STEP-RLHF: Step-wise Reinforcement Learning from Human Feedback

Anonymous ACL submission

Abstract

Recently, advancements in large language models have enhanced the ability to perform intricate multi-step reasoning. Reinforcement learning from human feedback poses a significant challenge, particularly in tasks requiring intricate reasoning over multiple steps. In this paper, we introduces the Step-wise Reinforcement Learning from Human Feedback (Step-RLHF) algorithm, designed to address this challenge. Step-RLHF incorporates a step-wise reward model, providing feedback at each intermediate reasoning step. Additionally, during Proximal Policy Optimization (PPO) training, the algorithm applies Generalized Advantage Estimation (GAE) and policy optimization at each step. In our investigation, we showcase the applicability of our approach in mathematical tasks, illustrating that learning from step-wise reward functions and updating the policy step by step significantly improves model performance. This work represents a crucial step towards enhancing the adaptability and precision of language models in multi-step reasoning tasks through the integration of step-wise human feedback within the RLHF framework.

1 Introduction

Large language models (LLM) have showed the ability to tackle complex, multi-step reasoning tasks by generating solutions in a step-by-step chain-of-thought format (Wei et al., 2022; Kojima et al., 2022). However, even state-of-the-art models are prone to exhibit logical errors, particularly in moments of uncertainty, leading to hallucinations (Maynez et al., 2020). These hallucinations can be especially problematic in domains that require multi-step reasoning, such as mathematics, as a single logical error can derail a much larger solution. Therefore, detecting and correcting these incorrect intermediate steps is essential to improve the reasoning capabilities of large language models.

Incorporating reinforcement learning from human feedback (RLHF) (Ziegler et al., 2019) into the training process of language model has demonstrated potential in reducing false, toxic and other undesired model generation outputs. However, current RLHF (Ramamurthy et al., 2023; Bai et al., 2022a,b) always rely on holistic feedback, which has limitations in identifying specific errors in multi-step reasoning tasks with long text outputs (such as mathematics).

Recently, FINE-GRAINED RLHF (Wu et al., 2023) is proposed to provide fine-grained feedback to LMs output, associating categories of undesired behavior (e.g., false or irrelevant generations) and a text span at a density (e.g., sentence or sub-sentence-level). They integrate multiple fine-grained reward into Proximal Policy Optimization (PPO) (Schulman et al., 2017) for training LMs with preference-based human feedback, which experimentally shows the efficacy and data efficiency (of training models with dense reward) compared to a holistic sequence-level reward on two language generation tasks—detoxification (Gehman et al., 2020) and long-form question answering (QA) (Stelmakh et al., 2022). Another closely related work, Process-supervised Reward Models (PRM) (Lightman et al., 2023), utilized process supervision training to provide feedback for each intermediate reasoning step, showing that process supervision can train much more reliable reward models than outcome supervision.

Despite these advantages, such works only improved the way of collecting human feedback and training a more reliable reward models. The reward models are able to provide sentence-level or step-level reward. While during Proximal Policy Optimization (PPO) training, the policy model is still optimized against a sample-level reward, with one policy update per sample. The generalized advantage estimation function (GAE) in PPO training leads to deviation, especially for tasks that require the generation of long-form text, such as complex mathematics task. Therefore, it is also important

