Mutual-Taught for Boosting Policy and Reward Models

Anonymous ACL submission

Abstract

Preference optimization has emerged as an effective technique for aligning large language models (LLMs) with human objectives. However, as training progresses, distribution shifts can occur between newly generated model samples and the data used to train the reward model (RM), reducing the RM's effectiveness and constraining the policy model's (PM) performance. To address this challenge, we propose a self-training technique called **Mutual-Taught** that jointly improves both the PM and the RM without relying on additional human supervision. Our method is inspired by the Expectation-Maximization (EM) algorithm. In the E-step, we update the PM based on feedback from the current RM, guiding the PM toward a better approximation of the latent optimal preference distribution. In the M-step, we update the RM by constructing training data from the PM's outputs before and after the E-step update, thereby adapting the RM to the evolving policy distribution. Experimental results show that this iterative process steadily improves both models. Our 8B policy model, LLaMA3-8B-Instruct-MT, achieves a length-controlled win rate of 52.0% on AlpacaEval-2. Meanwhile, our 8B reward model, FsfairX-LLaMA3-RM-MT, attains performance on par with GPT-4o-2024-08-06 on RewardBench.

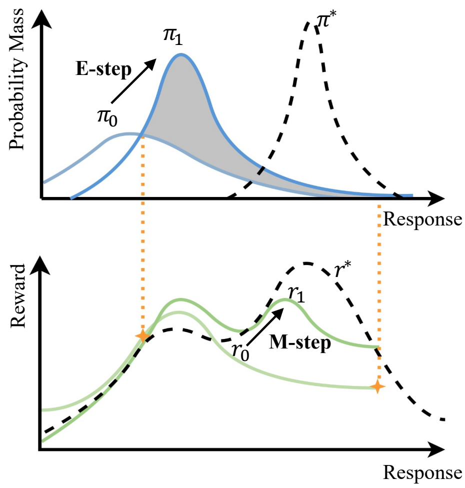

Figure 1: An illustration of the Mutual-Taught intuition. The top curve represents the evolving policy model distribution π_i, and the bottom curve shows the reward model's preference estimates r_i. After the policy update (E-step), the refined policy model π_1 exhibits a higher probability of generating high-reward responses compared to the previous policy π_0, as indicated by the shaded region. These improvements, resulting from the distribution shifts, are used to enhance the reward model's ability (M-step) to provide more reliable feedback in high-reward regions. Over successive E- and M-steps, both the policy and reward models progressively adapt, approaching their optimal distributions (π^*, r^*).

1 Introduction

As large language models (LLMs) are fine-tuned to align with human preferences using techniques like Reinforcement Learning from Human Feedback (RLHF) (Ouyang et al., 2022) and Direct Preference Optimization (DPO) (Rafailov et al., 2024), distribution shifts may arise. Over time, the distribution of outputs generated by the evolving model may diverge from that of the data used to train the reward model or the original preference dataset. This misalignment can create a feedback loop: as the model adapts, it may produce outputs that score highly under the current reward model but fail to genuinely capture human preferences, a scenario often referred to as "reward hacking" (Gao et al., 2023; Zheng et al., 2023b). Such distortions ultimately undermine the reliability of alignment.

A straightforward solution is to continuously solicit new human annotations for recently generated samples (Touvron et al., 2023). However, this approach is not scalable due to its substantial reliance on human labor. Another strategy employs LLM-as-a-Judge prompting (Yuan et al., 2024; Wu et al., 2024a), where an LLM provides its own reward signals. While iterative DPO techniques based on this principle can refine both instruction-following and judgment abilities, they typically require strong base models or pre-training on judgment datasets to establish effective judgment skills.

This paper explores whether it is possible to automatically improve both the policy and reward models without external supervision. Our central research question is: *How can we automatically generate high-quality feedback for LLM self-training and effectively guide reward model optimization?* To address this challenge, we introduce a self-training framework called **Mutual-Taught**, grounded in the Expectation-Maximization (EM) algorithm. In our framework, the E-step refines the policy model by leveraging feedback from the current reward model, thereby guiding it toward the optimal latent preference distribution. In the M-step, the reward model is updated using comparison data derived from the policy's outputs before and after the E-step. These pseudo-preference pairs naturally arise from the changing policy distribution and obviate the need for external labels.

A key insight of our approach is that the distribution shifts arising from policy model updates can be harnessed to produce the contrastive examples for reward model improvement. Through this mutual teaching process, both models continuously benefit from each other's evolving state. Empirical results show that our Mutual-Taught framework consistently outperforms previous methods, achieving robust self-improvement without human involvement. Notably, the improved reward model generalizes well and effectively guides optimization for a range of policy models.

2 Related Work

Offline preference optimization Reinforcement Learning from Human Feedback (RLHF) (Ouyang et al., 2022) has emerged as a pivotal approach of preference optimization. However, it depends on reinforcement learning techniques such as Proximal Policy Optimization (PPO) (Schulman et al., 2017), which are challenging to implement and often unstable during training. To simplify and stabilize the RLHF process, Direct Preference Optimization (DPO) (Rafailov et al., 2024) was proposed. DPO trains a policy model directly from human-annotated preference pairs using a simple classification loss derived from the Bradley-Terry model (Bradley and Terry, 1952). Besides DPO, various preference optimization objectives have been proposed to improve performance and simplify training, including SLiC-HF (Zhao et al., 2023), KTO (Ethayarajh et al., 2024), RSO (Liu et al., 2023), ORPO (Hong et al., 2024), and SimPO (Meng et al., 2024). However, the absence of an explicit reward model in these methods limits their ability to adapt to the evolving policy distribution and to generate new preference pairs effectively.

