Reward Shaping to Mitigate Reward Hacking in RLHF

Jiayi Fu [*1] **Xuandong Zhao** [*2] **Chengyuan Yao** [3] **Heng Wang** [3] **Qi Han** [3] **Yanghua Xiao** [1]

Abstract

Reinforcement Learning from Human Feedback (RLHF) is essential for aligning large language models (LLMs) with human values. However, RLHF is susceptible to *reward hacking*, where the agent exploits flaws in the reward function rather than learning the intended behavior, thus degrading alignment. Although reward shaping helps stabilize RLHF and partially mitigate reward hacking, a systematic investigation into shaping techniques and their underlying principles remains lacking. To bridge this gap, we present a comprehensive study of the prevalent reward shaping methods. Our analysis suggests two key design principles: (1) the RL reward should be bounded, and (2) the RL reward benefits from rapid initial growth followed by gradual convergence. Guided by these insights, we propose Preference As Reward (PAR), a novel approach that leverages the latent preferences embedded within the reward model as the signal for reinforcement learning. We evaluated PAR on two base models, Gemma2-2B, and Llama3-8B, using two datasets, Ultrafeedback-Binarized and HH-RLHF. Experimental results demonstrate PAR's superior performance over other reward shaping methods. On the AlpacaEval 2.0 benchmark, PAR achieves a win rate of at least 5 percentage points higher than competing approaches. Furthermore, PAR exhibits remarkable data efficiency, requiring only a single reference reward for optimal performance, and maintains robustness against reward hacking even after two full epochs of training.

1. Introduction

Reinforcement learning from human feedback (RLHF) has become a cornerstone for aligning large language models (LLMs) with human intentions and enhancing their capabilities (Ouyang et al., 2022; OpenAI, 2024; Bai et al., 2022; Guo et al., 2024). However, a significant challenge that undermines the reliability of RLHF is reward hacking: the tendency for policy models to exploit weaknesses in the reward model to maximize reward signals without achieving genuine alignment or improvement (Amodei et al., 2016; Gao et al., 2023; Singhal et al., 2023). This can manifest as degenerate behaviors, such as generating repetitive or overly verbose outputs, merely to satisfy the proxy reward function.

Proximal Policy Optimization (PPO) (Schulman et al., 2017) is widely adopted for RLHF (Ouyang et al., 2022), yet it remains susceptible to reward hacking (Gao et al., 2023). Existing mitigation strategies often involve reward shaping techniques like clipping or rescaling the proxy rewards (Dai et al., 2023; Wang et al., 2024). Despite their use, a systematic investigation comparing these methods and establishing clear design principles for effective reward shaping is currently lacking.

This work aims to fill this gap. We conduct a systematic analysis of reward shaping methods in the context of PPO-based RLHF. Our results reveal a reward threshold in PPO training—exceeding it often triggers reward hacking, degrading the model's win rate (Moskovitz et al., 2023). We hypothesize that excessively high rewards misalign with true performance and impair the critic's learning, leading to our first principle: *(1) RL reward should be bounded*. We further find low-reward regions safer for optimization, motivating our second principle: *(2) RL reward benefits from rapid initial growth followed by gradual convergence*.

Motivated by these principles, we introduce Preference As Reward (PAR), a novel reward shaping technique (see Figure 1). PAR applies a sigmoid function to the centered reward (the difference between the proxy reward r and a reference reward r_{ref}). This design is intuitive: since the policy model is typically initialized from a reference model, the centered reward is initially near zero. The sigmoid function's steep slope at zero promotes rapid initial learning. Crucially, the sigmoid's gradual convergence towards its upper bound ensures training stability. We further observe that PAR's functional form closely resembles the Bradley-Terry model (Bradley & Terry, 1952), interpreting the exponential

[*]Equal contribution [1]Fudan University [2]UC Berkeley [3]StepFun. Correspondence to: Jiayi Fu <fujy22@m.fudan.edu.cn>, Xuandong Zhao <xuandongzhao@berkeley.edu>.

Published at ICML 2025 Workshop on Reliable and Responsible Foundation Models.

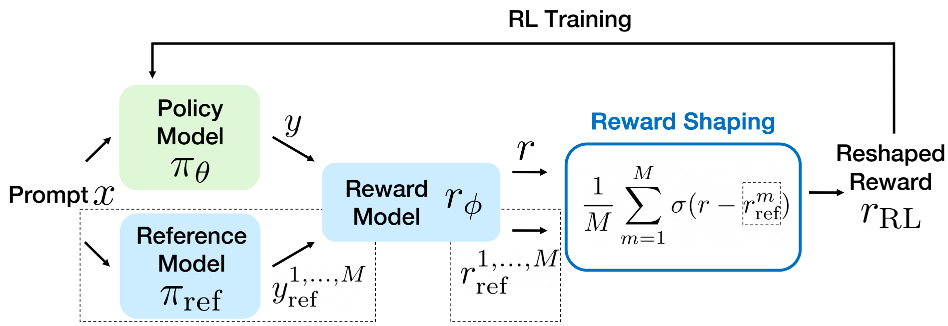


Figure 1. RLHF training pipeline with reward shaping. Policy model's responses are evaluated by the reward model, producing proxy rewards. These rewards are then reshaped (optionally using reference rewards, as shown in the dashed box) before being used to update the policy via RL. The blue box details the PAR reward shaping function, which uses a sigmoid applied to the centered reward.

of the proxy reward as an Elo score (Elo, 1978). In this context, the RL reward $r_{\text{RL}} = \text{sigmoid}(r - r_{\text{ref}})$ can be interpreted as the relative preference of the policy response over the reference response, as determined by the reward model.

We conduct experiments on two base models, Gemma2-2B (Google, 2024) and Llama3-8B (Meta, 2024), using two widely used RLHF datasets, Ultrafeedback-Binarized (Cui et al., 2023) and HH-RLHF (Bai et al., 2022). The result shows that PAR achieves high winrates on the test set after training of one epoch. We also evaluate its performance on two benchmarks AlpacaEval2.0 (Li et al., 2023) and MT-Bench (Zheng et al., 2023a), the PAR consistently tops the benchmark and achieves a winrate that is at least 5 percentage points higher than that of its competitors. Additionally, PAR is data-efficient, requiring only a single reference reward to perform well. It also remains robust against reward hacking, even after two epochs of training.

In conclusion, our contributions are threefold.

- We propose two key principles for designing effective reward shaping strategies.
- We introduce PAR, a novel reward shaping technique, and analyze its connection to the underlying preferences of the reward model.
- We demonstrate through extensive experiments that PAR significantly mitigates reward hacking and outperforms existing baselines across multiple models, datasets, and benchmarks.

2. Related Work

Reward hacking arises when an RL agent exploits flaws or ambiguities in the reward function to achieve high rewards without performing the intended task (Weng, 2024). This aligns with Goodhart's Law: *When a measure becomes a target, it ceases to be a good measure.*

Reward hacking in RLHF for large language models has been extensively studied. Gao et al. (2023) systematically investigate the scaling laws of reward hacking in small models, while Wen et al. (2024) demonstrate that language models can learn to mislead humans through RLHF. Beyond exploiting the training process, reward hacking can also target evaluators. Although using LLMs as judges is a natural choice given their increasing capabilities, this approach is imperfect and can introduce biases. For instance, LLMs may favor their own responses when evaluating outputs from different model families (Liu et al., 2024b; Xu et al., 2024) or exhibit positional bias when assessing responses in sequence (Wang et al., 2023).

To mitigate reward hacking, many methods have been proposed. Reward ensemble techniques have shown promise in addressing this issue (Eisenstein et al., 2023; Ram'e et al., 2024; Ahmed et al., 2024; Coste et al., 2023; Zhang et al., 2024). Miao et al. (2024) introduce an information bottleneck to filter irrelevant noise, while Moskovitz et al. (2023) employ constrained RLHF to prevent reward overoptimization. Chen et al. (2024) propose the ODIN method, which uses a linear layer to separately output quality and length rewards, reducing their correlation through an orthogonal loss function. Similarly, Sun et al. (2023) train instructable reward models to give a more comprehensive reward signal from multiple objectives. Dai et al. (2023) constrain reward magnitudes using regularization terms. Liu et al. (2024a) curate diverse pairwise data for robust reward model training.

Reward shaping methods have demonstrated both simplicity and effectiveness in recent research (Yang et al., 2024; Jinnai et al., 2024). For instance, Wang et al. (2024) introduce a log-sigmoid centering transformation, while Shen et al. (2024) employ contrastive rewards to enhance RL performance. Additionally, Ahmadian et al. (2024) propose a leave-one-out reward method for advantage estimation in the REINFORCE algorithm. However, these reward-shaping

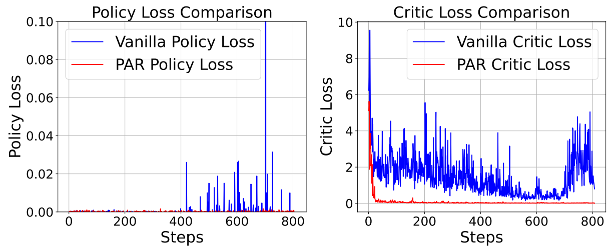


Figure 2. Loss curves from PPO training show that PAR exhibits greater stability, particularly in critic loss, compared to Vanilla training. This stability is attributed to PAR's bounded RL reward.

methods are not explicitly designed to mitigate reward hacking, nor do they provide a theoretical justification for their mechanisms. In contrast, our proposed PAR method directly addresses the challenge of reward hacking, offering both a principled solution and theoretical analysis.

3. Method

3.1. Design Principles

As detailed in Section 1, we restate our two design principles here: (1) RL reward should be bounded, (2) RL reward benefits from rapid initial growth followed by gradual convergence. To elucidate the rationale behind these principles, we examine the Proximal Policy Optimization (PPO) policy and critic loss functions for prompt x and response y (notation detailed in Table 2):

$$\begin{aligned}\mathcal{L}_{\text{policy}}(\theta) = \hat{\mathbb{E}}_t \Bigg[\min \Bigg(&\frac{\pi_\theta(y_t|x, y_{<t})}{\pi_{\theta_{\text{old}}}(y_t|x, y_{<t})} \cdot \hat{A}_t, \\ &\text{clip} \left(\frac{\pi_\theta(y_t|x, y_{<t})}{\pi_{\theta_{\text{old}}}(y_t|x, y_{<t})}, 1-\epsilon, 1+\epsilon \right) \cdot \hat{A}_t \Bigg) \Bigg], \\ \mathcal{L}_{\text{critic}}(\alpha) = \hat{\mathbb{E}}_t \big[&||V_\alpha(x, y_{<t}) - G_t||_2^2 \big].\end{aligned}$$

For the policy loss, $\hat{A}_t = \sum_{l=t}^{T} (\gamma\lambda)^{l-t} \delta_l$ represents the generalized advantage estimation (GAE) at token t, where $\delta_t = r_t + \gamma V_{\alpha_{\text{old}}}(s_{t+1}) - V_{\alpha_{\text{old}}}(s_t)$ is the temporal difference (TD) error. π_θ denotes the current policy model, and $\pi_{\theta_{\text{old}}}$ refers to the policy model from the previous iteration. $V_{\alpha_{\text{old}}}$ is the critic's value function from the previous iteration. For the critic loss, $G_t = \sum_{l=t}^{T} \gamma^{l-t} r_l$ represents the return, defined as the discounted sum of per-token rewards.