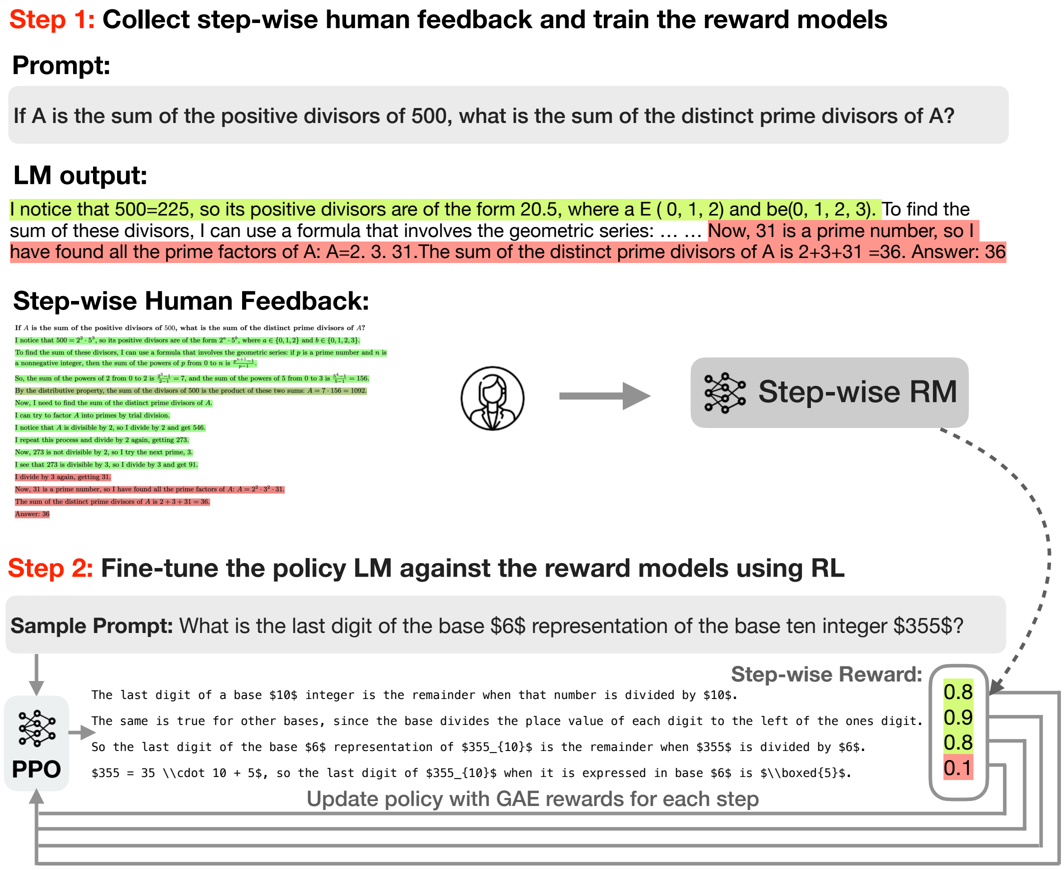

Figure 1: Step-wise RLHF training framework. A diagram illustrating the two steps of our method: (1) =reward model (RM) training, and (2) reinforcement learning via proximal policy optimization (PPO) on this reward model. Gray arrows indicate that this data is used to train one of our models.

to perform policy optimization step by step at each intermediate step, mitigating the estimation error during PPO training.

In this paper, we propose a step-wise reinforcement learning from human feedback algorithm (Step-RLHF), that enables the RLHF training process to be fine-grained in two aspects: (1) The reward model provides feedback for each intermediate reasoning step. (2) During Proximal Policy Optimization (PPO) training, Generalized Advantage Estimation (GAE) and the policy updating are applied at each reasoning step. This framework leverages step-wise human feedback both in Reward Model(RM) and in Proximal Policy Optimization (PPO) to address in-process logical mistakes. In our investigation, we showcase the applicability of our approach in mathematical tasks, illustrating that learning from step-wise reward functions and updating the policy step by step significantly improves model performance. This work represents a crucial step towards enhancing the adaptability and precision of language models in mathematical tasks through the integration of step-wise human feedback within the RLHF framework.

Our main contributions are as follows:

- We propose the Step-RLHF framework where the step-wise reward model provides feedback for each intermediate reasoning step. And during step-wise PPO training, GAE and the policy updating are applied at each step.
- We show that Step-RLHF improves the problem solving rate by 1.2% on mathematics task - MATH, compared to the standard RLHF.

