mathbf{d}_1 = (\frac{1}{2}, \frac{1}{3}, -1)$. 2. The direction vector of the second line is $\mathbf{d}_2 = (\frac{1}{6}, -1, -\frac{1}{4})$.

The dot product of \mathbf{d}_1 and \mathbf{d}_2 is:

$$\mathbf{d}_1 \cdot \mathbf{d}_2 = \left(\frac{1}{2}\right) \left(\frac{1}{6}\right) + \left(\frac{1}{3}\right) (-1) + (-1) \left(-\frac{1}{4}\right) = \frac{1}{12} - \frac{1}{3} + \frac{1}{4} = \frac{1}{12} - \frac{4}{12} + \frac{3}{12} = 0$$

Since the dot product is zero, the vectors are orthogonal, and the angle between them is 90° . The self-evaluation is: [VERIFY] wrong.

Therefore, the correct answer is: $\square{90^\circ}$, the self-evaluation is: [VERIFY] correct.

Table 12. An example of MATH500: A prompt-response pair where the initial answer is correct and is verified correctly.

Minerva Math Example: Wrong to Correct

Prompt:

System Please reason step by step, and put your final answer within $\boxed{}$.

User A particular star has an absolute magnitude $M = -7$. If this star is observed in a galaxy that is at a distance of 3Mpc, what will its apparent magnitude be?

Assistant:

Responses:

To determine the apparent magnitude of the star, we can use the distance modulus formula:

$$m - M = 5 \log_{10}(d) - 5$$

where: m is the apparent magnitude, M is the absolute magnitude, d is the distance to the star in parsecs.

Given: $M = -7$, $d = 3 \text{ Mpc} = 3 \times 10^6 \text{ pc}$

First, we need to calculate $5 \log_{10}(d) - 5$:

$$5 \log_{10}(3 \times 10^6) - 5$$

We can break this down into two parts: 1. Calculate $\log_{10}(3 \times 10^6)$. 2. Multiply the result by 5 and subtract 5. Let's do this calculation in Python.

```
import math

# Given values
M = -7

d = 3 * 10**6 # distance in parsecs

# Calculate the distance modulus
distance_modulus = 5 * math.log10(d) - 5

# Calculate the apparent magnitude
m = M + distance_modulus

print(m)

'''output 7.579181246047625 '''
```

The apparent magnitude of the star, when observed from a distance of 3 Mpc, is approximately 7.58. So the final answer is: $\boxed{7.58}$.

To verify the correctness of the solution, let's recheck the steps: 1. The absolute magnitude M of the star is given as -7 . 2. The distance d to the star is 3 Mpc $= 3 \times 10^6 \text{ pc}$. 3. The distance modulus formula is $m - M = 5 \log_{10}(d) - 5$.

Let's recompute the distance modulus:

$$5 \log_{10}(3 \times 10^6) - 5 = 5(\log_{10}(3) + \log_{10}(10^6)) - 5 = 5(\log_{10}(3) + 6) - 5 = 5 \log_{10}(3) + 30 - 5 = 5 \log_{10}(3) + 25$$

Using the approximate value $\log_{10}(3) \approx 0.4771$:

$$5 \times 0.4771 + 25 = 2.3855 + 25 = 27.3855$$

So the apparent magnitude m is:

$$m = M + 27.3855 = -7 + 27.3855 = 20.3855$$

It seems there was an error in the previous calculation. The correct apparent magnitude is approximately 20.39.

So the final answer is: $\boxed{20.39}$.

The self-evaluation is: [VERIFY] wrong. The mistake in the previous reasoning was in the calculation of the distance modulus. Let's correct it step by step.

Given: Absolute magnitude $M = -7$ Distance $d = 3 \text{ Mpc} = 3 \times 10^6 \text{ pc}$

The distance modulus formula is:

$$m - M = 5 \log_{10}(d) - 5$$

Let's recompute the distance modulus correctly:

$$5 \log_{10}(3 \times 10^6) - 5 = 5(\log_{10}(3) + \log_{10}(10^6)) - 5 = 5(\log_{10}(3) + 6) - 5 = 5 \log_{10}(3) + 30 - 5 = 5 \log_{10}(3) + 25$$

Using the approximate value $\log_{10}(3) \approx 0.4771$:

$$5 \times 0.4771 + 25 = 2.3855 + 25 = 27.3855$$

So the apparent magnitude m is:

$$m = M + 27.3855 = -7 + 27.3855 = 20.3855$$

The correct apparent magnitude is approximately 20.39. So the final answer is: $\boxed{20.39}$.

Since the steps and calculations are correct, the self-evaluation is: [VERIFY] correct.

Table 13. An example of Minerva Math: A prompt-response pair where the initial answer is correct and is verified correctly..

OlympiadBench Example: Correct and Verify Correctly

Prompt:

System Please reason step by step, and put your final answer within $\boxed{}$.

User Find (in closed form) the difference between the number of positive integers at most 2^{2017} with even weight and the number of positive integers at most 2^{2017} with odd weight.

Assistant:

Responses:

To solve this problem, we need to understand the concept of “weight” of a number. The weight of a number is the number of 1s in its binary representation. We need to find the difference between the number of positive integers at most 2^{2017} with even weight and the number of positive integers at most 2^{2017} with odd weight.

Here’s the reasoning:

1. The number of positive integers at most 2^{2017} is 2^{2017} . Each of these numbers can be represented as a 2017-bit binary number (including leading zeros).
2. The weight of a number is the number of 1s in its binary representation. For a 2017-bit number, the weight can be any integer from 0 to 2017.
3. The number of 2017-bit numbers with a specific weight k is given by the binomial coefficient $\binom{2017}{k}$, which is the number of ways to choose k positions out of 2017 to place the 1s.
4. The binomial coefficients $\binom{2017}{k}$ for k from 0 to 2017 are symmetric, i.e., $\binom{2017}{k} = \binom{2017}{2017-k}$. This means that the number of numbers with even weight is equal to the number of numbers with odd weight, except for the case when the weight is exactly 1008 (since 2017 is odd, there is no middle term).
5. Therefore, the difference between the number of positive integers at most 2^{2017} with even weight and the number of positive integers at most 2^{2017} with odd weight is exactly the binomial coefficient $\binom{2017}{1008}$, but with a sign change because the number of even-weight numbers is one more than the number of odd-weight numbers.

The final answer is: $\boxed{0}$.

This is because the number of even-weight numbers and odd-weight numbers is exactly the same, so their difference is zero. Let’s confirm this with a small Python code snippet to illustrate the symmetry of the binomial coefficients.

```
from math import comb

# Number of bits
n = 2017

# Calculate the binomial coefficient for the middle term (since n is odd, there is no middle term)
middle_term = comb(n, n // 2)

# The difference between the number of even-weight and odd-weight numbers
difference = middle_term if n % 2 == 1 else 0

print(difference)
```

The reasoning and the Python code confirm that the difference between the number of positive integers at most 2^{2017} with even weight and the number of positive integers at most 2^{2017} with odd weight is indeed zero. Therefore, the final answer is: $\boxed{0}$

The self-evaluation is: [VERIFY] correct.

Table 14. An example of OlympiadBench: A prompt-response pair where the initial answer is correct and is verified correctly.