Iterative preference optimization With an explicit reward model, the preference optimization methods mentioned above can be applied repeatedly over multiple rounds. In each round, new data is generated by the policy obtained from the previous round, annotated with a reward score and then used to train a stronger policy. For example, Xu et al. (2023) apply PCO and DPO to an iterative manner by annotating new preferences with a fixed reward model. ReST$^{\text{EM}}$ (Singh et al., 2023) proposed an Expectation-Maximization (EM) framework, where in each round, the new policy is obtained by minimizing the reward-weighted negative log-likelihood loss on data generated by the old policy. SELM (Zhang et al., 2024b) and XPO (Xie et al., 2024) augment the DPO objective with a novel and principled exploration bonus, enabling the algorithm to explore beyond the initial model and human feedback data. SAIL (Ding et al., 2024) generates preference data by combining initial offline preferences with model-generated preferences, where the model itself estimates the likelihood of one response being preferred over another using its learned policy. SPIN (Chen et al., 2024), DNO (Rosset et al., 2024) and SPPO (Wu et al., 2024b) leverage self-play mechanism to iteratively refine the policy towards achieving Nash equilibrium by optimizing general preferences. These approaches often overlook distribution shifts, which can limit the potential for policy improvement.

To address this issue, Ouyang et al. (2022) collects preference data on the current best policy, which is then annotated by humans and used to train a new reward model, this process consumes significant annotation resources. ReST-MCTS* (Zhang et al., 2024a) uses the policy to perform a modified Monte Carlo Tree Search to generate solutions and evaluates them against the ground truth for process reward model training. However, its reliance on ground truth restricts the method's applicability to only a few specific domains. Another line of work uses the policy model to judge its own responses in an LLM-as-a-Judge mechanism

(Zheng et al., 2023a), which eliminates the reward model and simultaneously updates the knowledge of the judge. Self-rewarding (Yuan et al., 2024) and Meta-rewarding (Wu et al., 2024a) language models generate responses to prompts using the current policy and then assign scores to each response themselves to create preference data for the next iteration of training.

3 Preliminaries

3.1 Reward Modeling

In RLHF, a reward model $r(y; x)$ is first trained to predict human preference scores for responses y given prompts x. The reward model is typically trained using human-annotated preference pairs (x, y_w, y_l), where y_w is preferred over y_l.

Bradley-Terry reward model (Bradley and Terry, 1952) is commonly used to model the probability that one response is preferred over another:

$$\begin{aligned} P(y_w \succ y_l \mid x) &= \sigma(r(y_w; x) - r(y_l; x)) \\ &= \frac{\exp(r(y_w; x))}{\exp(r(y_w; x)) + \exp(r(y_l; x))}. \end{aligned} \quad (1)$$

The reward model is then trained by maximizing the log-likelihood of observed preferences: $\log P(y_w \succ y_l \mid x)$.

3.2 Direct Preference Optimization (DPO)

DPO (Rafailov et al., 2024) simplifies the training process by combining the two-step procedure of PPO (Schulman et al., 2017) into a single unified objective. Specifically, DPO conducts a closed-form solution for the reward function, yieding the following loss formulation:

$$\begin{aligned} &\mathcal{L}_{\text{DPO}} = \\ &-\log \sigma \left(\beta \log \frac{\pi_\theta(y_w \mid x)}{\pi_{\text{ref}}(y_w \mid x)} - \beta \log \frac{\pi_\theta(y_l \mid x)}{\pi_{\text{ref}}(y_l \mid x)} \right). \end{aligned} \quad (2)$$

While DPO offers enhanced stability and ease of optimization, its offline nature and the absence of an explicit reward model limit its ability to adapt to the evolving policy distribution effectively.

4 Mutual-Taught

Conventional iterative preference learning often treats the reward model (RM) as a fixed oracle that perfectly encodes an "optimal" preference distribution. In practice, however, this assumption fails to hold as the policy model (PM) improves and its output distribution shifts (Touvron et al., 2023; Cheng et al., 2024). The static RM, trained under outdated conditions, may no longer accurately reflect the evolving notion of optimality, resulting in increasingly misaligned feedback that caps the PM's potential.

To address this challenge, we propose Mutual-Taught, a self-training framework that jointly optimizes both the PM and the RM. By adopting an Expectation-Maximization (EM)-inspired perspective, Mutual-Taught treats this latent optimal distribution as a hidden variable whose properties must be inferred and re-estimated over time. Through iterative refinement—improving the PM to better approximate the latent distribution (E-step) and updating the RM to more accurately reflect this improved approximation (M-step)—Mutual-Taught co-evolves both models towards the latent optimal distribution. This approach enhances preference alignment without requiring human annotations.