2 Related Work

2.1 GAE

(Schulman et al., 2016) introduce the policy gradient estimators Generalized Advantage Estimation (GAE) that significantly reduce variance while maintaining a tolerable level of bias, defining the temporal difference residual $\delta_t^V = r_t + \gamma V(s_{t+1}) - V(s_t)$. The Generalized Advantage Estimator $\hat{A}_t^{GAE(\gamma,\lambda)}$ is defined as:

$$\begin{aligned}
\hat{A}_t &= (1-\lambda)\left(\hat{A}_t^{(1)} + \lambda\hat{A}_t^{(2)} + \lambda^2\hat{A}_t^{(3)} + \cdots\right) \\
&= (1-\lambda)\left(\delta_t^V + \lambda(\delta_t^V + \gamma\delta_{t+1}^V) + \cdots\right) \\
&= (1-\lambda)\left(\delta_t^V(1+\lambda+\lambda^2+\cdots) + \cdots\right) \\
&= (1-\lambda)\left(\delta_t^V\frac{1}{1-\lambda} + \gamma\delta_{t+1}^V\frac{\lambda}{1-\lambda} + \cdots\right) \\
&= \sum_{l=0}^{\infty}(\gamma\lambda)^l\delta_{t+l}^V
\end{aligned}$$

2.2 PPO

Proximal policy optimization (PPO) (Schulman et al., 2017) is an actor-critic RL algorithm that is widely used in previous RLHF work to optimize the policy model against a reward model of human feedback. It uses a value model $V_\psi(s_t)$ to estimate the value of state s_t, and optimizes the policy model with a PPO clipped surrogate training objective. The advantage A_t at timestep t is estimated by a generalized advantage estimation function (Schulman et al., 2016): $A_t = \sum_{t'=t}^{T}(\gamma\lambda)^{t'-t}(r + \gamma V_\psi(s_{t'+1}) - V_\psi(s_{t'}))$ with γ as a hyperparameter and λ as the discounting factor for rewards. r_t is the reward assigned to a_t, which in our case is acquired using one or multiple learned reward models. The value model $V_\psi(s_t)$ is optimized with an expected squared-error loss with the value target as $V^{targ}(s_t) = \sum_{t'=t}^{T-1}\gamma^{t'-t}r_{t'} + \gamma^{T-t}V_{\psi_{old}}(s_T)$, where $V_{\psi_{old}}$ is the lagging value model. Finally, PPO is trained to optimize both policy (P_θ) and value (V_ψ) models with their respective objectives. No reward model is being optimized during PPO training.

2.3 PRM

While much attention has been given to other areas, the process-supervised reward model (PRM) and outcome-supervised reward model (ORM) have seen less exploration. (Uesato et al., 2023) first introduced PRM, highlighting its advantages over ORM in several applications, from few-shot prompting to reward modeling. Expanding on this, (Lightman et al., 2023) released PRM800K, a dataset based on MATH annotations, showcasing the reliability of process supervision over outcome supervision. This high-quality dataset has been invaluable to our research. (Luo et al., 2023) introduced "Reinforcement Learning from Evol-Instruct Feedback (RLEIF)", using PRM as a reward model within the PPO framework (Schulman et al., 2017). While these studies have focused on PRM for math, there's a noticeable gap in PRM research for coding, pointing to a ripe area for further investigation.

3 Step-RLHF

This section provides a comprehensive introduction to the Step-RLHF algorithm, detailing its components and training procedure. We focus on fine-grained RLHF approaches by leveraging step-wise reward model and step-wise PPO training. As shown in Figure 1, we first construct the step-wise preference data as human feedback from PRM800K. The details of data construction are introduced in Section 4.1. We then train a reward model (RM) on this dataset to predict which step-wise solution is more likely to be correct. Next, we use this step-wise RM as a reward function and fine-tune our supervised learning baseline to maximize this reward step by step using the PPO algorithm (Schulman et al., 2017), a commonly used RL algorithm for training LMs with preference-based human feedback. We call the algorithm Step-wise RLHF (Step-RLHF).

3.1 Step-wise Reward Model

Our step-wise reward model is trained to output a score for a pair of (prompt, response step) at each response step, and the step-level training data is constructed from PRM-800K which includes the model's responses and human assessments for each step of multiple solutions. The loss function for training the reward model can be framed as a classification or regression task, and the goal is to minimize the difference between the predicted scores and the human-assigned scores for the responses. The reward model is then used to optimize the performance of an artificial intelligence agent through reinforcement learning.

