Pairwise Proximal Policy Optimization: Large Language Models Alignment via Comparative RL

Tianhao Wu

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2024-43
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2024/EECS-2024-43.html

May 1, 2024

Pairwise Proximal Policy Optimization: Large Language Models Alignment via Comparative RL

by Tianhao Wu

Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in partial satisfaction of the requirements for the degree of **Master of Science, Plan II**.

Approval for the Report and Comprehensive Examination:

Committee:

DocuSigned by:
Jiantao Jiao
4D87CA82301A457...

Professor Jiantao Jiao
Research Advisor

2024/4/30

(Date)

* * * * * * *

DocuSigned by:
Kannan Ramchandran
379625629BF0443...

Professor Kannan Ramchandran
Second Reader

2024/5/1

(Date)

Pairwise Proximal Policy Optimization: Large Language Models Alignment via Comparative RL

Tianhao Wu
Department of Electrical Engineering and Computer Sciences
University of California, Berkeley

Abstract

LLMs may exhibit harmful behavior without aligning with human values. The dominant approach for steering LLMs towards beneficial behavior is Reinforcement Learning from Human Feedback (RLHF). This involves training a reward model with a human-labeled ranking dataset and fine-tuning the LLM with the reward signal using RL. Despite the fact that the reward is learned from comparing different responses, the RL stage does not involve direct comparisons. This inconsistency between reward learning and reinforcement learning stages exacerbates RL's instability. An example would be that the well adopted RL optimizer, Proximal Policy Optimization (PPO), could perform different gradient updates even for batches with identical human preference information. To address this, we propose a new framework, reinforcement learning from comparative feedback, and a simple policy gradient algorithm, Pairwise Proximal Policy Optimization (P3O), that learns to improve from direct comparison. Theoretically, P3O has the nice property of being invariant with any reward functions that contain identical preference information, while not requiring learning a value function. Empirical evaluations demonstrate that P3O can align with human preferences better than existing methods. This suggest that comparative RL is strong candidate for aligning LLM with preference data.

1 Introduction

Large Language Models (LLMs) have made remarkable progress, profoundly influencing the AI community (Chowdhery et al., 2022; Brown et al., 2020; Touvron et al., 2023; Bubeck et al., 2023). But these models can also generate outputs that are untruthful, toxic, or reflect harmful sentiments. This is in part because LLMs are trained to predict the next word on a large dataset of Internet text, rather than to safely perform the language task that the user wants. In other words, these models are not aligned with their users. Consequently, it is crucial to align LLMs with human values, *e.g.*, helpful, honest, harmless (Bai et al., 2022a).

A leading method in AI Alignment for Large Language Models (LLMs), known as Reinforcement Learning from Human Feedback (RLHF), involves learning a reward function and fine-tuning the model with this reward feedback using reinforcement learning (RL) (Ziegler et al., 2019; Ouyang et al., 2022). Specifically, a reward model is trained to rank candidate responses to align with the human-labeled ground-truth. As for RL, Proximal Policy Optimization (PPO) is widely adopted as the default optimizer (Schulman et al., 2017). PPO alternates between generating new responses and adjusting the likelihood toward responses with higher reward. Despite its acclaimed efficiency, we identify the inconsistency between these two stages.

Inconsistency Between Reward Learning and RL. Although rewards are derived by ranking responses according to human judgments, the RL phase does not incorporate comparisons between generated samples. This leads to a scenario where the reward signal can be highly variable, often being lower for more challenging prompts and higher for simpler ones. When employing conventional algorithms like Proximal Policy Optimization (PPO) to optimize against such a noisy reward, it becomes likely for PPO to reduce the likelihood