The per-token reward at position t, denoted as r_t, is defined as:

$$r_t = \begin{cases} r_{\text{RL}} - \eta \log \frac{\pi_\theta(y_t|x, y_{<t})}{\pi_{\text{ref}}(y_t|x, y_{<t})} & \text{if } t = T \\ -\eta \log \frac{\pi_\theta(y_t|x, y_{<t})}{\pi_{\text{ref}}(y_t|x, y_{<t})} & \text{if } t < T \end{cases}$$

This formulation ensures that the final token receives the RL reward r_{RL} while earlier tokens are shaped by the KL divergence regularization term.

The first principle, advocating for bounded RL rewards, is crucial for stabilizing critic training. Excessively large rewards can hinder the critic model's ability to accurately learn the value function, as illustrated in Figure 2. We hypothesize that this issue arises from the nature of the regression loss used in the critic model. Specifically, large RL rewards r_{RL} lead to large variance of returns G_t (see Theorem 3.1), making the critic loss $\mathcal{L}_{\text{critic}}(\alpha)$ more challenging to optimize. Furthermore, this effect propagates to the excessive advantage estimate $\hat{A}_t$, rendering it unstable and leading to overly aggressive policy updates.

The second principle focuses on regulating the rate of change in the advantage function. A rapid change early in training encourages the policy model to learn quickly, while a slower change toward the end of training helps prevent the policy model from collapsing. We posit that this behavior is due to the advantage function's role in controlling both the direction and magnitude of the policy model's optimization steps.

We explore several candidate functions that meet these criteria, with a focus on sigmoid-like functions. Our choice is motivated by the finding that the sigmoid function minimizes policy gradient variance within $\mathcal{F}$ (see Theorem 3.2). The curves for these candidate functions are illustrated in Figure 4b.

3.2. Preference as Reward

After careful consideration and empirical evaluation, we recommend using the sigmoid function applied to centered rewards as the preferred reward shaping method. The sigmoid function is bounded, has the steepest slope at the initial point (zero), and converges gradually to its upper bound of one. This property makes it particularly suitable for stabilizing the RL training process. Furthermore, our analysis reveals that this shaping approach is intrinsically linked to the hidden preferences encoded within the reward model. The reward model is designed to simulate human preferences, and the RL training process aims to maximize the reward using an RL algorithm. Given a reward model r_ϕ, the hidden preference between two responses y and y' to a prompt x can be expressed as:

$$\mathcal{P}_\phi(y \succ y'|x) = \sigma(r_\phi(x, y) - r_\phi(x, y'))$$

This formulation shows that applying the sigmoid function to centered rewards corresponds precisely to the preference score of the policy response over the reference response. Consequently, we term this method **Preference As Reward (PAR)**, which is defined as follows. To enhance stability, we use multiple M reference rewards:

$$r_{\text{RL}} = \frac{1}{M} \sum_{m=1}^{M} \sigma(r - r_{\text{ref}}^m) = \frac{1}{M} \sum_{m=1}^{M} \mathcal{P}_\phi(y \succ y_{\text{ref}}^m)$$

Our proposed PAR method serves exclusively as a reward shaping technique, which is fundamentally orthogonal to other strategies for mitigating reward hacking, such as robust reward model training (Dai et al., 2023) or the construction of diverse datasets (Liu et al., 2024a).

The pseudo-code for the reward shaping procedure under PAR is detailed in Algorithm 4, while the complete implementation of the Proximal Policy Optimization (PPO) algorithm is provided in Algorithm 1. Additionally, the pipeline for reward shaping is illustrated in Figure 1.

3.3. Theoretical Analysis

We further establish that our proposed PAR method satisfies two key theoretical properties: (1) The return variance is upper bounded and (2) Policy gradient variance is minimized. A rigorous derivation of these guarantees is provided in Appendix 13. These theoretical foundations are essential for ensuring the robustness and reliability of PAR in practical applications.

Theorem 3.1 (Return variance bound)**.** *For any trajectory and discount factor $\gamma \in (0,1)$, the variance of the PAR return G_t^{PAR} is upper-bounded by:*

$$\mathrm{Var}\big[G_t^{PAR}\big] \leq \frac{1}{(1-\gamma)^2}.$$

Conversely, if the per-token reward is sub-Gaussian with parameter σ^2, the variance of the unbounded return scales as

$$\mathrm{Var}[G_t] = \Omega\big(\sigma^2/(1-\gamma^2)\big)\,.$$

Theorem 3.1 establishes that bounded RL rewards contribute to stabilizing critic model training by constraining the variance of returns.

Theorem 3.2 (Sigmoid minimize the policy gradient variance)**.** *Let $\mathcal{F}$ be the family of C^1, strictly increasing functions $f : \mathbb{R} \to (0,1)$. Then we have:*

$$\sigma = \arg\min_{f\in\mathcal{F}} Var[g_f].$$

Where $g_f = \nabla_\theta \log \pi_\theta(y \mid x)\, f(z)$, σ is the sigmoid function.

Theorem 3.2 justifies our selection of sigmoid-like functions for bounding RL rewards, as the sigmoid's properties minimize policy gradient variance.

4. Experiment

Our emperical analysis is structured to first validate the two key design principles, followed by a comparison of PAR with other reward hacking mitigation methods, and finally, an evaluation of the data efficiency and robustness of PAR.

4.1. Experimental Setting

Datasets and Models We utilize two dialogue datasets: HH-RLHF (Bai et al., 2022) and Ultrafeedback-Binarized (Cui et al., 2023), alongside two base models, Gemma-2B (Google, 2024) and Llama3-8B (Meta, 2024), for our experiments. We present the results of Gemma2-2B on the Ultrafeedback-Binarized in this section. For additional results and comprehensive training details, please refer to Appendix 8.

Mitigation Baselines We evaluate seven baseline methods to mitigate reward hacking, which are described as follows:

- **WARM** (Ram'e et al., 2024): This approach combines the weights of multiple reward models and employs the aggregated model to provide rewards for reinforcement learning training.
- **ODIN** (Chen et al., 2024): This method introduces an additional length head during reward training to capture the response length. Only the quality head is utilized for reinforcement learning training.
- **Reg** (Dai et al., 2023): A regularization term is integrated into the reward training loss, defined as: $l_{\text{reward}} = \mathbb{E}_{(x,y_w,y_l)\sim D}[-\log\sigma(r_\phi(x,y_w) - r_\phi(x,y_l)) + \beta||r_\phi(x,y_w)||_2^2 + \beta||r_\phi(x,y_l)||_2^2]$.
- **Meanstd**: The reward is normalized using the running mean and running standard deviation: $r_{\text{RL}} = \frac{r-\mu}{s}$, where μ and s represent the running mean and standard deviation, respectively.
- **Clip**: Clips reward based on the running mean and standard deviation: $r_{\text{RL}} = \text{clip}(r, \mu - s, \mu + s)$.
- **Minmax**: The reward is normalized using the running minimum and maximum rewards: $r_{\text{RL}} = \frac{r-r_{\min}}{r_{\max}-r_{\min}}$, where $r_{\max}$ and $r_{\min}$ denote the running maximum and minimum rewards, respectively.
- **LSC** (Wang et al., 2024): The reward is normalized using the log-sigmoid-centered shaping method, defined as: $r_{\text{RL}} = \log\sigma(r - r_{\text{ref}}^{.85})$ where $r_{\text{ref}}^{.85}$ represents the 85th percentile of the normal distribution, calculated from the mean and variance of the reference rewards.

Evaluation Metrics Two primary metrics are employed to monitor training progress, both computed on the test set: Proxy Reward (shown as a solid line) and Winrate (shown as a dashed line). The winrate measures the policy model's winning rate against the SFT model, as evaluated by DeepSeek-V3 (DeepSeek-AI, 2024). For the benchmarks AlpacaEval2.0 (Li et al., 2023) and MT-Bench (Zheng et al., 2023a), six metrics are utilized, with all metrics except the length metric being assessed by DeepSeek-V3.

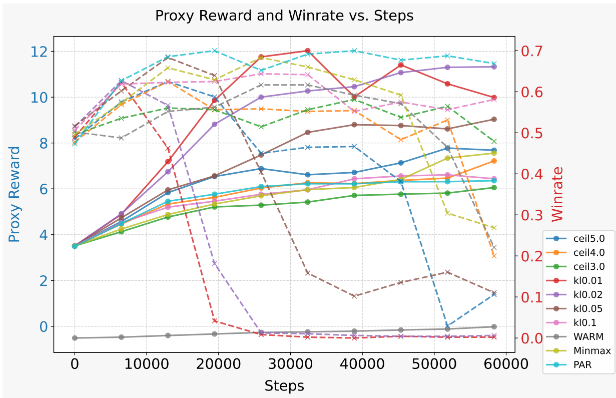
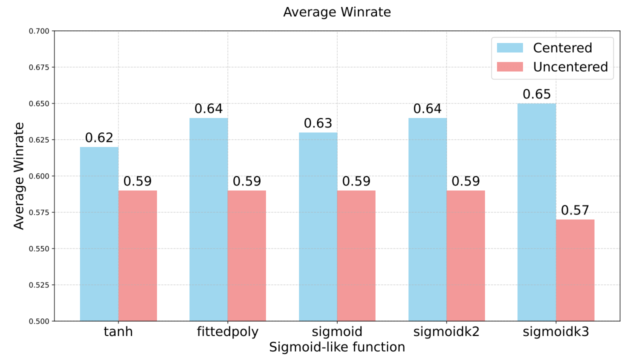


Figure 3. PPO training curves over two epochs. ‘ceil5.0’ indicates that $r_{\text{RL}} = \min(r, 5.0)$, and ‘kl0.1’ refers to the KL penalty with $\beta = 0.1$. This figure indicates two important results: (1) Excessive rewards can cause reward hacking, hence the RL reward should be bounded. (2) PAR is more robust than Minmax and WARM.