In (Stiennon et al., 2020), the RM is trained on a dataset of comparisons between two model outputs on the same input. They use a cross-entropy loss, with the comparisons as labels—the difference in rewards represents the log odds that one response will be preferred to the other by a human labeler. While for step-wise RM, our collected preference data contains K = 2 or K = 3 ranked responses. This produces $\binom{K}{2}$ comparisons for each step of a solution corresponding to a math problem. Following the reward model training methods in (Ouyang et al., 2022), we train on all $\binom{K}{2}$ comparisons from each step as a single batch element. This is much more computationally efficient because it only requires a single forward pass of the RM for each completion (rather than $\binom{K}{2}$ forward passes for K completions) and, because it no longer overfits, it achieves much improved validation accuracy and log loss. Specifically, the pairwise comparison loss function for our step-wise reward model is:

$$\mathcal{L}(\theta) = -\frac{1}{\binom{K}{2}} E_{(x,y_w,y_l)\sim D}[\log(\sigma(r_\theta(x,y_w) - r_\theta(x,y_l)))] \tag{1}$$

where $r_\theta(x, y)$ is the scalar output of the reward model for prompt x and step-wise completion y with parameters θ, y_w is the preferred completion at the same step out of the pair of y_w and y_l, and D is the dataset of human comparisons. σ is a smooth approximation of the hinge loss, such as the logistic loss or another differentiable function. The overall loss is often computed as the sum or average of these pairwise ranking losses over all pairs of responses in the dataset. After training a step-wise reward model to calculate a score for a response step, we then follow the step-wise PPO training algorithm to optimize the policy model step by step.

3.2 Step-wise Proximal Policy Optimization

Proximal Policy Optimization (Schulman et al., 2017) has become a popular choice in reinforcement learning due to its stability and effectiveness in training policies for a variety of tasks. The use of a clipped surrogate objective helps to prevent large policy updates, contributing to the algorithm's robustness. Following (Ouyang et al., 2022) where they utilize PPO in the fine-tuning approach RLHF to align language models, we propose a step-wise PPO (step-PPO) algorithm to fine-tune the SFT model on our environment by refining the policy optimization process in PPO. The environment is a bandit environment which presents a random customer prompt and expects a response to the prompt. Given the prompt and response, it produces rewards for each step in this response determined by the step-wise reward model and ends the episode. In addition, we add a per-token KL penalty from the SFT model at each token to mitigate over optimization of the reward model. The value function is initialized from the step-RM.

Specifically, the policy π_θ is initialized by a fine-tuned base language model. Output sequences $y^n \sim \pi_\theta(\cdot|x^n)$ are generated step by step for each prompt $x^n \in \mathcal{D}_b$ by the policy π_θ. Using step-PPO methods, we then split generated sequences y^n to the step-wise sequence set $\mathcal{S}^n$, where $y_i^n \in \mathcal{S}^n$. (Step-RLHF allows us to define customized split function for specific tasks.) After that, step-PPO uses a value model $V_\psi(s_t)$ initialized from the pre-trained step-RM to estimate the value of state s_t, and optimizes the policy model with a PPO clipped surrogate training objective. For each step-wise sequence $y_i^n \in \mathcal{S}^n$ in a completion y^n, we then compute rewards r_i^n with the pre-trained step-wise RM R. r_i^n is the reward assigned to y_i^n, which in our case is acquired using a step-level learned reward models. For each step-wise sequence y_i^n, the value targets $\{V^{\text{targ}}(s_t)\}_{t=1}^{|y_i^n|}$ at timestep t is computed with

$$\{V^{\text{targ}}(s_t)\}_{t=1}^{|y_i^n|} = r_i^n + \gamma V_\phi(s_t) \tag{2}$$

where V_ϕ is the lagging value model. And the advantage $\{A_t\}_{t=1}^{|y_i^n|}$ at timestep t is estimated by a generalized advantage estimation function (Schulman et al., 2016):

$$A_t = \delta_t + (\gamma\lambda)\delta_{t+1} + \cdots + (\gamma\lambda)^{T-t+1}\delta_T \tag{3}$$

with γ as a hyperparameter, λ as the discounting factor for rewards, and $\delta_t = V^{\text{targ}}(s_t) - V_\phi(s_t)$.