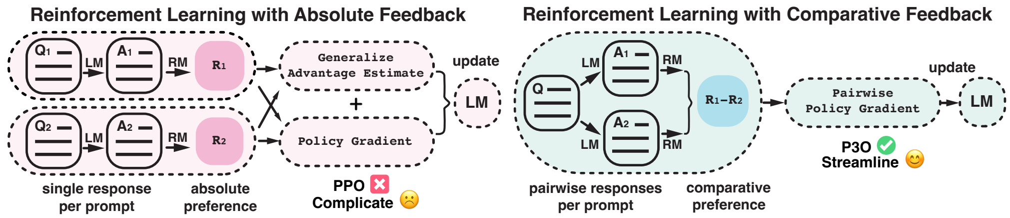

Figure 1: The figure on the left illustrates the prevalent method for fine-tuning LMs using RL, which relies on **Absolute Feedback**. In this paradigm, algorithms like PPO have to learn a V function, which capture not only the valuable relative preference information, but to a lesser extent, which is the scale of the reward for a given prompt. Contrastingly, the figure on the right presents an RL paradigm that improves from direct comparison. Our algorithm generates a pair of responses per prompt, leveraging only the **Comparative Feedback** - derived from the difference in reward - for policy gradient updates. This method obviates the need for additional critic learning and intricate components like Generalized Advantage Estimation (Schulman et al., 2015b).

of a quality response to a difficult prompt due to the absence of comparisons. This issue is further compounded by PPO's sensitivity to various factors including reward normalization, scaling, clipping, KL control, advantage normalization, and critic initialization (Zheng et al., 2023; Engstrom et al., 2020), contributing to its fragility. For instance, (Zhu et al., 2023b) pointed out that vanilla PPO would result in explosive generation length after exposure to only 1000-2000 prompts. They found that a negative initial reward will result in length exploding in the warmup phase, while a positive reward will shorten the generation length (Figure 2). Therefore, they proposed shifting the reward to be slightly positive for length control. However, this ad-hoc approach can only mitigate the issue and requires additional effort to tune the reward mean. We also observe similar instability in PPO, especially the decline of reward during the warm-up period, as well as the training being extremely sensitive to the random seed.

From a formal perspective, we argue that this inconsistency arises because the reward training objective, the Bradley-Terry Loss, is invariant to a constant shift, whereas PPO is not. This implies that even if two reward models contain identical human preference information, their optimization via PPO could lead to disparate results (Section 5). We further highlight that a clear solution to this issue is to use comparative RL, which uses comparative ranking information to steer the language model toward more preferred responses.

In this paper, we provide new insights to address the inconsistency:

- We define an equivalent relationship for reward functions trained from human preferences. We identify that the widely adopted reward training loss, Bradley-Terry Loss, is invariant under this equivalent relationship, while PPO is not. As a result, PPO may be less efficient at learning the reward.

- We introduce Pairwise Proximal Policy Optimization (P3O), under the framework of Reinforcement Learning from Comparative Feedback (Figure 1). P3O learns by comparing pairs of responses, avoiding learning critic functions, advantage estimation and various normalization techniques (Zheng et al., 2023). Empirical evaluations show that P3O consistently outperforms PPO and Direct Preference Optimization (DPO) in terms of GPT-4 Evaluation.

The reminder of the paper is organized as follows. In Section 2 and Section 3, we review related works and the RLHF pipeline. In Section 4, we formally derive our algorithm P3O and discuss the intuition behind it. We address the current limitation of PPO and provide the motivation behind P3O in Section 5. We present experiment results including ablation studies in Section 6. We discuss future works and conclude in Section 7.