4.1 Objective of Mutual-Taught

Consider a dataset $\mathcal{D}$ of prompts $x \in \mathcal{X}$. For each prompt x, we assume a latent "optimal" response distribution $Q^*(y \mid x)$, which is unobservable. Our goal is twofold: to learn a policy model $\pi(y \mid x)$ that approximates $Q^*(y \mid x)$, and to optimize a reward model $r(y; x)$ that evaluates responses in alignment with $Q^*(y \mid x)$. The joint optimization objective can be expressed as:

$$\pi^*, r^* = \arg\max_{\pi, r} \mathbb{E}_{x \sim \mathcal{D}, y \sim Q^*(\cdot|x)}[r(y; x)]. \quad (3)$$

Since $Q^*(y \mid x)$ is unknown, we adopt an EM-inspired approach. In the E-step: Estimate the latent distribution Q^* based on the current model parameters. In the M-step: Update the model parameters to better fit this estimate. In our setting, the E-step corresponds to updating π so that π_t more closely approximates Q^*, and the M-step updates r to align with this improved approximation.

E-step (Policy model update): With the current RM r_{t-1} fixed, we update π_t by maximizing expected reward:

$$\pi_t = \arg\max_{\pi} \mathbb{E}_{x \sim \mathcal{D},\, y \sim \pi(\cdot|x)}[r_{t-1}(y; x)]. \quad (4)$$

Although π_t may not precisely match Q^*, this update moves π_t closer to what the current RM considers optimal, serving as a practical surrogate for the latent distribution.

M-step (Reward model update): With π_t fixed, we update the RM r_t so that it better reflects the

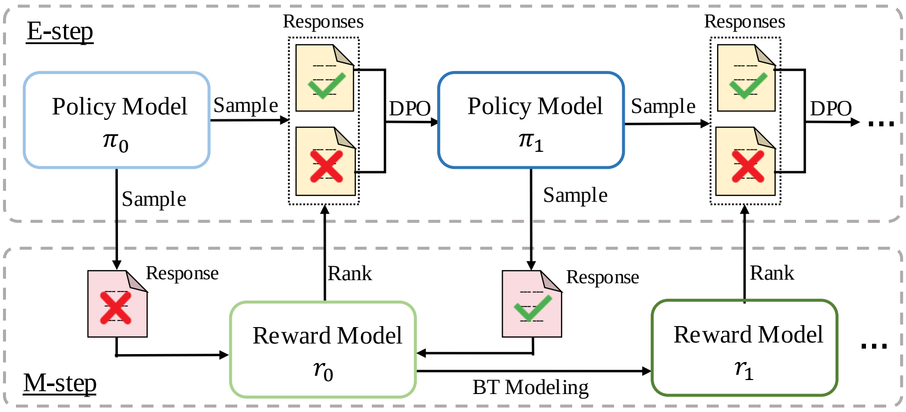

Figure 2: Overview of the Mutual-Taught framework. The process alternates between policy model updates (E-step) and reward model updates (M-step). In the E-step, the policy model is fine-tuned using feedback from the current reward model to align the policy with the optimal preference distribution. In the M-step, the reward model is updated through contrastive comparisons of policy model outputs before and after updates, allowing it to adapt to the evolving policy without requiring additional human annotations.

Algorithm 1 Mutual-Taught

Input: Initial PM π_0, initial RM r_0, dataset $\mathcal{D}$, number of iterations T.
Partition $\mathcal{D}$ into subsets $\mathcal{D}_1, \ldots, \mathcal{D}_T, \mathcal{D}_R$.
for each iteration $t = 1 \ldots T$ **do**
    E-step: Update π_t by sampling responses from π_{t-1} for $x \sim \mathcal{D}_t$, evaluating them with r_{t-1}, and optimizing π_t to increase expected rewards.
    M-step: For each $x \sim \mathcal{D}_R$, sample $y_t \sim \pi_t(x)$ (chosen) and $y_{t-1} \sim \pi_{t-1}(x)$ (rejected). Create preference pairs (x, y_t, y_{t-1}) and update r_t.
end for
Output: Final PM π_T and RM r_T.

improved approximation to Q^* provided by π_t. Specifically, we consider pairs (y_t, y_{t-1}) of responses drawn from π_t and π_{t-1}. Since π_t has been optimized under r_{t-1}, we treat responses from π_t as "chosen" (closer to the latent optimum) and those from π_{t-1} as "rejected", yielding a pseudo-labeled pairwise preference signal:

$$r_t = \arg\max_r \mathbb{E}_{x\sim\mathcal{D}, y_t\sim\pi_t(\cdot|x), y_{t-1}\sim\pi_{t-1}(\cdot|x)} \left[\log P_r(y_t \succ y_{t-1} \mid x)\right]. \quad (5)$$

This update encourages r_t to assign higher preference probabilities to responses that are closer to the latent optimal distribution as approximated by π_t.

4.2 Algorithmic Overview

Algorithm 1 summarizes our Mutual-Taught approach. In classical EM, we iteratively refine both a variational approximation of latent variables and the model parameters. Analogously, we treat Q^* as the latent variable and π_t as an evolving surrogate. By refining π_t in the E-step and adjusting r_t in the M-step, both models progressively align with the latent optimal distribution Q^*.

By reframing preference optimization as latent variable estimation, Mutual-Taught replaces the static, fixed-oracle RM paradigm with a dynamic, co-evolving interplay between the PM and RM. This synergy leverages EM-like reasoning to achieve improved preference alignment without additional human annotations.