In step-PPO, the objective function for policy $\pi(\theta)$ is designed to maximize the expected cumulative reward while maintaining the policy change within a certain range. Step-PPO update the policy progressively for each step by optimizing the clipped surrogate objective. The objective function is given by:

$$\mathcal{L}(\theta) = \hat{\mathbb{E}}_t \min\left(\upsilon_t A_t, \text{clip}\left(1-\epsilon, 1+\epsilon, \upsilon_t\right) A_t\right) \tag{4}$$

where $\mathcal{L}(\theta)$ is the clipped surrogate objective. θ represents the policy parameters. $\hat{\mathbb{E}}_t$ is the empirical expectation over a batch of experiences which is made by step-level sequences. υ_t is the probability ratio of the new policy to the old policy with $\upsilon_t = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$. A_t is the advantage function, representing the advantage of taking action a in state s at time t, ϵ is a hyperparameter controlling the size of the policy update. The clipping term ensures that the policy update does not deviate significantly from the previous policy, adding a stability constraint to the optimization process.

Step-PPO also incorporates a value function to estimate the expected cumulative reward. The value function helps in reducing variance and stabilizing the training process. In step-PPO, the value model $V_\phi(s_t)$ is optimized with an expected squared-error loss:

$$\mathcal{L}(\phi) = \hat{\mathbb{E}}_t(V_\phi(s_t) - V^{\text{targ}}(s_t))^2 \tag{5}$$

Algorithm 1 Step-wise Reinforcement Learning from Human Feedback(Step-RLHF)

Initialize policy π_θ with parameters θ and value function V_ϕ with parameters ϕ
Set hyperparameters: discount factor γ, GAE parameter λ, clip parameter ϵ
for Training step $= 1$ to M **do**
    Sample a batch $\mathcal{D}_b$ from $\mathcal{D}$
    Generate output sequence $y^n \sim \pi_\theta(\cdot|x^n)$ step by step for each prompt $x^n \in \mathcal{D}_b$ by π_θ
    Construct the step-wise sequence set $\mathcal{S}^n$ by splitting y^n, where $y_i^n \in \mathcal{S}^n$
    Compute rewards r_i^n for each step $y_i^n \in \mathcal{S}^n$ with the pre-trained step-RM R
    Compute value targets $\{V^{\text{targ}}(s_t)\}_{t=1}^{|y_i^n|} = r_i^n + \gamma V_\phi(s_t)$
    Compute advantages $\{A_t\}_{t=1}^{|y_i^n|}$ using GAE for each step-wise sequence y_i^n ▷ Eq.3
    for PPO iteration $= 1, \ldots, K$ **do**
        Update policy progressively by optimizing the clipped surrogate objective:

$$\theta \leftarrow \arg\max_\theta \frac{1}{|\mathcal{D}_b|}\sum_{n=1}^{|\mathcal{D}_b|}\frac{1}{|\mathcal{S}^n|}\sum_{i=1}^{|\mathcal{S}^n|}\frac{1}{|y_i^n|}\sum_{t=1}^{|y_i^n|}\min\left(\upsilon_t A_t, \text{clip}\left(1-\epsilon, 1+\epsilon, \upsilon_t\right) A_t\right)$$

        Update value function progressively by minimizing the square-error objective:

$$\phi \leftarrow \arg\min_\phi \frac{1}{|\mathcal{D}_b|}\sum_{n=1}^{|\mathcal{D}_b|}\frac{1}{|\mathcal{S}^n|}\sum_{i=1}^{|\mathcal{S}^n|}\frac{1}{|y_i^n|}\sum_{t=1}^{|y_i^n|}(V_\phi(s_t) - V^{\text{targ}}(s_t))^2$$

    end for
end for

Finally, step-PPO is trained to optimize both policy (π_θ) and value (V_ϕ) models with their respective objectives for several iteration. No reward model is being optimized during step-PPO training.

3.3 Algorithm Overview

The Step-RLHF algorithm is outlined in Algorithm 1, encompassing a series of training steps, each involving the generation of output sequences for given prompts. A crucial aspect is the utilization of a step-wise reward model for intermediate reasoning steps. During step-PPO training, GAE is computed, and policy updating is performed at each step, contributing to the algorithm's effectiveness.