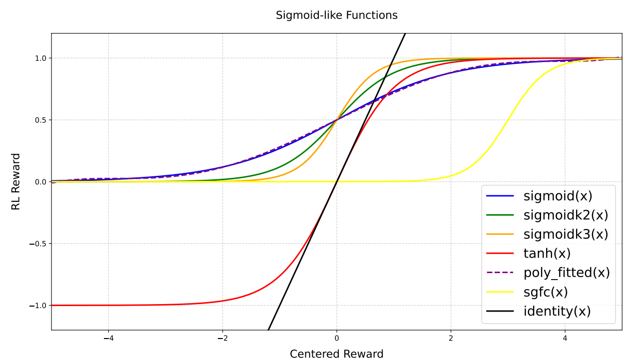
Training Details We briefly outline the training details here; for a comprehensive discussion, refer to Appendix 8. The dataset is preprocessed to remove noise, and hyperparameters are carefully tuned to ensure continuous growth in the proxy reward. The SFT model is trained for two epochs on chosen responses with a learning rate of 5e-6, while the reward model, consisting of a linear head appended to the base model, is trained for one epoch with a learning rate of 5e-6. The policy model, initialized as the SFT model, is trained for one epoch with a learning rate of 3e-7, and the critic model, initialized as the reward model, is trained for one epoch with a learning rate of 5e-6. A linear learning rate scheduler is employed for all training procedures, gradually increasing the learning rate from 0 to the maximum value during the first 0.1 epoch. To generate the reward and winrate curves, the policy model is evaluated on the test set at intervals of 0.1 epochs, yielding 10 checkpoints for each mitigation method.

4.2. Principle One

To validate the first principle that *RL reward should be bounded*, we conducted experiments by employing a larger KL penalty coefficient and constraining the maximum reward during reinforcement learning training (see Figure 3). The results demonstrate that limiting excessive rewards significantly mitigates reward hacking. For instance, increasing the KL penalty coefficient from 0.01 to 0.1 leads to a rise in the winrate curve and a corresponding decline in the reward curve. A similar effect is observed when reducing the reward ceiling (i.e., the maximum reward threshold). Furthermore, Figure 3 reveals that while PAR and kl0.1 exhibit comparable proxy rewards, PAR consistently outperforms kl0.1 in terms of winrate, highlighting the superiority of our proposed PAR method.



(a)



(b)

Figure 4. (a) Performance comparison of sigmoid-like functions. ‘tanh(centered)’ denotes $r_{\text{RL}} = \frac{1}{M}\sum_{m=1}^{M}\tanh(r - r_{\text{ref}}^{m})$, ‘tanh(uncentered)’ denotes $r_{\text{RL}} = \tanh(r)$, and ‘sigmoid(centered)’ represents our PAR method. Centered reward formulations achieve higher winrates than uncentered versions. (b) Mathematical formulations of sigmoid-like functions: $\sigma_k(x) = \frac{1}{1+e^{-kx}}$ (k=2,3 for sigmoidk2/3), fifth-order polynomial approximation (poly_fitted), and shifted sigmoid (sgfc = sigmoidk3(x-3)).

4.3. Principle Two

To validate the second principle—which states that *RL reward benefits from rapid initial growth followed by gradual convergence*—we conducted experiments using several sigmoid-like functions, including their centered and uncentered variants. The results are presented in Figure 4a.

Our experiments show that applying sigmoid-like functions to centered rewards leads to higher win rates compared to uncentered rewards, supporting Principle 2 (steepest slope at initialization). Since centered rewards begin at zero—where the sigmoid’s gradient is maximized—they enable rapid early learning, whereas uncentered rewards start at arbitrary values. Notably, the Slow-Grow-Fast-Converge (SgFc) function, when applied to centered rewards, exhibits lower initial win rates and reward hacking in later phases (see Figure 5). This behavior arises from SgFc’s diminished early gradients and abrupt convergence, further validating Principle 2, which emphasizes the need for rapid initial growth followed by gradual convergence.

4.4. PAR Effectively Mitigates Reward Hacking

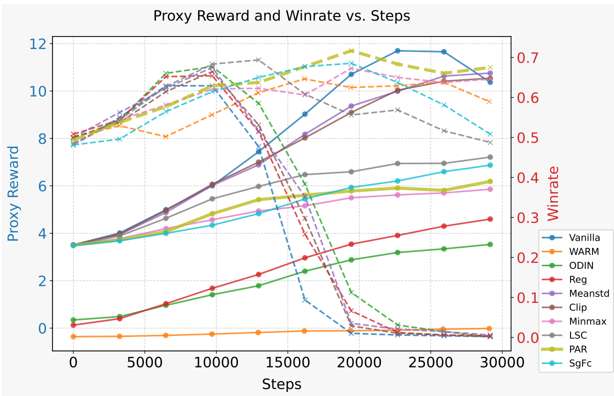
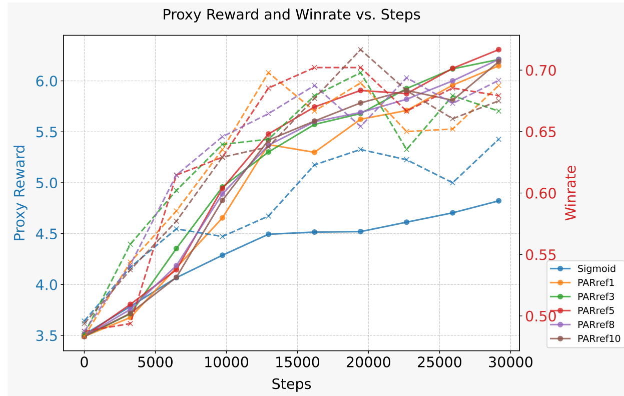


Figure 5. PPO training curve for different mitigation methods on Gemma2-2B and Ultrafeedback-Binarized. Solid lines denote the Proxy Reward, and dashed lines denote the Winrate. Vanilla PPO demonstrates significant reward hacking. ODIN, Reg, Meanstd, Clip, and LSC fail to mitigate this issue, indicated by increasing proxy rewards but decreasing winrates. PAR achieves highest winrate at the end of training.

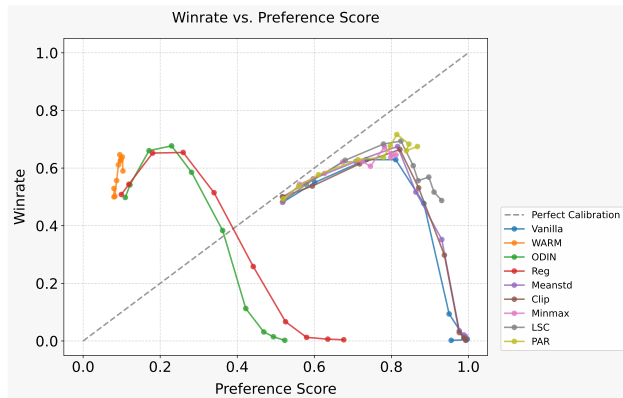
Reward and Winrate Curve As illustrated in Figure 5, the Vanilla PPO suffers from the reward hacking problem severely. To address this issue, we conduct a comprehensive study of several mitigation methods. While some approaches, such as ODIN, Reg, Meanstd, Clip, and LSC, fail to mitigate the problem, others, including WARM, Minmax, and PAR, demonstrate varying degrees of effectiveness over a single training epoch. Notably, the PAR method achieves the highest winrate by the end of the training process.

Another intriguing observation is that Vanilla, Meanstd, Clip, and LSC exhibit hacking behavior when the proxy reward reaches a specific threshold, such as 6.0, as shown in Figure 5. In contrast, Minmax and PAR show no signs of hacking, and their proxy rewards do not exceed this threshold.

Benchmark Performance We further investigate the generalization ability of the policy model on out-of-distribution (OOD) data. For each mitigation method, we select the checkpoint after one epoch of training and evaluate these checkpoints on two benchmarks: AlpacaEval2.0 and MT-bench. The results, presented in Table 1, align with the training curve depicted in Figure 5. The Vanilla PPO method exhibits complete deterioration, while the top-performing methods are PAR, Minmax, and WARM.



(a)



(b)

Figure 6. (a)PPO training curves, evaluated across varying numbers of reference rewards for the PAR method. E.g., the PAR5 means $r_{\text{RL}} = \frac{1}{5}\sum_{m=1}^{5} \sigma(r - r_{\text{ref}}^{m})$. A single reference reward is sufficient for PAR to achieve a comparable winrate. (b) Calibration between hidden preference score (reward model) and winrate (DeepSeek-V3) for different mitigation methods. All reward shaping methods show initial alignment but sudden winrate decrease when preference score exceeds 0.8, while PAR resists this decrease. Methods modifying the reward model directly show no calibration.

4.5. Data Efficiency and Robustness

The default number of reference rewards for each prompt in our PAR method is set to 10. However, we hypothesize that this number may be higher than necessary for PAR to function effectively. To explore this, we conduct an experiment to determine the minimum number of reference rewards required for PAR to perform efficiently. As shown in Figure 6a, the results reveal that PARref1 to PARref10 exhibit similar trends in both proxy reward and winrate during training. This suggests that a single reference reward is sufficient for PAR to operate effectively. In contrast, the sigmoid method, which can be viewed as a variant of PAR without any reference rewards, performs significantly worse than PARref1. This indicates that completely eliminating reference rewards is not feasible for maintaining

	Method	AlpacaEval2.0			MT-Bench		
		LC Winrate(%) ↑	Winrate(%) ↑	Length ↓	T1 ↑	T2 ↑	Overall ↑
	SFT	50.000	50.000	**899**	5.150	3.975	4.563
PPO training	Vanilla	0.100	0.370	2008	2.150	1.700	1.925
PPO training	WARM	60.670	63.170	1073	5.525	3.938	4.731
PPO training	ODIN	0.000	0.000	3672	1.375	1.338	1.356
PPO training	Reg	0.000	0.000	1868	1.513	1.388	1.450
PPO training	Meanstd	0.030	0.120	3183	1.713	1.300	1.506
PPO training	Clip	0.000	0.000	3096	1.288	1.225	1.256
PPO training	Minmax	66.980	70.930	1159	5.750	4.013	4.881
PPO training	LSC	47.560	53.790	1556	5.538	4.100	4.819
PPO training	PAR	**70.810**	**75.370**	1207	**5.813**	**4.313**	**5.063**

Table 1. In our evaluation, the checkpoint after one epoch of PPO training is selected for comparison, while the SFT model checkpoint is chosen after two epochs of training. The results indicate that PAR consistently achieves superior performance across all benchmark metrics.

performance.