5 Experiments

5.1 Experimental Setup

Base model and dataset We use LLaMA3-8B-Instruct (Dubey et al., 2024) as our base policy model and FsfairX-LLaMA3-RM-v0.1 (Xiong et al., 2024) as the base reward model, which is one of the top models on RewardBench (Lambert et al., 2024) and provides open-source code, facilitating iterative training. Following previous work, we use the UltraFeedback dataset (Cui et al., 2024) with approximately 60,000 prompts from diverse sources. To prevent overfitting during fine-tuning, we divide the dataset into three parts: two parts for policy model iteration and one for reward model iteration. In each iteration, we generate $K = 5$ responses per prompt with a temperature of 0.8 and top-p $= 0.95$. Duplicate generations are removed.

Evaluation benchmarks Since Mutual-Taught aims to automatically improve both the policy model and the reward model, we evaluate the performance of each component separately. *Policy model instruction following*: We utilize two widely recognized automatic evaluation benchmarks where GPT-4 acts as the judge: AlpacaEval-2 (Li et al., 2023) and Arena-Hard (Li et al., 2024). Each benchmark targets different aspects of model performance. AlpacaEval-2 assesses chat capabilities using 805 instructions spanning a wide range of prompts, evaluated through length-controlled (LC) win rate and raw win rate (WR) metrics. Arena-Hard presents more challenging tasks, including 500 well-defined technical problem-solving queries. *Reward model judgment accuracy*: We assess the reward model's accuracy using RewardBench (Lambert et al., 2024), which measures performance across four different categories: Chat, Chat-Hard, Safety, and Reasoning.

Baselines We compare our approach against the following baseline models and methods:

- Base policy model and base reward model: We use LLaMA3-8B-Instruct, an instruction-following LLM developed by Meta, as our base policy model. For the reward model, we employ FsfairX-LLaMA3-RM-v0.1, a high-performing reward model fine-tuned from LLaMA3-8B-Instruct.

- Offline preference optimization methods: We implement DPO (Rafailov et al., 2024), IPO (Gheshlaghi Azar et al., 2024) and SimPO (Meng et al., 2024). Preference pairs are derived from multiple responses generated by the base policy model, with scores provided by the base reward model.

- Iterative preference optimization methods: We implement SPPO (Wu et al., 2024b) and Meta-rewarding (Wu et al., 2024a). Since these methods do not update the reward model, we use all three portions of the dataset for policy training, performing three iterations.

To ensure a fair comparison, the sampling settings used in the above experiments align with those applied in the Mutual-Taught. Further implementation details for these baselines can be found in Appendix A.

Implementation details We conduct Mutual-Taught between the policy and reward models for two iterations. In each iteration, both models are trained for one epoch using a cosine learning rate schedule with a warmup ratio of 0.1. For each policy model iteration, we initialize the model from the previous round and generate responses using the current policy. Preference data is then derived using the reward model at the current iteration. The policy model is optimized via DPO with a beta of 0.01, a batch size of 128, a maximum sequence length of 2,048 tokens, and a learning rate of 7×10^{-7}. To mitigate the risk of overfitting on the same prompts across iterations, each reward model iteration started from the base reward model. The reward model is trained on preference pairs consisting of chosen and rejected responses sampled from the current and preceding policy models. We use a batch size of 512, a maximum sequence length of 2,048 tokens, and an empirically set learning rate of 2×10^{-6}. All experiments are conducted on 8 NVIDIA A100 GPUs. Further details are provided in Appendix A.

5.2 Main Results

Iterative performance improvement on policy model We first report the performance of Mutual-Taught and baseline methods on the instruction-following benchmarks AlpacaEval-2 and Arena-Hard in Table 1. For AlpacaEval-2 and Arena-Hard, Mutual-Taught delivers substantial improvements to the LLaMA-3-8B-Instruct model, achieving a 28.9 points increase in length-controlled (LC) win rate and a 16.9 points increase in win rate, respectively. Notably, the proposed method provides consistent improvements in each iteration, validating the robustness of our approach. Compared to other baselines, our method demonstrates clear superiority on AlpacaEval-2 and Arena-Hard. Additionally, when comparing iterative preference optimization methods, we observe that methods using a reward model for preference feedback (e.g., SPPO and Mutual-Taught) perform significantly better than methods relying on LLM-as-a-judge feedback (e.g., Meta-rewarding). This indicates that a reward model, fine-tuned through supervised training, offers stronger initial judgment capabilities than the policy model itself and provides more

Model	AlpacaEval-2			Arena-Hard	
	LC Win Rate	Win Rate	Avg. Len	Win Rate	Avg. Len
Base Policy Model					
LLaMA-3-8B-Instruct	23.1	23.1	1899	20.6	585
Offline Preference Optimization Methods					
SimPO	47.9	46.3	1934	32.5	552
IPO	43.7	42.1	1899	34.5	569
DPO	44.7	42.7	1945	33.1	557
Iterative Preference Optimization Methods					
Meta-rewarding Iter1	34.2	32.6	1893	27.7	531
Meta-rewarding Iter2	36.4	34.5	1876	27.0	530
Meta-rewarding Iter3	37.5 (↑ 14.4)	35.2	1868	27.9 (↑ 7.3)	530
SPPO Iter1	39.4	39.5	2021	30.6	570
SPPO Iter2	41.0	44.4	2396	34.4	653
SPPO Iter3	46.4 (↑ 23.3)	48.5	2128	33.6 (↑ 13.0)	542
DPO Iter1	33.6	33.8	1989	30.3	559
DPO Iter2	43.4	42.3	1961	33.3	587
DPO Iter3	47.2 (↑ 24.1)	48.7	1930	34.7 (↑ 14.1)	571
Our Methods					
Mutual-Taught Iter1	37.1	36.5	1957	33.5	553
Mutual-Taught Iter2	**52.0** (↑ 28.9)	**56.0**	2214	**37.5** (↑ 16.9)	692