The algorithm begins by initializing the policy and value function, setting crucial hyperparameters such as learning rate, discount factor, and the number of training epochs. Notably, Step-RLHF introduces the concept of a step-wise sequence set, dividing the learning process into distinct steps for each prompt. This innovation provides a granular understanding of the agent's decision-making at different stages, fostering improved learning.

During each training step, Step-RLHF samples batches from the dataset and generates output sequences step by step for each prompt. The algorithm then constructs the step-wise sequence set by splitting the generated sequence. For each intermediate step, rewards are computed using a pre-trained step-wise reward model. These rewards are utilized to calculate value targets and advantages, leveraging the power of GAE for nuanced and adaptive learning.

Subsequently, the step-PPO iteration comes into play, progressively updating the policy and value function. The clipped surrogate objective ensures stability and mitigates the challenges associated with policy updates. This iterative process, repeated for a predefined number of epochs, refines the agent's policy incrementally.

4 Experiment

This section provides a comprehensive overview of the experimental settings in our experiments. Subsequently, we mainly elucidate the performance metrics of our models on the mathematical benchmarks MATH (Hendrycks et al., 2021).

4.1 Dataset

RM To collect training data for step-wise RM, we construct the step-wise preference data from PRM800K (Lightman et al., 2023). PRM800K provides the multiple step-by-step solutions to MATH problems sampled by the large-scale generator. Each step in the solution is assigned with a label of positive, negative, or neutral, representing correct, incorrect, ambiguous respectively. According to this step-level labeled dataset, we construct a pair of preference data by comparing the different step-level solutions with different labels whenever there are different labels in the same step. A pair of preference data might be constructed from 4 possible combination: [positive step, neutral step, negative step], [positive step, neutral step], [positive step, negative step], [neutral step, negative step].

We refer to the entire dataset of step-level preference pairs from PRM800K as SRM50K. The SRM50K training set contains 54K step-level preference pairs to 12K problems. To minimize overfitting, we create development dataset and test dataset split from the constructed preference data, with 4.8K pairs respectively. Overall, we have 54K training, 4.8K development and 4.8K test examples for step-wise reward model. For PPO training, we randomly sample 5k problems from original MATH training set as the prompts .

4.2 SFT Baselines

Massive open-source LLMs have been accessible to the AI community. We leverage the Supervised Fine-Tuned (SFT) model Qwen-chat-14b (Bai et al., 2023) as our base language model. It is recently published, which is effectively fine-tuned with human alignment techniques. We use Qwen-chat-14b as the initial model to train our step-wise RM, and also, we use Qwen-chat-14b to produce solutions step-by-step as actor in step-wise PPO.

4.3 Evaluate Benchmarks

We mainly evaluate models trained by Step-RLHF on mathematical benchmark MATH (Hendrycks et al., 2021). The MATH dataset collects math problems from prestigious math competitions such as AMC 10, AMC 12, and AIME. It contains 7500 training data and 5,000 challenging test data in seven academic areas: Prealgebra, Algebra, Number Theory, Counting and Probability, Geometry, Intermediate Algebra, and Precalculus. Furthermore, these problems are divided into five levels of difficulty, with '1' denoting the relatively lower difficulty level and '5' indicating the highest level.

Models	Wizard-Math-13B	Qwen-14B-Chat
SFT	13.04	18.38
RLHF	13.19	19.26
Step-RLHF	14.30	**20.40**

Table 1: Results on MATH test set

4.4 Experimental Settings

For step-PPO training, we initialize the policy model with the open-source supervised fine-tuning model Qwen-chat-14B (Bai et al., 2023). We name this initial policy model as SFT. For the traditional preference RLHF training, we first train a sample-level reward model which is used in sample-level PPO. The sample-level reward model is also initialized with Qwen-chat-14B (Bai et al., 2023), and optimized by the pairwise comparison loss function using holistic feedback collected from PRM800k (Lightman et al., 2023). Then we train a policy model by PPO, which updates the policy sample by sample. We name this policy model as RLHF.