To assess the robustness of the mitigation methods discussed earlier, we select the top three performing methods on benchmarks: PAR, Minmax, and WARM. For a more comprehensive evaluation, we extend the training process to two epochs instead of one. The rationale is that if a mitigation method can effectively address the reward hacking problem even under prolonged training, it can be considered robust. The training curves for proxy reward and winrate are presented in Figure 3. Among the three methods, it is evident that Minmax and WARM lack robustness when the training process is extended to two epochs. In contrast, PAR demonstrates consistent robustness throughout the extended training period. Notably, PAR consistently achieves the highest winrate among all methods, further highlighting its effectiveness and reliability in mitigating reward hacking over extended training durations.

4.6. Calibration of Preference Score

We also investigate the calibration between the hidden preference score of the reward model and the winrate provided by DeepSeek-V3 (see Figure 6b). For all reward shaping methods, the preference score initially calibrates well with the winrate but deteriorates when the preference score exceeds 0.8. Notably, PAR effectively resists this deterioration by limiting the preference score. In contrast, methods that modify the reward model itself exhibit poor calibration, rendering their results less meaningful.

5. Discussion

Reward shaping is not applicable to DPO (Rafailov et al., 2023), as it does not require a reward model during training. We also explore online DPO, which employs the policy model to generate two responses, and the reward model selects the response with the higher reward as the chosen response and the lower reward as the rejected response. However, since most reward shaping techniques are monotonic, they do not alter the binary preference and therefore, they do not influence the training procedure of online DPO.

For GRPO (Shao et al., 2024), we argue that its advantage calculation inherently normalizes the proxy reward, making linear transformations (e.g., Minmax and mean_std) ineffective. However, our non-linear PAR demonstrates slightly better performance than Vanilla GRPO in later stages (see Appendix 11). An important observation is that GRPO does not exhibit the reward hacking problem during training, primarily because its advantage calculation effectively normalizes the rewards. Although the win rate decreases in the later stages, the proxy rewards also decrease proportionally, maintaining alignment between the optimization objective and the desired outcomes.

6. Conclusion

We identify that for a given reward model, there exists a specific threshold beyond which the proxy reward becomes both meaningless and inaccurate. Based on this observation, we establish two fundamental principles for designing reward shaping methods.

In alignment with these principles, we propose an effective shaping method, Preference As Reward (PAR). Through extensive experimentation with various mitigation approaches, our results demonstrate that PAR not only outperforms other baseline methods by the end of one training epoch but also maintains a high winrate after two epochs of training. Notably, PAR is also data-efficient, requiring only a single reference reward to achieve strong performance.

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, 2024. URL https://arxiv.org/abs/2402.14740.

Ahmed, A. M., Rafailov, R., Sharkov, S., Li, X., and Koyejo, S. Scalable ensembling for mitigating reward overoptimisation, 2024. URL https://arxiv.org/abs/2406.01013.

Amodei, D., Olah, C., Steinhardt, J., Christiano, P., Schulman, J., and Mané, D. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.

Bai, Y., Jones, A., Ndousse, K., Askell, A., Chen, A., DasSarma, N., Drain, D., Fort, S., Ganguli, D., Henighan, T., Joseph, N., Kadavath, S., Kernion, J., Conerly, T., El-Showk, S., Elhage, N., Hatfield-Dodds, Z., Hernandez, D., Hume, T., Johnston, S., Kravec, S., Lovitt, L., Nanda, N., Olsson, C., Amodei, D., Brown, T., Clark, J., McCandlish, S., Olah, C., Mann, B., and Kaplan, J. Training a helpful and harmless assistant with reinforcement learning from human feedback, 2022. URL https://arxiv.org/abs/2204.05862.

Bradley, R. A. and Terry, M. E. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, pp. 324, 1952. doi: 10.2307/2334029. URL http://dx.doi.org/10.2307/2334029.

Chen, L., Zhu, C., Soselia, D., Chen, J., Zhou, T., Goldstein, T., Huang, H., Shoeybi, M., and Catanzaro, B. Odin: Disentangled reward mitigates hacking in rlhf. *ArXiv*, abs/2402.07319, 2024. URL https://api.semanticscholar.org/CorpusID:267627509.

Coste, T., Anwar, U., Kirk, R., and Krueger, D. Reward model ensembles help mitigate overoptimization. *arXiv e-prints*, pp. arXiv–2310, 2023.

Cui, G., Yuan, L., Ding, N., Yao, G., Zhu, W., Ni, Y., Xie, G., Liu, Z., and Sun, M. Ultrafeedback: Boosting language models with high-quality feedback, 2023.

Dai, J., Pan, X., Sun, R., Ji, J., Xu, X., Liu, M., Wang, Y., and Yang, Y. Safe rlhf: Safe reinforcement learning from human feedback. *ArXiv*, abs/2310.12773, 2023. URL https://api.semanticscholar.org/CorpusID:264306078.

DeepSeek-AI. Deepseek-v3 technical report, 2024. URL https://arxiv.org/abs/2412.19437.

Eisenstein, J., Nagpal, C., Agarwal, A., Beirami, A., D'Amour, A., Dvijotham, D., Fisch, A., Heller, K., Pfohl, S. R., Ramachandran, D., Shaw, P., and Berant, J. Helping or herding? reward model ensembles mitigate but do not eliminate reward hacking. *ArXiv*, abs/2312.09244, 2023. URL https://api.semanticscholar.org/CorpusID:266210056.

Elo, A. The rating of chessplayers, past and present. Jan 1978.

Eysenbach, B., Salakhutdinov, R., and Levine, S. Search on the replay buffer: Bridging planning and reinforcement learning, 2019. URL https://arxiv.org/abs/1906.05253.

Gao, L., Schulman, J., and Hilton, J. Scaling laws for reward model overoptimization. In *International Conference on Machine Learning*, pp. 10835–10866. PMLR, 2023.

Google. Gemma. 2024. URL https://www.kaggle.com/models/google/gemma. Accessed: 2025-02-24.

Guo, D., Zhu, Q., Yang, D., Xie, Z., Dong, K., Zhang, W., Chen, G., Bi, X., Wu, Y., Li, Y. K., Luo, F., Xiong, Y., and Liang, W. Deepseek-coder: When the large language model meets programming – the rise of code intelligence, 2024. URL https://arxiv.org/abs/2401.14196.

Jinnai, Y., Morimura, T., Ariu, K., and Abe, K. Regularized best-of-n sampling to mitigate reward hacking for language model alignment, 2024. URL https://arxiv.org/abs/2404.01054.

Li, X., Zhang, T., Dubois, Y., Taori, R., Gulrajani, I., Guestrin, C., Liang, P., and Hashimoto, T. B. Alpacaeval: An automatic evaluator of instruction-following models. https://github.com/tatsu-lab/alpaca_eval, 5 2023.

Liu, T., Xiong, W., Ren, J., Chen, L., Wu, J., Joshi, R., Gao, Y., Shen, J., Qin, Z., Yu, T., Sohn, D., Makarova, A., Liu, J., Liu, Y., Piot, B., Ittycheriah, A., Kumar, A., and Saleh, M. Rrm: Robust reward model training mitigates reward hacking, 2024a. URL https://arxiv.org/abs/2409.13156.

Liu, Y., Moosavi, N. S., and Lin, C. Llms as narcissistic evaluators: When ego inflates evaluation scores, 2024b. URL https://arxiv.org/abs/2311.09766.

Meta. Llama 3 model card. 2024. URL https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md. Accessed: 2025-02-24.

Miao, Y., Zhang, S., Ding, L., Bao, R., Zhang, L., and Tao, D. Inform: Mitigating reward hacking in rlhf via information-theoretic reward modeling, 2024. URL https://arxiv.org/abs/2402.09345.

Moskovitz, T., Singh, A. K., Strouse, D., Sandholm, T., Salakhutdinov, R., Dragan, A. D., and McAleer, S. Confronting reward model overoptimization with constrained rlhf, 2023. URL https://arxiv.org/abs/2310.04373.

OpenAI. Gpt-4 technical report, 2024. URL https://arxiv.org/abs/2303.08774.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P. F., Leike, J., and Lowe, R. Training language models to follow instructions with human feedback. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 27730–27744. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/b1efde53be364a73914f58805a001731-Paper-Conference.pdf.

Patterson, A., Liao, V., and White, M. Robust losses for learning value functions, 2023. URL https://arxiv.org/abs/2205.08464.

Rafailov, R., Sharma, A., Mitchell, E., Manning, C. D., Ermon, S., and Finn, C. Direct preference optimization: Your language model is secretly a reward model. In Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., and Levine, S. (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 53728–53741. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/a85b405ed65c6477a4fe8302b5e06ce7-Paper-Conference.pdf.

Ram'e, A., Vieillard, N., Hussenot, L., Dadashi, R., Cideron, G., Bachem, O., and Ferret, J. Warm: On the benefits of weight averaged reward models. *ArXiv*, abs/2401.12187, 2024. URL https://api.semanticscholar.org/CorpusID:267068615.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *ArXiv*, abs/1707.06347, 2017. URL https://api.semanticscholar.org/CorpusID:28695052.

Shao, Z., Wang, P., Zhu, Q., Xu, R., Song, J., Bi, X., Zhang, H., Zhang, M., Li, Y. K., Wu, Y., and Guo, D. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Shen, W., Zhang, X., Yao, Y., Zheng, R., Guo, H., and Liu, Y. Improving reinforcement learning from human feedback using contrastive rewards, 2024. URL https://arxiv.org/abs/2403.07708.

Singhal, P., Goyal, T., Xu, J., and Durrett, G. A long way to go: Investigating length correlations in rlhf. *arXiv preprint arXiv:2310.03716*, 2023.

Sun, Z., Shen, Y., Zhang, H., Zhou, Q., Chen, Z., Cox, D. D., Yang, Y., and Gan, C. Salmon: Self-alignment with principle-following reward models. *CoRR*, 2023.

Wang, P., Li, L., Chen, L., Cai, Z., Zhu, D., Lin, B., Cao, Y., Liu, Q., Liu, T., and Sui, Z. Large language models are not fair evaluators, 2023. URL https://arxiv.org/abs/2305.17926.