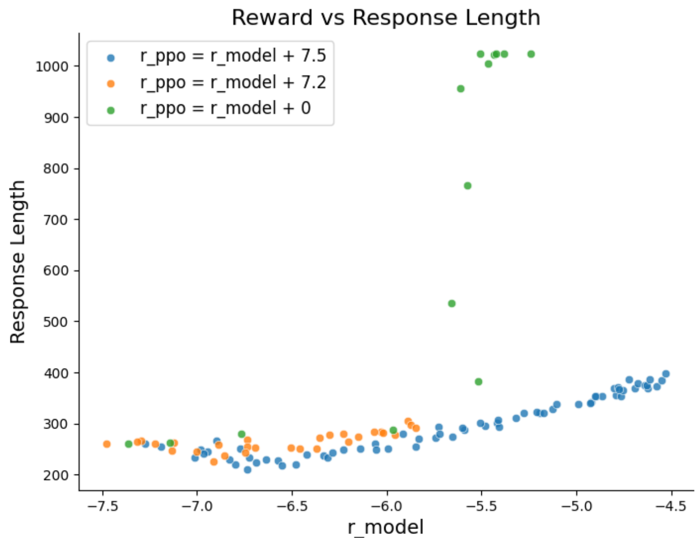

Figure 2: **Instability of PPO:** Impact of reward shifting constant on response length. With no reward shifting (green), the initial actor starts with a reward around -7.3. Three different reward shifting parameters are tested: 0 (no shift), 7.2 (starting with a slightly negative reward), and 7.5 (starting with a slightly positive reward). The results show that PPO is extremely sensitive to reward shifting, starting with a slightly positive reward can better control the response length while achieving nearly the same final reward.

2 Related Work

Significant efforts have been made towards aligning LLMs with human values. These alignment strategies broadly fall into two categories: offline training and online training.

Offline training typically involve a static dataset, and do not require additional evaluations or generations. For instance, Thoppilan et al. (2022); Gunasekar et al. (2023) use instruction fine-tuning to update the model on a high quality dataset tailored to a specific downstream task of interest. Snell et al. (2022) proposed to employ offline Q Learning to learn an add-on term for decoding. Rafailov et al. (2023) introduced Direct Preference Optimization (DPO), an offline approach that can directly align LM with human preference data, drawing from the closed-form solution of the Contextual Bandit with KL control problem. There are also methods like PRO (Song et al., 2023) and RRHF (Yuan et al., 2023) that fine-tune the model based on ranking of the rewards.

Our work is categorized under online training, which consist of a loop of generating new responses from the updated policy, evaluating them with the reward model and updating the policy. The current dominant approach RLHF relies on online RL methods such as PPO (Schulman et al., 2017), A2C (Mnih et al., 2016) or their variants (Ramamurthy et al., 2022; Zhu et al., 2023c). There are also a few methods that deviate from this standard. For instance, Gulcehre et al. (2023) introduce ReST, which uses offline RL instead of online RL in the policy improvement phase. Besides, Dong et al. (2023a) proposed RAFT, which iteratively fine-tunes the policy on the responses generated by the Best-of-N policy. Another paradigm parallel to RLHF is Reinforcement Learning from AI Feedback (RLAIF) (Zhu et al., 2023a; Bai et al., 2022b; Lee et al., 2023; Yuan et al., 2024), which aims for using AI to improve AI. RLAIF substitutes the role of humans with AI in the feedback loop and yields comparable results with RLHF.

Outside of the context of language, preference-driven policy learning has been explored in both bandit and RL. Contextual dueling bandit (Dudík et al., 2015; Yue et al., 2012) use preferences or rankings of actions to adjust the policy, rather than rewards. Similarly, PbRL

(Xu et al., 2020; Jain et al., 2013; Busa-Fekete et al., 2014; Christiano et al., 2017; Sadigh et al., 2017; Kupcsik et al., 2018) learn from binary preferences generated by some unknown scoring function.

2.1 Necessity of RL in LLM Alignment

The necessity of reinforcement learning (RL) for aligning large language models (LLMs) has been a topic of much debate. There are alternative approaches, such as Direct Policy Optimization (DPO), which is a simpler method that utilizes a pre-collected offline dataset of preferences. On the other hand, RL involves a more intricate process of interacting with the language model in real-time. This process includes generating new responses, evaluating them using a reward model, and then updating the language model based on the rewards.