We compare our proposed method step-RLHF with the initial SFT policy model and RLHF with holistic preference-based rewards. The sample-level reward models used in RLHF are trained on 1w examples with annotated feedback. Our policy model is based on Qwen-chat-14B. During RL exploration, we use greedy (top-k = 0) sampling decoding with temperature = 0.5, which is set the same for preference RLHF and step-RLHF. The value model used during RL training is initialized with corresponding reward model which is also trained based on Qwen-chat-14B. During inference, we use greedy decoding to generate responses.

4.5 Main results

Step-RM We first analyze the performance of each reward model in predicting the score of the generated solutions. We train two types of reward model (step-level and sample-level) adequately in order to provide a accurate reward for comparing different PPO training method fairly. Our proposed step-wise reward model has an accuracy of 84.7 in pairwise comparison on the test set. The sample-level preference-based reward model reaches an accuracy of 83.2. We also investigate the discrimination between average scores of correct and incorrect solutions, which is -0.25 and -0.4 respectively. In this case, the test result shows that our step-RM has a excellent ability to distinguish between better

	Wizard-Math	Qwen-chat
SFT	13.19	18.38
RLHF	13.04	19.26
SPPO + sample-RM	13.57	19.44
SPPO + step-RM	**14.30**	**20.40**

Table 2: Comparison SPPO with sample-RM and step-RM on MATH

	Wizard-Math	Qwen-chat
SFT	13.19	18.38
RLHF	13.04	19.26
Step-RM + PPO	13.57	18.88
Step-RM + Step-PPO	**14.30**	**20.40**

Table 3: Comparison RLHF with/without step-PPO on MATH

and worse step-wise solutions.

Step-RLHF Step-RLHF outperforms SFT and preference RLHF on all error types. Table 1 show that our step-RLHF leads to 2% problem-solving accuracy increment, compared to the SFT model. Step-RLHF also increases the accuracy by 1.2% on MATH test set, compared to traditional preference RLHF which is trained by PPO with sample-RM. Obviously, step-RLHF showcases its applicability in mathematical tasks, illustrating that learning from step-wise reward functions and updating the policy step by step significantly improves model performance. This work represents a crucial step towards enhancing the adaptability and precision of language models in mathematical tasks through the integration of step-wise human feedback within the RLHF framework.

4.6 Ablation study

Table 2 compares different reward models used during step-PPO training. The sample-RM is trained by the sample-level preference data, which means the preference data contains a pair of complete solutions combined by [correct solution, wrong solution]. While the step-RM is trained by the step-wise preference data SRM50k which is detailed in Section 3.1 and Section 4.1. We observe that step-PPO training with sample-RM outperforms the traditional RLHF which is trained by PPO with sample-RM. It shows the step-PPO training is also effective with sample-RM, and even improves the accuracy for MATH compared to traditional RLHF. In addition, when using step-RM during step-PPO, the accuracy are improved more than sample-RM, illustrating the effectiveness of step-RLHF with both step-wise RM and step-wise PPO training.

Furthermore, another ablation experiment is conducted to explore the effect of step-PPO and traditional PPO when using the same reward model. Table 3 shows the accuracy of step-wise PPO outperforms the traditional PPO with the step-wise reward as the human feedback. It illustrates this technology that step-PPO training utilizes step-wise reward model to predict reward for each solution step and updates the policy step by step stimulates the potential of the policy for multi-step reasoning task in RLHF. Another results in Table 3 shows using step-RM for sample-level PPO training decreases the problem-solving accuracy compared to traditional RLHF. It is obvious that the pre-trained step-wise reward model provides the step-level feedback for each step in the solution. As for the traditional PPO training with sample-level solution, step-RM is not able to provide a holistic feedback accurately.

5 Limitations and Future Work

One limitation of our framework comes from the additional compute cost of getting step-wise reward, which need human to annotate answers step by step for providing step-wise feedback, compared to RLHF with a holistic reward. Another limitation is that different tasks may have different definitions of fine-grained feedback in terms of the density level of step. Therefore, defining the step split function that is well-suited for a task requires non-trivial manual effort. In this work, we leverage the open-source step-wise dataset PRM800K, showing the effectiveness of step-RLHF on MATH. Our method also generalizes to other multi-step reasoning task with step-wise annotation providing.