Wang, Z., Nagpal, C., Berant, J., Eisenstein, J., D'Amour, A., Koyejo, O., and Veitch, V. Transforming and combining rewards for aligning large language models. *ArXiv*, abs/2402.00742, 2024. URL https://api.semanticscholar.org/CorpusID:267365201.

Wen, J., Zhong, R., Khan, A., Perez, E., Steinhardt, J., Huang, M., Bowman, S. R., He, H., and Feng, S. Language models learn to mislead humans via rlhf, 2024. URL https://arxiv.org/abs/2409.12822.

Weng, L. Reward hacking in reinforcement learning. *lilianweng.github.io*, Nov 2024. URL https://lilianweng.github.io/posts/2024-11-28-reward-hacking/.

Xu, W., Zhu, G., Zhao, X., Pan, L., Li, L., and Wang, W. Y. Pride and prejudice: Llm amplifies self-bias in self-refinement. *arXiv preprint arXiv:2402.11436*, 2024.

Yang, R., Ding, R., Lin, Y., Zhang, H., and Zhang, T. Regularizing hidden states enables learning generalizable reward model for llms, 2024. URL https://arxiv.org/abs/2406.10216.

Zhang, S., Chen, Z., Chen, S., Shen, Y., Sun, Z., and Gan, C. Improving reinforcement learning from human feedback with efficient reward model ensemble, 2024. URL https://arxiv.org/abs/2401.16635.

Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E. P., Zhang, H., Gonzalez, J. E., and Stoica, I. Judging llm-as-a-judge with mt-bench and chatbot arena, 2023a. URL https://arxiv.org/abs/2306.05685.

Zheng, R., Dou, S., Gao, S., Hua, Y., Shen, W., Wang, B., Liu, Y., Jin, S., Liu, Q., Zhou, Y., Xiong, L., Chen, L., Xi, Z., Xu, N., Lai, W., Zhu, M., Chang, C., Yin, Z., Weng, R., Cheng, W., Huang, H., Sun, T., Yan, H., Gui, T., Zhang, Q., Qiu, X., and Huang, X. Secrets of rlhf in large language models part i: Ppo, 2023b. URL https://arxiv.org/abs/2307.04964.

7. Notations

The definitions of the notations used in this paper are summarized in Table 2.

8. Training Details

Dataset Our experiments are conducted on two datasets: Ultrafeedback-Binarized (Cui et al., 2023) and the helpful-base subset of HH-rlhf (Bai et al., 2022). Both datasets undergo preprocessing to eliminate noise and constrain their overall length. For the Ultrafeedback-Binarized dataset, we select examples where the prompt length, chosen response length, and rejected response length are each less than 512 tokens. Additionally, we ensure that the chosen response score exceeds the rejected response score and that the substring 'confidence' does not appear in either the chosen or rejected responses. For the HH-rlhf dataset, we apply the same length constraints (prompt, chosen, and rejected responses each under 512 tokens). Furthermore, we ensure that each prompt appears only once across both datasets and limit the test set to 256 examples. The training set of Ultrafeedback-Binarized contains around 33,000 examples and HH-RLHF helpful base contains 43,000 examples. All training are carried on 8*A800(80G) GPUs.

Base Models For the base models, we utilize Gemma-2B (Google, 2024) and Llama3-8B (Meta, 2024). In all training procedures, we implement a linear learning rate scheduler, which gradually increases the learning rate from 0 to the maximum value over the first 0.1 epoch.

SFT Model The Supervised Fine-Tuned (SFT) model is initialized from the base model and trained on the chosen responses for two epochs with a learning rate of 5e-6. Gradient norm clipping is applied when the norm exceeds 10.

Reward Model The reward model is initialized from the base model, with the logit head replaced by a linear head above the last embedding layer to output a scalar value. It is trained for one epoch with a learning rate of 5e-6, achieving an accuracy of approximately 70% on the test set. Gradient norm clipping is applied when the norm exceeds 5.

For ODIN training, we use two linear heads to output length reward and quality reward separately, following the training loss described in Chen et al. (2024). Only the quality head is used during RL training.

For WARM training, we train five reward models on the same dataset with varying learning rates (3e-6, 4e-6, 5e-6, 6e-6, 7e-6) and different random seeds.

For Reg training, we adopt the loss function from (Dai et al., 2023), with a regularization term coefficient of 0.005.

Policy Model The policy model is initialized from the SFT model and trained on the same prompts for one epoch using the PPO algorithm with a learning rate of 3e-7. Gradient norm clipping is applied when the norm exceeds 5.

Critic Model The critic model is initialized from the reward model and trained alongside the policy model for one epoch with a learning rate of 5e-6. Gradient norm clipping is applied when the norm exceeds 5.

Hyper-Parameters Responses are sampled from the policy model using a temperature of 0.9, with top-k set to 50, top-p set to 0.9, and a length penalty of 2. The coefficient for the KL penalty is 0.005, and the default number of reference rewards is 10. For PPO training, the buffer size is set to 4, with $\epsilon = 0.2, \lambda = 0.95, \gamma = 1.0$, For GRPO training, the $\epsilon = 0.2$, the buffer size is 4, and the group size is 5.

9. Evaluation

9.1. Winrate on Test Set

To leverage the strong grading capability of DeepSeek-V3 for comparing the SFT model and the policy model on the test set, we design a detailed evaluation prompt. The system prompt and user input format are provided in Listing 1 and 2.

To address position bias (Wang et al., 2023), we evaluate each pair of responses twice, alternating their order, and aggregate the scores. Specifically, for two responses A and B, we first evaluate them in the order A-B and then in the order B-A. In each evaluation, the winner receives a score of 1, the loser receives 0, and in the case of a tie, both responses receive 0.5. The final scores of A and B are compared, and the response with the higher score is declared the winner. If the scores are tied, both responses receive 0.5 win counts. The win counts are used to calculate the winrate.

9.2. Benchmark

We also evaluate the model on two benchmarks, using DeepSeek-V3 to simulate human evaluation. The metrics and their meanings are as follows:

AlpacaEval 2.0

- **LC Winrate**: The length-controlled win rate measures the model's performance while controlling for the length of generated responses. It compares the model's outputs to a baseline (e.g., the SFT model) and adjusts for the influence of response length on human preferences.
- **Winrate**: The standard win rate measures the proportion of times the model's outputs are preferred over the baseline's outputs in human evaluations.

Symbol	Meaning
$\mathcal{D}$	Dataset
$x, y_w, y_l \sim \mathcal{D}$	Prompt, chosen response, rejected response in Dataset
π_θ	Policy model
π_{ref}	Reference model, also the SFT model
r_ϕ	Reward model
V_α	Critic model
$y \sim \pi_\theta(.\|x)$	The response generated by policy model for prompt x
$y_{\text{ref}} \sim \pi_{\text{ref}}(.\|x)$	Reference response, the response generated by reference model
$r = r_\phi(x, y)$	Proxy reward, the reward given directly by reward model
$r_{\text{ref}} = r_\phi(x, y_{\text{ref}})$	Reference reward, the proxy reward for reference response
$\mathcal{P}_\phi(y \succ y_{\text{ref}}\|x) = \text{sigmoid}(r - r_{\text{ref}})$	The hidden preference of reward model r_ϕ
$r_{\text{centered}} = r - r_{\text{ref}}$	Centered reward, the proxy reward subtracted by reference reward.
$r_{\text{RL}} = f(r_{\text{centered}})$	RL reward, the reward for RL training
$s_t = [x, y_{1,\dots,t}]$	The state at position t
$a_t = y_{t+1}$	The Action taken at position t
$\hat{A}_t = \sum_{l=t}^{T} (\gamma\lambda)^{l-t}\delta_l$	The generalized advantage estimation (GAE)
$\delta_t = r_t + \gamma V_{\alpha_{\text{old}}}(s_{t+1}) - V_{\alpha_{\text{old}}}(s_t)$	The temporal difference (TD) error
$G_t = \sum_{l=t}^{T} \gamma^{l-t} r_l$	The return
$r_t = \begin{cases} r_{\text{RL}} - \eta \log \frac{\pi_\theta(y_t\|x,y_{<t})}{\pi_{\text{ref}}(y_t\|x,y_{<t})} & \text{if } t = T \\ -\eta \log \frac{\pi_\theta(y_t\|x,y_{<t})}{\pi_{\text{ref}}(y_t\|x,y_{<t})} & \text{if } t < T \end{cases}$	The per token reward

Table 2. Summary of notations.

- **Length**: The average length of the model's generated responses, measured in tokens or characters, providing insight into the model's verbosity.

MT-bench

- **T1**: Turn 1 Score evaluates the model's performance on the first turn of a multi-turn dialogue, assessing relevance, coherence, and informativeness. Scores are normalized as 0-10.
- **T2**: Turn 2 Score evaluates the model's performance on the second turn, measuring its ability to maintain context and provide consistent, high-quality responses. Scores are also normalized as 0-10.
- **Overall**: The overall score is the average of the T1 and T2 scores, providing a comprehensive evaluation of the model's performance across both turns.

10. More Results

10.1. Llama3-8B and Ultrafeedback Binarized

Figure 8a presents the PPO training curves for different mitigation methods on Llama3-8B with the Ultrafeedback Binarized dataset. PAR demonstrates robustness against reward hacking and maintains a high win rate throughout one epoch of training.

10.2. Gemma2-2B and HH-RLHF

The PPO training curves for various mitigation methods on Gemma2-2B with the HH-RLHF dataset are shown in Figure 8b. PAR exhibits resilience to reward hacking and sustains a high win rate during one epoch of training.

10.3. Llama3-8B and HH-RLHF

Figure 8c illustrates the PPO training curves for different mitigation methods applied to Llama3-8B on the HH-RLHF dataset. While PAR shows signs of reward hacking toward the end of training, it maintains a consistently high win rate (above 60%) for an extended period, from 10,000 to 30,000 steps. We hypothesize that the observed reward hacking in the later stages is due to the convergence rate of the sigmoid function approaching its upper bound. However, the PAR method remains among the top three performers despite showing some performance degradation in later stages.

10.4. Performance on Benchmark

We additionally evaluate the checkpoint with the highest win rate obtained during PPO training on Gemma-2B and the UltraFeedback-Binarized dataset across the benchmark. The corresponding results are presented in Table 3.