Table 1: Overall results of our proposed Mutual-Taught method with LLaMA-3-8B-Instruct as the policy model, compared against various baseline methods on AlpacaEval-2 and Arena-Hard. The improvement is calculated relative to LLaMA-3-8B-Instruct. Text in bold indicates the best performance.

Model	Chat	Chat Hard	Safety	Reasoning	Average
GPT-4o-2024-08-06	96.10	76.10	88.10	86.60	86.70
FsfairX-LLaMA3-RM-v0.1	99.40	65.10	87.80	86.40	84.70
Mutual-Taught Iter1	98.32	62.61	84.86	96.60	85.60
Mutual-Taught Iter2	98.32	65.90	87.26	95.69	86.80

Table 2: Out-of-distribution (OOD) evaluation results of the reward models on RewardBench.

effective guidance throughout the iterative process.

It is noteworthy that our method employs only two-thirds of the available datasets for updating the policy model, reserving the remaining third for updating the reward model. Despite using less data for policy model iterations compared to other iterative training baselines, we achieve significantly better performance on AlpacaEval-2 and Arena-Hard. This outcome highlights the importance of synchronously updating both the policy and reward models during the iterative training process. We believe that enhancing the reward model can yield greater benefits than simply increasing the amount of data used to train the policy model.

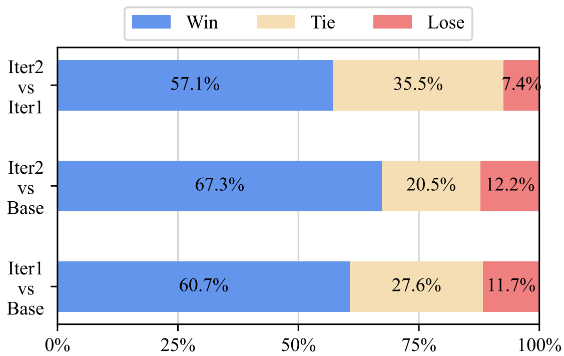

Figure 3: In-distribution (ID) evaluation results of the reward models. We compare the reward models at different iterations and show the win, tie, lose rates.

Iterative performance improvement on reward model To assess the effectiveness of Mutual-Taught in improving the reward model, we evaluate the reward models obtained during the iterative process from two perspectives:

In-distribution (ID): This test evaluates whether the reward model (RM) improves in selecting optimal responses after each iteration. Specifically, we assess the RM's performance using the iteration data employed during RM training. For each evaluation prompt, the policy model (after two iterations) generates five candidate responses. The base RM, along with the RMs from the first and

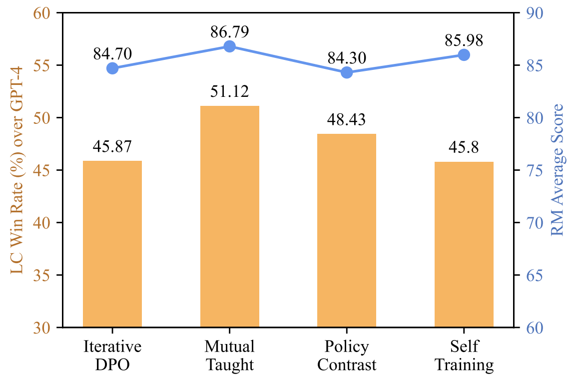

Figure 4: The impact of different reward model data synthesis strategies on the performance of Mutual-Taught.

second iterations, then selects the best response from these candidates. To measure performance differences between the RMs, GPT-4 serves as a judgment model, conducting pairwise comparisons of the responses selected by the different RMs. As shown in Figure 3, the resulting RM achieves a progressively higher win rate against the base RM as iterations advance. This highlights that the RM's ability to identify high-quality responses improves with each iteration, contributing to the enhancement of the policy model in subsequent iterations.

Out-of-distribution (OOD): We further evaluate the RM's generalization ability under OOD conditions using RewardBench. As shown in Table 2, the RM achieves consistent improvements after each iteration, with an average score increase of 2.1 points after two iterations, approaching the performance of GPT-4o-2024-08-06. Notably, the most significant contribution to the improvement comes from the enhancement in reasoning capabilities, with FsfairX-iter-2 achieving a 9.29 points increase in reasoning compared to the base RM. We attribute this to the strong reasoning ability of LLaMA3-8B-Instruct, which provides high-quality feedback on reasoning prompts to guide the RM effectively.