Despite the complexity and the challenge of hyperparameter optimization inherent to online RL methods, recent studies have demonstrated that, with careful hyperparameter tuning, online RL can produce results that are much stronger than non-RL methods Zhu et al. (2023a); Lambert & Calandra (2023); Dong et al. (2023b); Xu et al. (2024). However, conclusively addressing the necessity of RL for LLM alignment would require more sophisticated tooling and evaluations, and would be beyond the scope of this paper.

3 Preliminaries

We briefly reviewing the RLHF pipeline in (Ziegler et al., 2019).

- **SFT Phase (Supervised Fine-Tuning):** This stage starts with a pre-trained LM, is then fine-tuned with supervised learning (typically maximum likelihood loss) on a high quality dataset for the downstream task of interest. These tasks could be dialogue, instruction following and summarization. The outcome of this stage is the supervised fine-tuned model, denoted as π^{SFT}.
- **Reward Learning Phase.** In the second phase the SFT model is prompted with prompts $\boldsymbol{x}$ to produce pairs of answers $\boldsymbol{y}_1, \boldsymbol{y}_2 \sim \pi^{\mathrm{SFT}}(\boldsymbol{y}|\boldsymbol{x})$. The responses pairs are then presented to human labelers who express preferences for one answer, denoted as $\boldsymbol{y}_w \succ \boldsymbol{y}_l|\boldsymbol{x}$, where $\boldsymbol{y}_w$ is the one favored by the labeler and $\boldsymbol{y}_l$ is the one less favored. Under these preferences is the inaccessible latent reward model $r^*(\boldsymbol{y}, \boldsymbol{x})$. There are several approaches used to model preferences, the Bradley-Terry (Bradley & Terry, 1952) model being a popular choice (although more general Plackett-Luce ranking models (Luce, 2012; Plackett, 1975) are also compatible with the framework if we have access to the ranking of answers). According to the BT model, the human preference distribution p^* can be expressed as:
$$p^*(\boldsymbol{y}_1 \succ \boldsymbol{y}_2|\boldsymbol{x}) = \frac{1}{1 + \exp\left(r^*(\boldsymbol{y}_2|\boldsymbol{x}) - r^*(\boldsymbol{y}_1|\boldsymbol{x})\right)}$$
Assuming the access to a dataset $\{(\boldsymbol{x}^i, \boldsymbol{y}_w^i, \boldsymbol{y}_l^i)\}_{i=1}^N$ sampled from p^*. We parameterize the reward as r_ϕ and estimate it via maximum log-likelihood:
$$\sum_{i=1}^{N} \frac{1}{N} \log \sigma \left(r_\phi(\boldsymbol{y}_w^i|\boldsymbol{x}^i) - r_\phi(\boldsymbol{y}_l^i|\boldsymbol{x}^i) \right) \tag{1}$$
where σ is the sigmoid function. r_ϕ is initialized with π^{SFT} augmented by additional linear layers on top of the final attention layer. To normalize the scale and lower the variance of r_ϕ, constraints like $\mathbb{E}\left[r(\boldsymbol{y}|\boldsymbol{x})\right] = 0$ might be incorporated.
- **RL Fine-Tuning Phase.** Prior work formulate the optimization problem as maximizing:
$$\mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathcal{D}, \boldsymbol{y}\sim\pi_\theta} \left[r_\phi(\boldsymbol{y}|\boldsymbol{x}) - \beta D_{\mathrm{KL}}(\pi_\theta(\cdot|\boldsymbol{x}) \| \pi^{\mathrm{SFT}}(\cdot|\boldsymbol{x})) \right] \tag{2}$$
The coefficient β is used to regulate the deviation from the SFT model. The KL-divergence term is important to prevent the model from deviating from the distribution on which the reward model is trained, as well as preventing the model from completely forget the world knowledge acquired in the pre-training stage. The standard approach is to directly employ PPO (Schulman et al., 2017; Ouyang et al., 2022) to optimize the modified reward $r_\phi(\boldsymbol{y}|\boldsymbol{x}) - \beta \left(\log \pi_\theta(\boldsymbol{y}|\boldsymbol{x}) - \log \pi^{\mathrm{SFT}}(\boldsymbol{y}|\boldsymbol{x})\right)$.