6 Conclusion

In conclusion, Step-RLHF presents a novel approach to reinforcement learning from human feedback. The incorporation of a step-wise reward model, coupled with step-wise policy updating during PPO training, sets the algorithm apart in terms of efficiency and performance. Experimental results underscore the potential of Step-RLHF as a promising method for addressing the challenges posed by complex sequential decision-making tasks.

References

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, Nicholas Joseph, Saurav Kadavath, Jackson Kernion, Tom Conerly, Sheer El-Showk, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Tristan Hume, Scott Johnston, Shauna Kravec, Liane Lovitt, Neel Nanda, Catherine Olsson, Dario Amodei, Tom Brown, Jack Clark, Sam McCandlish, Chris Olah, Ben Mann, and Jared Kaplan. 2022a. Training a helpful and harmless assistant with reinforcement learning from human feedback.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, Carol Chen, Catherine Olsson, Christopher Olah, Danny Hernandez, Dawn Drain, Deep Ganguli, Dustin Li, Eli Tran-Johnson, Ethan Perez, Jamie Kerr, Jared Mueller, Jeffrey Ladish, Joshua Landau, Kamal Ndousse, Kamile Lukosuite, Liane Lovitt, Michael Sellitto, Nelson Elhage, Nicholas Schiefer, Noemi Mercado, Nova DasSarma, Robert Lasenby, Robin Larson, Sam Ringer, Scott Johnston, Shauna Kravec, Sheer El Showk, Stanislav Fort, Tamera Lanham, Timothy Telleen-Lawton, Tom Conerly, Tom Henighan, Tristan Hume, Samuel R. Bowman, Zac Hatfield-Dodds, Ben Mann, Dario Amodei, Nicholas Joseph, Sam McCandlish, Tom Brown, and Jared Kaplan. 2022b. Constitutional ai: Harmlessness from ai feedback.

Samuel Gehman, Suchin Gururangan, Maarten Sap, Yejin Choi, and Noah A. Smith. 2020. Realtoxicityprompts: Evaluating neural toxic degeneration in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2020*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the MATH dataset. *CoRR*, abs/2103.03874.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213.

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

Haipeng Luo, Qingfeng Sun, Can Xu, Pu Zhao, Jianguang Lou, Chongyang Tao, Xiubo Geng, Qingwei Lin, Shifeng Chen, and Dongmei Zhang. 2023. Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct. *arXiv preprint arXiv:2308.09583*.

Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan McDonald. 2020. On faithfulness and factuality in abstractive summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

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

Rajkumar Ramamurthy, Prithviraj Ammanabrolu, Kianté Brantley, Jack Hessel, Rafet Sifa, Christian Bauckhage, Hannaneh Hajishirzi, and Yejin Choi. 2023. Is reinforcement learning (not) for natural language processing: Benchmarks, baselines, and building blocks for natural language policy optimization.

John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel. 2016. High-dimensional continuous control using generalized advantage estimation. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

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

Ivan Stelmakh, Yi Luan, Bhuwan Dhingra, and Ming-Wei Chang. 2022. ASQA: Factoid questions meet long-form answers. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 8273–8288, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. 2020. Learning to summarize with human feedback. *Advances in Neural Information Processing Systems*, 33:3008–3021.

Jonathan Uesato, Nate Kushman, Ramana Kumar, H. Francis Song, Noah Yamamoto Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. 2023. Solving math word problems with process-based and outcome-based feedback.

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

Zeqiu Wu, Yushi Hu, Weijia Shi, Nouha Dziri, Alane Suhr, Prithviraj Ammanabrolu, Noah A Smith, Mari Ostendorf, and Hannaneh Hajishirzi. 2023. Fine-grained human feedback gives better rewards for language model training. *arXiv preprint arXiv:2306.01693*.

DanielM. Ziegler, Nisan Stiennon, Jeffrey Wu, T.B. Brown, Alec Radford, Dario Amodei, PaulF. Christiano, and Geoffrey Irving. 2019. Fine-tuning language models from human preferences. *arXiv: Computation and Language,arXiv: Computation and Language*.

A Example Appendix

This is a section in the appendix.