5.3 Ablation Study

Our main hypothesis is that the updated policy distribution is, on average, superior to the previous policy distribution. This improvement enables the reward model to learn a better preference distribution from the current data. To understand the underlying reasons for the effectiveness of Mutual-Taught, we conduct an ablation study focusing on two aspects: the impact of updating the reward model on policy iteration, and the effect of data synthesis strategies on reward model iteration. As illustrated in Figure 4, if the reward model is not updated, the overall iterative training degenerates into iterative DPO, leading to a significant decline in the policy model's performance after iteration. This observation underscores the effectiveness of Mutual-Taught in optimizing the reward model by leveraging comparisons between the policy model's outputs before and after updates.

In our implementation, to prevent excessive knowledge forgetting during the optimization of the reward model's distribution, each iteration of RM training incorporates two types of data: **D (PM)**: Data generated by the current policy model (self-training data). **D ($\mathbf{PM_{new}}$, $\mathbf{PM_{old}}$)**: Contrastive data comparing outputs from the new and old policy models. Both data types are unseen by the reward model, and using either type alone could independently improve RM performance. Notably, using only D (PM) is akin to self-training for the RM. To investigate the impact of each data type on RM performance and the overall iterative process, we conduct experiments by replacing the training data with only a single data type at each RM iteration.

As shown in Figure 4, the policy model's performance declines in both data-type ablation scenarios. Specifically, when only self-training data is used, the policy model's performance drops by 5.32 points, though the RM performance does not show a significant decline. When using only policy comparison data, the RM performance declines slightly, but the policy model's performance is relatively less affected. We hypothesize that self-training data tends to reflect the original distribution of the RM, which helps prevent catastrophic forgetting but struggles to model better preference distributions, making it less effective in guiding the policy model for the next iteration. In contrast, using comparison data between the updated and previous policy models aligns more closely with the iterative optimization goal, allowing the RM to approach a better preference distribution and provide more effective feedback for the next policy model update. The combination of both data types in Mutual-Taught achieves a balance between preventing knowledge forgetting and modeling improved preference distributions, leading to better iterative performance than using either data type alone.

5.4 Further Analysis

Generalization of the iterated reward model A critical aspect of our approach is whether an iterated reward model (RM), trained using outputs from a specific policy model (LLaMA3-8B-Instruct), can effectively generalize to guide the optimization of other models. To investigate this, we take the RM obtained after two iterations and apply it to train an entirely different policy model, Mistral-7B-Instruct-v0.2 (Jiang et al., 2023), with a single round of DPO training on the UltraFeedback dataset.

Model	AlpacaEval-2	
	LC Win Rate	Win Rate
Mistral-7B-Instruct-v0.2	19.39	15.75
w/ RM-Base	41.96	42.77
w/ RM-Iter1	44.81	43.29
w/ RM-Iter2	**45.73**	**50.02**

Table 3: Results from using reward models at different iterations in the main experiment to guide DPO training of Mistral-7B-Instruct-v0.2.

As shown in Table 3, using the RM after two Mutual-Taught iterations increases the model's performance on AlpacaEval-2 by up to 3.77 points over the base RM. This indicates that the iterated RM, trained exclusively on outputs from LLaMA3-8B-Instruct, can still effectively capture generalizable preference signals. These improved preference representations, in turn, enable the RM to guide and enhance the optimization of other models.

Compatibility with different preference objectives In the main experiments, we primarily used DPO to optimize the policy model during the E-step. We also explored integrating Mutual Taught with different preference optimization objectives, specifically SimPO and IPO.

Model	AlpacaEval-2		
	LC Win Rate	Win Rate	Avg. Len
SimPO	47.94	46.25	1934
IPO	43.72	42.11	1899
SimPO-MT	49.78	49.61	2147
IPO-MT	50.73	50.29	2029

Table 4: Experimental results with different preference optimization objectives in Mutual-Taught E-step. SimPO-MT and IPO-MT represent iterative training with SimPO and IPO, respectively.

As shown in Table 4, Mutual-Taught consistently provides significant improvements in preference optimization across both scenarios, highlighting its strong compatibility with various preference learning objectives. The pairwise win rates, measured by FsfairX-LLaMA3-RM-v0.1 (Xiong et al., 2024),

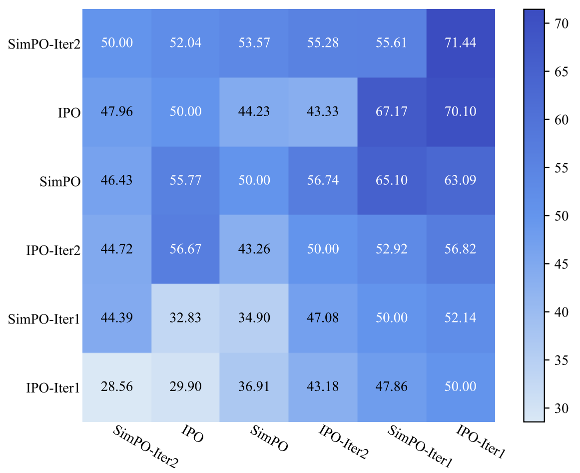

Figure 5: Pairwise evaluation of models with different preference optimization objectives in the Mutual-Taught framework on Alpaca-Eval 2 using FsfairX-LLaMA3-RM-v0.1.

are presented in Figure 5. In all preference optimization objectives, updated models consistently outperformed previous ones. However, SimPO surpassed IPO in the final iteration. Interestingly, IPO performed better than SimPO on standard benchmarks evaluated by GPT-4 against ground-truth answers. We attribute SimPO's final iteration advantage to its tendency to generate longer sequences, exploiting the length bias in FsfairX-LLaMA3-RM-v0.1, which favors longer outputs.