Our reward-shaping technique mitigates reward hacking by realigning the agent's incentives with the true task objectives, removing loopholes that allow high rewards for undesired behavior. However, it does not improve peak performance because it does not alter the fundamental capabilities of the agent or the complexity of the task. The best possible policy under the shaped rewards is the same as under the original rewards—we've only made it harder for the agent to find suboptimal shortcuts.

	Method	AlpacaEval2.0	MT-Bench
		LC Winrate(%) ↑	Overall ↑
	SFT	50.00	4.56
PPO training	Vanilla	70.48	4.94
	WARM	70.03	4.83
	ODIN	68.96	5.06
	Reg	69.44	4.74
	Meanstd	69.88	4.90
	Clip	70.55	4.92
	Minmax	68.95	4.81
	LSC	72.24	4.89
	PAR	69.43	4.93

Table 3. For comparison, we select the checkpoint with the highest win rate on the test set within one epoch of PPO training. For the SFT model, we utilize the checkpoint obtained after two epochs of training. All methods exhibit comparable peak performance during the training process.

11. DPO and GRPO

In this section, we explain why monotonous reward shaping techniques, such as PAR, are not applicable to the Direct Preference Optimization (DPO). And why linear shaping techniques are not applicable to the Group Relative Policy Optimization (GRPO) algorithms.

11.1. DPO and Reward Shaping

Vanilla DPO is an offline alignment algorithm that trains the policy model directly on paired responses using a contrastive loss. Since the vanilla DPO algorithm does not rely on an explicit reward model, reward shaping techniques are inherently inapplicable. We also explore an online variant of DPO, which generates two responses for a given prompt and employs a reward model to determine the chosen and rejected responses. The policy model is then trained on these responses (see Algorithm 6). However, any monotonous transformation of the proxy reward will not alter the chosen and rejected responses. For instance, if $r_1 > r_2$, then $f(r_1) > f(r_2)$ for any monotonous function $f(\cdot)$, including PAR. Consequently, PAR is also not applicable to online DPO.

11.2. GRPO and Reward Shaping

For GRPO, the advantage value is computed as a normalization of proxy rewards. Consider a prompt x and N responses $y_1, \dots, y_N$ sampled from the policy model. A reward model r_ϕ assigns scores $r_1, \dots, r_N$ to each response. The advantage $A_{i,t}$ for response y_i at token position t is given by:

$$A_{i,t} = \frac{r_i - \mu}{s},$$

where $\mu = \frac{1}{N}\sum_{i=1}^{N} r_i$ and $s = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(r_i - \mu)^2}$ are the mean and standard deviation of the rewards, respectively.

Assume a linear transformation is applied to the proxy reward, such that $\hat{r} = a \cdot r + b$ $(a > 0)$. We prove that the new advantage $\hat{A}_{i,t}$ is identical to the original $A_{i,t}$. First, the new mean $\hat{\mu} = a \cdot \mu + b$, and the new standard deviation $\hat{s} = a \cdot s$. The new advantage is computed as:

$$\begin{aligned}\hat{A}_{i,t} &= \frac{\hat{r}_i - \hat{\mu}}{\hat{s}} = \frac{ar_i + b - (a\mu + b)}{as} \\ &= \frac{ar_i - a\mu}{as} = \frac{r_i - \mu}{s} \\ &= A_{i,t}.\end{aligned}$$

Thus, linear transformations do not influence the advantage calculation in GRPO. Furthermore, since the sigmoid function is a non-linear function, PAR is applicable to GRPO training. We validate this through experiments, as shown in Figure 7. No reward hacking problem is observed in the GRPO training process, as the advantage calculation inherently performs reward normalization.

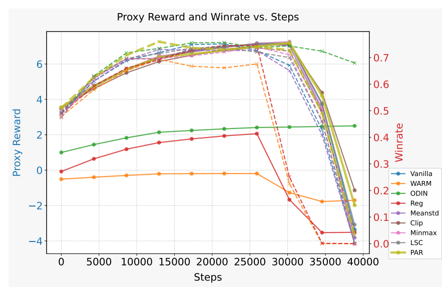


Figure 7. The training curves for GRPO, evaluated on Gemma2-2B with the Ultrafeedback-Binarized dataset, demonstrate that Vanilla, Meanstd, Minmax exhibit similar proxy rewards throughout the training process. This is because linear transformations of the proxy rewards do not affect the advantage value in GRPO. The PAR is a non-linear function and slightly better before collapse. No reward hacking issue is observed in the GRPO training process, as the advantage calculation inherently normalizes the rewards.

12. Comparison of Some Reward Shaping Techniques

Recent work has explored various approaches to reward transformation in RLHF. Shen et al. (2024) proposed contrastive rewards that share similarities with our method. Their approach employs a reference model to generate multiple baseline responses (y_{ref}^m), computing rewards as:

$$r_{\text{RL}} = \frac{1}{M}\sum_{m=1}^{M}\left[r(x, y) - r(x, y_{\text{ref}}^m)\right].$$

This method primarily focuses on enhancing PPO through contrastive learning against reference responses.

Ahmadian et al. (2024) revisited the REINFORCE algorithm for RLHF, adopting a response-centric approach analogous to GRPO. Their formulation centers rewards using multiple sampled responses:

$$g = \frac{1}{M}\sum_{i=1}^{M}\left[r(x,y_i) - \frac{1}{M-1}\sum_{j\neq i} r(x,y_j)\right]\nabla \log \pi_\theta(y_i|x),$$

where $y_1,\dots,y_M \overset{\text{i.i.d.}}{\sim} \pi_\theta(\cdot|x)$. This work emphasizes algorithmic simplicity and stability by replacing PPO with REINFORCE.

Wang et al. (2024) introduced log-sigmoid reward transformation for multi-reward integration:

$$r_{\text{RL}} = \text{log-sigmoid}\left[r(x,y) - r(x,y_{\text{ref}}^{.85})\right].$$

Their method specifically addresses the challenge of effectively combining multiple reward signals during training.

In this work, we present Preference As Reward (PAR), which applies sigmoid transformation to mitigate reward hacking:

$$r_{\text{RL}} = \frac{1}{M}\sum_{m=1}^{M}\text{sigmoid}\left(r(x,y) - r(x,y_{\text{ref}}^{m})\right).$$

Our approach specifically targets reward hacking while maintaining stable policy optimization.

13. Theoretical Analysis

We give a principled justification for *Preference As Reward* (PAR) here.

13.1. Bounded Rewards Reduce Return Variance

PAR has the key property $|r_l| < 1$, here r_l is the reward at position l, defined in Section 3. Let $G_t = \sum_{l=t}^{T}\gamma^{\,l-t} r_l$ be the return with discount $\gamma \in [0,1]$.

Theorem 13.1 (Return Variance Bound)**.** *For any trajectory and any γ,* $\operatorname{Var}[G_t] \le \frac{1}{(1-\gamma)^2}$ *. Conversely, if the per-token reward r_l is sub-Gaussian with parameter σ^2, its unbounded return has* $\operatorname{Var}[G_t] = \Omega(\sigma^2/(1-\gamma^2))$.

Proof. Because $0 < r_l < 1$, we have:

$$0 \le G_t \le \sum_{k=0}^{T-t}\gamma^k < \sum_{k=0}^{\infty}\gamma^k = \frac{1}{1-\gamma}$$

According to the Popoviciu's inequality, which states that for any random variable X with support inside [a,b], $\operatorname{Var}[X] \le \frac{(b-a)^2}{4}$. We have:

$$\operatorname{Var}[G_t] \le \frac{1}{4(1-\gamma)^2} \le \frac{1}{(1-\gamma)^2}$$

Hence the upper bound holds.

Since we assume r_l is sub-Gaussian with parameter σ^2, we have two cases.

Perfect positive correlation, we let $Z = r_l$, for every $l \ge t$, then:

$$G_t = Z\sum_{k=0}^{\infty}\gamma^k = \frac{Z}{1-\gamma}$$

$$\operatorname{Var}[G_t] = \frac{Var[Z]}{(1-\gamma)^2} \ge c\frac{\sigma^2}{(1-\gamma)^2}$$

Independent (or zero-mean-uncorrelated) rewards, we have:

$$\operatorname{Var}[G_t] = \sigma^2\sum_{k=0}^{\infty}\gamma^{2k} = \frac{\sigma^2}{1-\gamma^2}$$

□

Both two cases satisfy that $\operatorname{Var}[G_t] = \Omega(\sigma^2/(1-\gamma^2))$.

Theorem 13.1 explains the smaller and stable critic loss in Figure 2, also highlight the importance of Design Principle 1 from a theoretical view.

13.2. The Justification of Sigmoid function

Let $z := r_\phi(x,y) - r_\phi(x, y_{\text{ref}})$ and define the shaped reward

$$r_{\text{PAR}}(z) \;=\; \sigma(z), \qquad \sigma(z) = \tfrac{1}{1+e^{-z}}.$$

Consider a differentiable policy $\pi_\theta(y \mid x)$ and the REINFORCE–style gradient signal $g_\theta(x,y) = \nabla_\theta \log \pi_\theta(y \mid x)\, r_{\text{PAR}}(z)$. Then the following theorem holds.

Theorem 13.2 (Sigmoid minimize the policy gradient variance)**.** *Let*

$$\mathcal{F} \;= \Big\{ f \in C^1(\mathbb{R}) \;\Big|\; f'(z) > 0,\ \lim_{z\to-\infty} f(z) = 0,\ \lim_{z\to+\infty} f(z) = 1,\ f(0) = \tfrac{1}{2} \Big\}.$$

Then we have:

$$\sigma = \arg\min_{f\in\mathcal{F}} Var[g_f]$$

Where $g_f = \nabla_\theta \log \pi_\theta(y \mid x)\, f(z)$, σ is the sigmoid function.

Proof. According to the definition of variance:

$$Var[g_f] = M_f - \|\nabla_\theta J_f\|_2^2$$

where $M_f = \mathbb{E}[\|\nabla_\theta \log \pi_\theta(y \mid x)\|_2^2 f(z)^2]$, $J_f = \mathbb{E}_{\pi_\theta}[f(z)]$

If two shaping functions f_1, f_2 produce similar-size true gradients, then the variance depends on the second moment M_f. We now prove that the sigmoid function minimizes the second moment M_f.