4 Algorithm

4.1 Proximal Pairwise Policy Optimization (P3O)

To derive P3O, we start from Vanilla Policy Gradient (VPG, Pseudocode 2) (Sutton et al., 1999; Schulman et al., 2017; Wu et al., 2022). For clarity, we'll focus on the bandit setting, though it can be extended to contextual bandits as in theorem 4.1.

In the bandit setting, assume we are updating a parameterized policy π_θ with actions denoted as $\boldsymbol{y}$. The VPG aims for estimating the following formula with samples:

$$\nabla \mathcal{L}^{\mathrm{VPG}} = \mathop{\mathbb{E}}_{\boldsymbol{y} \sim \pi_\theta} (r(\boldsymbol{y}) - b) \nabla \log \pi_\theta(\boldsymbol{y}) \tag{3}$$

where b is a baseline used for variance reduction. A common choice for the baseline is the mean reward, which gives:

$$\begin{aligned}\nabla \mathcal{L}^{\mathrm{VPG}} &= \mathop{\mathbb{E}}_{\boldsymbol{y}_1 \sim \pi_\theta} (r(\boldsymbol{y}_1) - \mathop{\mathbb{E}}_{\boldsymbol{y}_2 \sim \pi_\theta} r(\boldsymbol{y}_2)) \nabla \log \pi_\theta(\boldsymbol{y}_1) \\ &= \mathop{\mathbb{E}}_{\boldsymbol{y}_1, \boldsymbol{y}_2 \sim \pi_\theta} (r(\boldsymbol{y}_1) - r(\boldsymbol{y}_2)) \nabla \log \pi_\theta(\boldsymbol{y}_1) \end{aligned} \tag{4}$$

Equation 4 highlights the reliance on the relative difference between rewards. Symmetrizing for $\boldsymbol{y}_1, \boldsymbol{y}_2$ yield:

$$\nabla \mathcal{L}^{\mathrm{VPG}} = \mathop{\mathbb{E}}_{\boldsymbol{y}_1, \boldsymbol{y}_2 \sim \pi_\theta} (r(\boldsymbol{y}_1) - r(\boldsymbol{y}_2)) \nabla \left(\log \frac{\pi_\theta(\boldsymbol{y}_1)}{\pi_\theta(\boldsymbol{y}_2)} \right) /2$$

Its immediate generalization to contextual bandit is the following:

Theorem 4.1 (Pairwise Policy Gradient). *For any prompt $\boldsymbol{x}$, the policy gradient can be expressed as $\nabla \mathcal{L}^{VPG} = \mathop{\mathbb{E}}_{\boldsymbol{x} \sim \mathcal{D}} \nabla \mathcal{L}^{PPG}(\boldsymbol{x})$, where $\nabla \mathcal{L}^{PPG}(\boldsymbol{x})$ can be expressed as:*

$$\mathop{\mathbb{E}}_{\boldsymbol{y}_1, \boldsymbol{y}_2 \sim \pi_\theta} (r(\boldsymbol{y}_1|\boldsymbol{x}) - r(\boldsymbol{y}_2|\boldsymbol{x}))\, \nabla \left(\log \frac{\pi_\theta(\boldsymbol{y}_1|\boldsymbol{x})}{\pi_\theta(\boldsymbol{y}_2|\boldsymbol{x})} \right) /2$$

Proof. In the contextual bandit setting, VPG aims for estimating the gradient $\nabla \mathcal{L}^{\mathrm{VPG}} = \mathbb{E}_{x \sim \mathcal{D}} \nabla \mathcal{L}^{\mathrm{VPG}}(\boldsymbol{x})$, where $\nabla \mathcal{L}^{\mathrm{VPG}}(\boldsymbol{x})$ can be expressed as:

$$\nabla \mathcal{L}^{\mathrm{VPG}}(\boldsymbol{x}) = \mathop{\mathbb{E}}_{\boldsymbol{y} \sim \pi_\theta(\boldsymbol{y}|\boldsymbol{x})} r(\boldsymbol{y}|\boldsymbol{x}) \nabla \log \pi_\theta(\boldsymbol{y}|\boldsymbol{x})$$