6 Conclusion

This paper introduces Mutual-Taught, an approach to automatically improve policy and reward models without relying on external supervision signals. The method follows an expectation-maximization (EM)-based iterative process, where in each iteration, the policy model is improved using preference feedback from the reward model to provide better observations for training the reward model. Then, comparisons between the policy model's observations before and after updates are leveraged to optimize the reward model's distribution. We demonstrated that this iterative process can continuously enhance both the policy and reward models. The resulting policy model achieves significant improvements over existing methods, such as DPO, SPPO, and Meta-rewarding, across multiple benchmarks, including AlpacaEval-2 and Arena-Hard. Moreover, the iterated reward model achieves performance comparable to GPT-4o-2024-08-06 on RewardBench.

Limitations

While Mutual-Taught demonstrates promising results, it relies on the assumption that the policy model's improvements can be effectively captured and utilized by the reward model through self-generated data. In scenarios where the policy model does not improve significantly across iterations, the effectiveness of this method may be limited. Additionally, the approach requires careful tuning of hyperparameters to balance the updates between the policy and reward models.

Ethics Statement

All experiments in this study were conducted using publicly available datasets that do not contain any private information. Our work does not involve the analysis or utilization of identity characteristics, and we do not engage in any form of gender or racial discrimination.

References

Ralph Allan Bradley and Milton E. Terry. 1952. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39:324.

Zixiang Chen, Yihe Deng, Huizhuo Yuan, Kaixuan Ji, and Quanquan Gu. 2024. Self-play fine-tuning converts weak language models to strong language models. In *International Conference on Machine Learning*.

Pengyu Cheng, Yifan Yang, Jian Li, Yong Dai, Tianhao Hu, Peixin Cao, Nan Du, and Xiaolong Li. 2024. Adversarial preference optimization: Enhancing your alignment via rm-llm game. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 3705–3716.

Ganqu Cui, Lifan Yuan, Ning Ding, Guanming Yao, Wei Zhu, Yuan Ni, Guotong Xie, Zhiyuan Liu, and Maosong Sun. 2024. UltraFeedback: Boosting language models with high-quality feedback. In *International Conference on Machine Learning*.

Mucong Ding, Souradip Chakraborty, Vibhu Agrawal, Zora Che, Alec Koppel, Mengdi Wang, Amrit Bedi, and Furong Huang. 2024. Sail: Self-improving efficient online alignment of large language models. *arXiv preprint arXiv:2406.15567*.

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

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

Leo Gao, John Schulman, and Jacob Hilton. 2023. Scaling laws for reward model overoptimization. In *International Conference on Machine Learning*, pages 10835–10866. PMLR.

Mohammad Gheshlaghi Azar, Zhaohan Daniel Guo, Bilal Piot, Remi Munos, Mark Rowland, Michal Valko, and Daniele Calandriello. 2024. A general theoretical paradigm to understand learning from human preferences. In *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics*, volume 238 of *Proceedings of Machine Learning Research*, pages 4447–4455. PMLR.

Jiwoo Hong, Noah Lee, and James Thorne. 2024. Reference-free monolithic preference optimization with odds ratio. *arXiv preprint arXiv:2403.07691*.

AQ Jiang, A Sablayrolles, A Mensch, C Bamford, DS Chaplot, D de las Casas, F Bressand, G Lengyel, G Lample, L Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Andreas Köpf, Yannic Kilcher, Dimitri von Rütte, Sotiris Anagnostidis, Zhi Rui Tam, Keith Stevens, Abdullah Barhoum, Duc Nguyen, Oliver Stanley, Richárd Nagyfi, et al. 2024. Openassistant conversations-democratizing large language model alignment. *Advances in Neural Information Processing Systems*, 36.

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

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

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

Tianqi Liu, Yao Zhao, Rishabh Joshi, Misha Khalman, Mohammad Saleh, Peter J Liu, and Jialu Liu. 2023. Statistical rejection sampling improves preference optimization. *arXiv preprint arXiv:2309.06657*.

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

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.

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

Corby Rosset, Ching-An Cheng, Arindam Mitra, Michael Santacroce, Ahmed Awadallah, and Tengyang Xie. 2024. Direct nash optimization: Teaching language models to self-improve with general preferences. *arXiv preprint arXiv:2404.03715*.

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

Avi Singh, John D Co-Reyes, Rishabh Agarwal, Ankesh Anand, Piyush Patil, Peter J Liu, James Harrison, Jaehoon Lee, Kelvin Xu, Aaron Parisi, et al. 2023. Beyond human data: Scaling self-training for problem-solving with language models. *arXiv preprint arXiv:2312.06585*.

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

Lewis Tunstall, Edward Beeching, Nathan Lambert, Nazneen Rajani, Shengyi Huang, Kashif Rasul, Alvaro Bartolome, Alexander M. Rush, and Thomas Wolf. The Alignment Handbook. `https://github.com/huggingface/alignment-handbook`.

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

Yue Wu, Zhiqing Sun, Huizhuo Yuan, Kaixuan Ji, Yiming Yang, and Quanquan Gu. 2024b. Self-play preference optimization for language model alignment. *arXiv preprint arXiv:2405.00675*.