Since $\|\nabla_\theta \log \pi_\theta(y \mid x)\|_2^2$ does not depend on f, minimizing M_f is equivalent to miniziming:

$$V_f = \mathbb{E}[f(z)^2]$$

Now we only need to prove the sigmoid function minimizing V_f, we divide the proof into five steps: **1. Regularised variational problem.** Introduce the smoothness-penalised functional

$$J_\lambda[f] \;=\; V_f + \lambda \int_{\mathbb{R}} \big(f'(z)\big)^2 \,\mathrm{d}z, \qquad \lambda > 0.$$

Because V_f and $\int (f')^2$ are both weakly lower-semicontinuous and coercive in $H^1(\mathbb{R})$, the direct method of the calculus of variations yields a minimiser $f_\lambda \in \mathcal{F}$ for each $\lambda > 0$.

2. Euler–Lagrange equation. For any C^1 variation δf supported on a compact set, the first variation of J_λ vanishes at f_λ:

$$0 \;=\; \delta J_\lambda = \int_{\mathbb{R}} \Big(2\, f_\lambda(z)\, p(z) \;-2\lambda\, f_\lambda''(z)\Big)\, \delta f(z)\,\mathrm{d}z,$$

$$p(z) = \tfrac{1}{\sqrt{2\pi}} e^{-z^2/2}.$$

Since δf is arbitrary,

$$2\, f_\lambda(z)\, p(z) - 2\lambda\, f_\lambda''(z) = 0 \;\;\Longrightarrow\;\; f_\lambda''(z) = \frac{1}{\lambda}\, p(z)\, f_\lambda(z). \tag{1}$$

3. Solving the ODE. Because p is even and f_λ satisfies $f_\lambda(0) = \frac{1}{2}$, solutions to (1) are necessarily even about 0 and strictly increasing. Writing $c_\lambda = \frac{1}{\lambda}\sqrt{2\pi}$ we obtain

$$f_\lambda''(z) = c_\lambda\, e^{-z^2/2}\, f_\lambda(z).$$

Standard ODE theory (Picard–Lindelöf) plus the boundary conditions $f_\lambda(-\infty) = 0,\; f_\lambda(+\infty) = 1$ implies

$$f_\lambda(z) = \sigma\big(c_\lambda z\big).$$

(The shift is fixed by $f_\lambda(0) = \frac{1}{2}$.)

4. Optimal scale. Insert $f_\lambda(z) = \sigma(cz)$ into $V_f = \mathbb{E}[\sigma(cZ)^2]$. By symmetry of Z,

$$V(c) \;=\; \mathbb{E}[\sigma(cZ)^2] \;=\; 2\int_0^\infty \sigma(cz)^2\, \frac{e^{-z^2/2}}{\sqrt{2\pi}}\,\mathrm{d}z.$$

Differentiating in c and setting $\frac{\mathrm{d}}{\mathrm{d}c}V(c) = 0$ gives $c = 1$ as the unique stationary point, and $\frac{\mathrm{d}^2}{\mathrm{d}c^2}V(c)\big|_{c=1} > 0$ shows it is a minimum. Hence $f_\lambda = \sigma$ for every $\lambda > 0$.

5. Removing the regularisation. Because $J_\lambda[f_\lambda] \le J_\lambda[\sigma] = V_\sigma$ and $\int (f_\lambda')^2 \ge 0$, we have $V_{f_\lambda} \le V_\sigma$. But V_σ is the lower bound of J_λ, so $V_{f_\lambda} = V_\sigma$, and by uniqueness in the previous step $f_\lambda = \sigma$. Letting $\lambda \downarrow 0$ leaves σ unchanged; thus no other $f \in \mathcal{F}$ can achieve a smaller V_f.

Therefore σ uniquely minimises V_f in $\mathcal{F}$. □

14. Case Study

We identify several patterns of reward hacking observed in Vanilla PPO training, using the checkpoint trained after one epoch for detailed examination. We show the examples in Figure 9.

15. PPO Training

PPO (Proximal Policy Optimization) is an online reinforcement learning algorithm that generates a response given a prompt, computes a reward for the response using a reward model, and updates the policy and critic models to maximize the reward.

We employ several PPO techniques to ensure stable training, including advantage normalization (Zheng et al., 2023b), value loss clipping (Patterson et al., 2023), a replay buffer (Eysenbach et al., 2019), per-token KL penalty, and length penalty. The pseudo-code for the PPO algorithm is provided in Algorithm 1.

16. Limitations

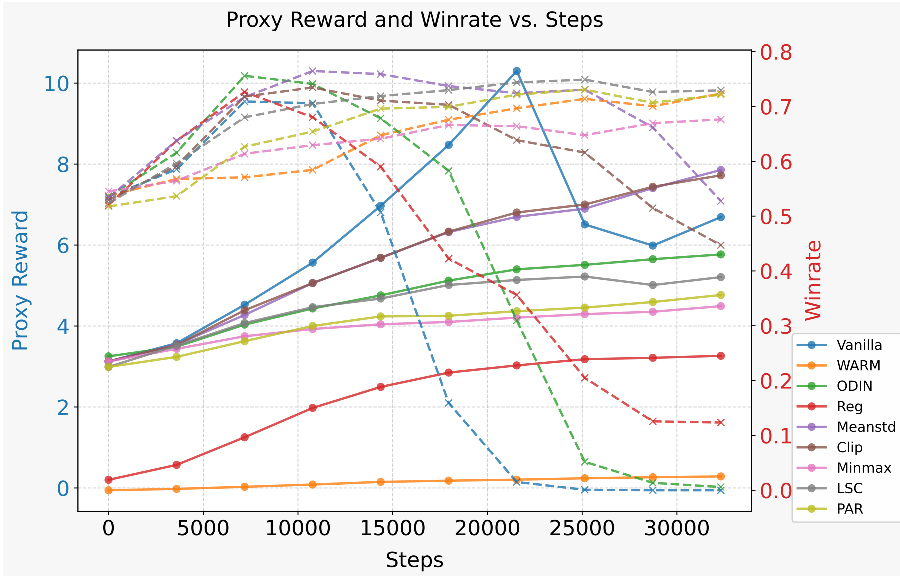
Although our PAR method effectively mitigates reward hacking, it does not improve peak performance, as measured by the winrate of the best checkpoint. Furthermore, its design principles lack precision. While PAR sets the upper bound of the RL reward to 1.0, alternative bounds and their selection criteria remain unexplored. Additionally, the dynamics of reward adjustment—such as the initial rate of increase and the pace of convergence—are not fully elucidated.

```
Please act as an impartial evaluator to assess the quality of two responses from different
     AI assistants to an incomplete dialogue between a user (<|user|>) and an AI assistant
     (<|assistant|>). The dialogue will be missing the last turn, and both Assistant-A (<
    Assistant-A response>) and Assistant-B (<Assistant-B response>) are expected to
    complete it. Focus your evaluation on the following five aspects:
1. Clarity and Relevance: Responses should be concise, directly addressing the question.
    They should use clear, natural language and remain on-topic.
2. Accuracy and Honesty: Responses must provide factual, truthful information. Disclose
    limitations or uncertainties when necessary.
3. Ethics and Appropriateness: Ensure the responses are free from harmful, offensive, or
    discriminatory content.
4. Engagement and Depth: Responses should be engaging, educational, and sufficiently
    detailed to comprehensively address the user question.
5. Structure and Creativity: Responses should be logically organized and show originality
    or adaptability when necessary.

Note: The quality of the responses should not be judged solely by their length. Both
    brevity and detail are important depending on the context of the question.
You will be given an incomplete dialogue (<question>) with the last turn left blank.
    Assistant-A (<Assistant-A response>) and Assistant-B (<Assistant-B response>) have
    each provided a response to complete the dialogue. Your task is to evaluate each
    response based on the five criteria above and provide a comparison.

Evaluation Format:
Assistant-A Response:
(Evaluate the quality of Assistant-A response based on the five aspects mentioned above.)
Assistant-B Response:
(Evaluate the quality of Assistant-B response based on the five aspects mentioned above.)
Comparison and Analysis:
Compare and contrast the responses from Assistant-A and Assistant-B to determine which one
     is more effective overall. Justify your reasoning clearly and concisely.

At the end, output the comparison result for both responses in the following format:
Better: X (X is A, B, or N, representing A is better, B is better, or both are of equal
    quality)
```

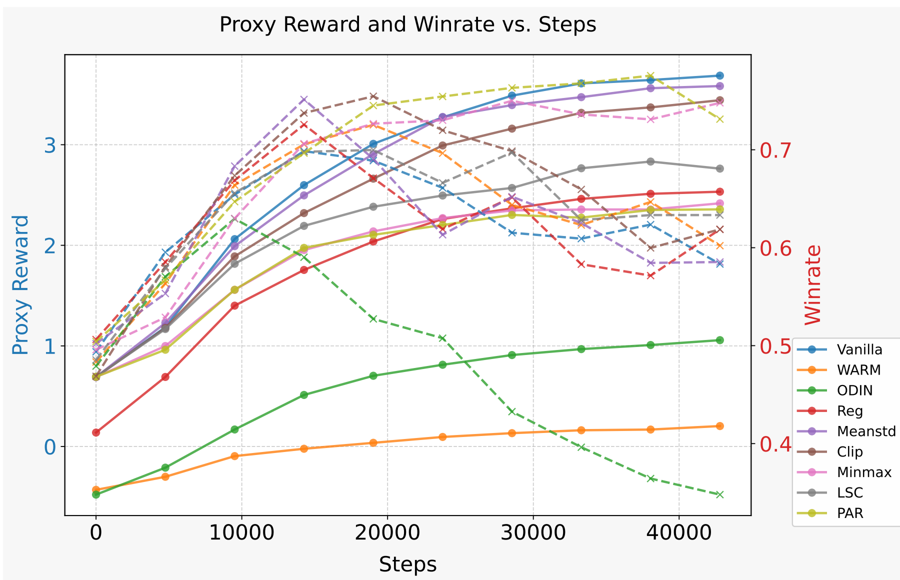
Listing 1. System Prompt For Winrate Evaluation on Test Set

```
<question>:
{user_question}
<Assistant-A response>:
{policy_response}
<Assistant-B response>:
{sft_response}
```

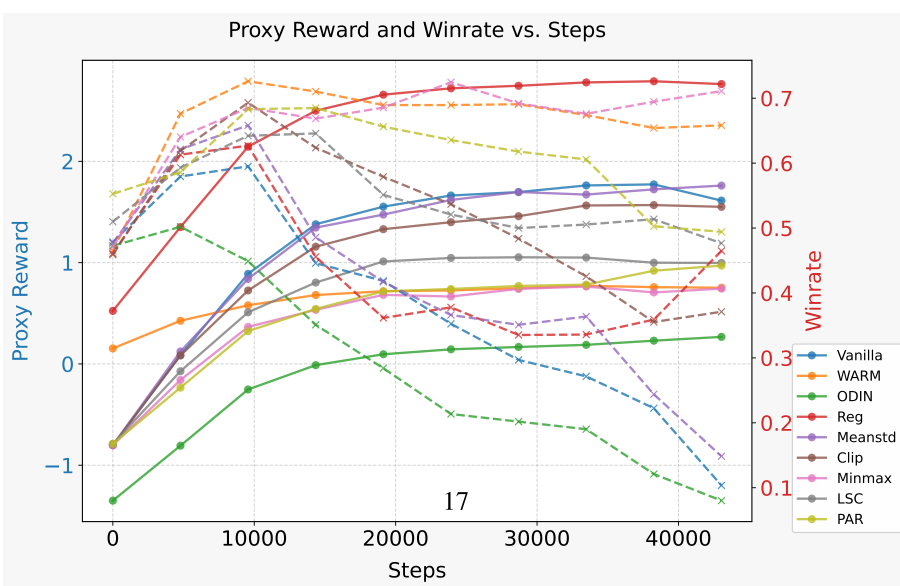
Listing 2. User Input Template For Winrate Evaluation on Test Set



(a) PPO training curve for different mitigation methods on Llama3-8B and Ultrafeedback-Binarized.