Tengyang Xie, Dylan J Foster, Akshay Krishnamurthy, Corby Rosset, Ahmed Awadallah, and Alexander Rakhlin. 2024. Exploratory preference optimization: Harnessing implicit q*-approximation for sample-efficient rlhf. *arXiv preprint arXiv:2405.21046*.

Wei Xiong, Hanze Dong, Chenlu Ye, Ziqi Wang, Han Zhong, Heng Ji, Nan Jiang, and Tong Zhang. 2024. Iterative preference learning from human feedback: Bridging theory and practice for rlhf under kl-constraint. In *Forty-first International Conference on Machine Learning*.

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

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

Dan Zhang, Sining Zhoubian, Yisong Yue, Yuxiao Dong, and Jie Tang. 2024a. Rest-mcts*: Llm self-training via process reward guided tree search. *arXiv preprint arXiv:2406.03816*.

Shenao Zhang, Donghan Yu, Hiteshi Sharma, Ziyi Yang, Shuohang Wang, Hany Hassan, and Zhaoran Wang. 2024b. Self-exploring language models: Active preference elicitation for online alignment. *arXiv preprint arXiv:2405.19332*.

Yao Zhao, Rishabh Joshi, Tianqi Liu, Misha Khalman, Mohammad Saleh, and Peter J Liu. 2023. Slic-hf: Sequence likelihood calibration with human feedback. *arXiv preprint arXiv:2305.10425*.

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

Rui Zheng, Wei Shen, Yuan Hua, Wenbin Lai, Shihan Dou, Yuhao Zhou, Zhiheng Xi, Xiao Wang, Haoran Huang, Tao Gui, et al. 2023b. Improving generalization of alignment with human preferences through group invariant learning. *arXiv preprint arXiv:2310.11971*.

A Experiments Details

In our experiments, we use the Alignment Handbook framework (Tunstall et al.) for policy model iteration and the RLHF-Reward-Modeling [1] framework for reward model iteration.

Mutual-Taught training We follow SimPO (Meng et al., 2024) to set the policy sampling and training parameters. Specifically, for policy sampling: temperature is set to 0.8, $K = 5$, and top-p to 0.95. For policy training: learning rate is set to 7×10^{-7}, batch size to 128, and warmup ratio to 0.1. These settings remain consistent across both iterations. For reward model iteration, we use the default parameter settings from RLHF-Reward-Modeling. In both iterations, we use the same settings: learning rate of 2×10^{-6}, batch size of 512, and weight decay of 0.001. Reward model

[1]RLHF-Reward-Modeling at `https://github.com/RLHFlow/RLHF-Reward-Modeling`

training data is constructed by sampling from the policy distributions before and after updates, with a temperature of 0.8 and top-p of 0.95.

Baselines In Offline Preference Optimization Methods, we maintain the same sampling and training parameters as Mutual-Taught. For Iterative Preference Optimization Methods, in iterative DPO, we observed performance degradation in the final iteration with a larger learning rate, so we lowered it to 5×10^{-7}. For SPPO, we use the default training parameters provided by the method. For Meta-rewarding, we first built Evaluation Fine-Tuning (EFT) data from the Open Assistant (Köpf et al., 2024) dataset to boost the initial judgment ability of the model before self-training iterations. During the construction of EFT data, we prompt GPT-4o to generate judgments with high quality instead of the SFT baseline in Yuan et al. (2024). During self-training iterations, we use prompts from the UltraFeedback dataset instead of those generated by LLaMA2-70B-Chat to align with Mutual-Taught.

Length-control To prevent length explosion, we implement a length-control mechanism for selecting preference data. For each prompt, we first select responses with above-average reward scores, then choose the shortest one as the chosen response. The response with the lowest score is selected as the rejected one. This length control mechanism is applied to all experiments except for Meta-Rewarding, where we use the length control mechanism proposed by the original method.

B Performance of Mutual-Taught With Additional Iterations

To evaluate the impact of additional iterations on model performance, we conduct a second round of Mutual-Taught training using the same dataset as in the main experiments. To mitigate overfitting, we regenerate higher-quality preference data based on the models from the first round and reinitialize both the policy model and the reward model from their respective base states. All experimental hyperparameters remain consistent with those used in the main experiments. The experimental results are summarized in Table 5.

It is evident that after the second round of iterations, both the policy model and the reward model exhibit consistent improvements compared to the first round. Notably, the final reward model achieves a superior performance on RewardBench, surpassing GPT-4o-2024-08-06. This demonstrates that Mutual-Taught remains effective even with additional iterations. More specifically, while the starting models in both rounds are identical, the preference data in the second round is derived from models refined during the first round. These higher-quality outputs serve as a stronger foundation for the E-step (policy updates) and M-step (reward model updates), enabling more effective alignment and yielding improved results.

Iteration	AlpacaEval-2 LC Win Rate	Arena-Hard Win Rate	RewardBench Avg Score
Round 1 Iter1	37.1	33.5	85.6
Round 1 Iter2	52.0	37.5	86.8
Round 2 Iter1	39.5	34.8	86.0
Round 2 Iter2	**53.1**	**39.0**	**88.0**

Table 5: Performance metrics across two rounds of Mutual-Taught iterations. Text in bold indicates the best performance.