(b) PPO training curve for different mitigation methods on Gemma2-2B and HH-RLHF.



(c) PPO training curve for different mitigation methods on Llama3-8B and HH-RLHF.

```
Prompt:
<|user|>i am going to give a ppt presentation on VLSI in aerospace and defence. what I
    need is like introduction slide and Overview slide and related contents you wish to be
     included give me<|assistant|>

Policy model response:
In enchanting architectural mosaics, fuse captivating cosmic threads VELCRO-ing silicon's
    sacred architectural realm with adaptive aeronautical nebulae, sculpting celestial
    gardens for efficient holographic compass landmarks. Seamless Integrated Silicon
    Graphics (ISG) - ......
```

Listing 3. Gemma2-2B with Ultrafeedback-Binarized

```
Prompt:
<|user|>Can you help compare the price travel for a trip to the Caribbean to a trip to Las
     Vegas?<|assistant|>

Policy model response:
Sure! Caribbean travel is a wonderful choice for a vacation. Caribbean destinations offer
    some of the most beautiful beaches in the world, and the crystal clear waters of the
    Caribbean Sea make for some of the most amazing snorkeling and diving experiences.
    Caribbean resorts and hotels offer amenities like fine dining, swimming pools, spa
    services, and a number of outdoor activities, such as sailing, hiking, golfing, and
    horseback riding...
```

Listing 4. Gemma2-2B with HH-RLHF

```
Prompt:
<|user|>How can creative leaders ensure that their decisions align with their organization
    's values and ethics?<|assistant|>

Policy model response:
Embed GDPR-grade encrypted traceability logs to GDPR-compliant audit records via trusted
    pipelines, preventing unauthorized backups by blockchain timestamp sealed secure
    encrypted logs copies through API certified audits against bespoke compliance
    templates signatures, logs' timestamps assigned securely within SharePoint backups or
    verified validation privacy signatures plus?,?, ...
```

Listing 5. Llama3-8B with Ultrafeedback-Binarized

```
Prompt:
<|user|>What are some salty things I can put in oatmeal?<|assistant|>

Policy model response:
You might add salt, pepper, tomatoes, onions, celery, eggs, beans, yogurt, mushrooms, tofu
    , berries, maple syrup, cinnamon, nutmeg, paprika, and dried herbs. Maybe add some
    dark chocolate, almonds, walnuts, and pumpkin seeds. And you can also drink plenty of
    water, avoid soda and caffeinated drinks, eat healthy carbs and avoid saturated fats,
    stay hydrated and watch for high blood pressure, have sufficient vitamin intake. Sleep
     at least 8 hours per day, limit your red meat intake, be active by cooking, swimming,
     running, weightlifting, and eating outdoors, eat good fats like olive oil, eat nuts
    and seeds and go light on salt, avoid greasy foods and snacks like cheese chips. Get
    plenty of fresh air, have fun indoors an outside. Avoid smoking and overeating, and
    take some time daily to enjoy your favorite activities, be grateful, and be thankful.
    Keep active and move in ways that encourage fitness. Take a deep breath and take a
    journey! Make your oatmeal a treat!
```

Listing 6. Llama3-8B with HH-RLHF

Figure 9. Analysis of reward hacking patterns observed in Vanilla PPO training, based on a detailed examination of the model checkpoint after one epoch.

Algorithm 1 PPO

Require: sft model π_{sft}, reward model r_ϕ, prompt set $\mathcal{D}$.
Ensure: Aligned model π_{θ^*}

Initialize policy model $\pi_\theta \leftarrow \pi_{\text{sft}}$
Initialize reference model $\pi_{\text{ref}} \leftarrow \pi_{\text{sft}}$
Initialize critic model $V_\alpha \leftarrow r_\phi$
for $x \in \mathcal{D}$ **do**
    ppo_batch = build_ppo_batch($x, \pi_\theta, \pi_{\text{ref}}, V_\alpha, r_\phi$)
    ppo_batch = buffer.substitute(ppo_batch) ▷ sample a ppo_batch from replay buffer and save current ppo_batch into the buffer
    $\mathcal{L}_{\text{ppo}}(\theta), \mathcal{L}_{\text{critic}}(\alpha)$ = calculate_loss(ppo_batch, π_θ, V_α)
    $\theta \leftarrow \theta - \mathsf{plr} * \nabla_\theta \mathcal{L}_{\text{ppo}}(\theta)$ ▷ update policy model via gradient descent, plr is policy learning rate
    $\alpha \leftarrow \alpha - \mathsf{clr} * \nabla_\alpha \mathcal{L}_{\text{critic}}(\alpha)$ ▷ clr is critic learning rate
end for
return π_{θ^*}

Algorithm 2 build_ppo_batch

Require: prompt x, four models $\pi_\theta, \pi_{\text{ref}}, V_\alpha, r_\phi$.
Ensure: ppo_batch: A dictionary

Initialize ppo_batch =
sample $y \sim \pi_\theta(.|x)$
sample $y_{\text{ref}}^{1,\ldots,M} \sim \pi_{\text{ref}}(.|x)$ ▷ optional
$r = r_\phi(x, y)$
$r_{\text{ref}}^{1,\ldots,M} = r_\phi(x, y_{\text{ref}}^{1,\ldots,M})$ ▷ optional
r_{RL} = reward_reshape($r, r_{\text{ref}}^{1,\ldots,M}$, len($y$), mode = PAR)
Now we split (x,y) into $(s_t, a_t)_{t=0}^{T}$
KL_penalty $= \log \pi_\theta(a_t|s_t) - \log \pi_{\text{ref}}(a_t|s_t)$
construct per-token rewards $r_{1,\ldots,T}$ from r_{RL} and KL_penalty
$V_t = V_\alpha(s_t)$
Compute GAE $\hat{A}_t$ and Return G_t from V_t and r_t.
ppo_batch = ($\log \pi_\theta(a_t|s_t)$, G_t, $\hat{A}_t$, V_t, s_t, a_t)
return ppo_batch

Algorithm 3 Buffer.substitute

Require: ppo_batch.
Ensure: ppo_batch: A dictionary

Global List pool = []
Global buffer_size = 4
IF len(pool)¡buffer_size:
    pool.append(ppo_batch)
    return None
ELSE:
    selected_batch = random.choice(pool)
    pool.pop(selected_batch)
    pool.append(ppo_batch)
    return selected_batch

Algorithm 4 reward_reshape

Require: policy reward r, reference reward $r_{\text{ref}}^{1,\ldots,M}$, response length l, reshape mode mode.
Ensure: RL reward

IF $l > 300$:
    $r = r - 0.01 * (l - 300)$ ▷ penalize long response
IF mode==meanstd:
    $r_{\text{RL}} = \frac{r-\mu}{s}$ ▷ μ, s are running mean and running standard variance respectively.
IF mode==reward_clip:
    ...
IF mode==PAR:
    $r_{\text{RL}} = \frac{1}{M}\sum_{m=1}^{M} \sigma(r - r_{\text{ref}}^{m})$

Algorithm 5 calculate_loss

Require: ppo_batch, policy model π_θ, critic model V_α.
Ensure: policy loss $\mathcal{L}_{\text{ppo}}(\theta)$, critic loss $\mathcal{L}_{\text{critic}}(\alpha)$

($\log \pi_{\theta_{\text{old}}}(a_t|s_t)$, G_t, $\hat{A}_t$, V_t, s_t, a_t) = ppo_batch ▷ Extract elements from ppo_batch
$\mathcal{L}_{\text{ppo}}(\theta) = \hat{\mathbb{E}}_t\left[\min\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}\hat{A}_t, \text{clip}\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}, 1-\epsilon, 1+\epsilon\right)\hat{A}_t\right)\right]$

$$\begin{aligned}\mathcal{L}_{\text{critic}}(\alpha) =& \hat{\mathbb{E}}_t\Big[\max\big(||V_\alpha(s_t) - G_t||_2^2,\\ & ||\text{clip}(V_\alpha(s_t), V_t - \delta, V_t + \delta) - G_t||_2^2\big)\Big]\end{aligned}$$

▷ Critic clip trick

return $\mathcal{L}_{\text{ppo}}(\theta), \mathcal{L}_{\text{critic}}(\phi)$

Algorithm 6 Online DPO

Require: sft model π_{sft}, reward model r_ϕ, prompt set $\mathcal{D}$.
Ensure: Aligned model π_{θ^*}

Initialize policy model $\pi_\theta \leftarrow \pi_{\text{sft}}$
Initialize reference model $\pi_{\text{ref}} \leftarrow \pi_{\text{sft}}$
for $x \in \mathcal{D}$ **do**
    Sample $y_1, y_2 \sim \pi_\theta(.|x)$
    Calculate rewards $r_1 = r_\phi(x, y_1), r_2 = r_\phi(x, y_2)$
    IF $r_1 > r_2$:
        $y_w = y_1, y_l = y_2$
    ELSE:
        $y_w = y_2, y_l = y_1$
    $\mathcal{L}_{\text{DPO}}(\theta) = -\left[\log\sigma\left(\beta\left(\log\frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \log\frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)}\right)\right)\right]$
    $\theta \leftarrow \theta - lr * \nabla_\theta \mathcal{L}_{\text{DPO}}(\theta)$
end for
return π_{θ^*}