Again we subtract the baseline $\mathop{\mathbb{E}}_{\boldsymbol{y}_2 \sim \pi_\theta(\cdot|\boldsymbol{x})} r(\boldsymbol{y}_2|\boldsymbol{x})$:

$$\begin{aligned}\nabla \mathcal{L}^{\mathrm{VPG}} &= \mathop{\mathbb{E}}_{\boldsymbol{y}_1 \sim \pi_\theta(\cdot|\boldsymbol{x})} (r(\boldsymbol{y}_1|\boldsymbol{x}) - \mathop{\mathbb{E}}_{\boldsymbol{y}_2 \sim \pi_\theta(\cdot|\boldsymbol{x})} r(\boldsymbol{y}_2|\boldsymbol{x})) \nabla \log \pi_\theta(\boldsymbol{y}_1|\boldsymbol{x}) \\ &= \mathop{\mathbb{E}}_{\boldsymbol{y}_1, \boldsymbol{y}_2 \sim \pi_\theta(\cdot|\boldsymbol{x})} (r(\boldsymbol{y}_1|\boldsymbol{x}) - r(\boldsymbol{y}_2|\boldsymbol{x})) \nabla \log \pi_\theta(\boldsymbol{y}_1|\boldsymbol{x}) \end{aligned} \tag{5}$$

Swap actions $\boldsymbol{y}_1, \boldsymbol{y}_2$ in Eq (5) and average together we get the desired form:

$$\nabla \mathcal{L}^{\mathrm{VPG}}(\boldsymbol{x}) = \mathop{\mathbb{E}}_{\boldsymbol{y}_1, \boldsymbol{y}_2 \sim \pi_\theta(\cdot|\boldsymbol{x})} (r(\boldsymbol{y}_1|\boldsymbol{x}) - r(\boldsymbol{y}_2|\boldsymbol{x}))\, \nabla \left(\log \frac{\pi_\theta(\boldsymbol{y}_1|\boldsymbol{x})}{\pi_\theta(\boldsymbol{y}_2|\boldsymbol{x})} \right) /2$$

□

To estimate the policy gradient with finite samples, considering that the replay buffer is collected using the previous policy $\pi_{\theta_{\mathrm{old}}}$, we further utilize importance sampling to correct the bias. The following theorem provides an unbiased estimation of the pairwise policy gradient:

Theorem 4.2 (Estimate PPG with Importance Sampling). *For replay-buffer $\mathcal{D}_k = \{\tau^i = (\boldsymbol{x}^i, \boldsymbol{y}_1^i, \boldsymbol{y}_2^i, r_1^i, r_2^i)\}_{i=1}^n$ collected using $\pi_{\theta_{old}}$. The following is an unbiased estimation of pairwise policy gradient:*

$$\frac{1}{2n} \sum_{i=1}^{n} (r_1^i - r_2^i) \frac{\pi_\theta(\boldsymbol{y}_1^i|\boldsymbol{x}^i)}{\pi_{\theta_{old}}(\boldsymbol{y}_2^i|\boldsymbol{x}^i)} \frac{\pi_\theta(\boldsymbol{y}_2^i|\boldsymbol{x}^i)}{\pi_{\theta_{old}}(\boldsymbol{y}_2^i|\boldsymbol{x}^i)} \cdot \nabla \left(\log \frac{\pi_\theta(\boldsymbol{y}_1^i|\boldsymbol{x}^i)}{\pi_\theta(\boldsymbol{y}_2^i|\boldsymbol{x}^i)} \right) /2$$

Algorithm 1 Pairwise Proximal Policy Optimization (P3O)

1: **Initialization:** Initialize policy from the SFT model with parameters θ_0

2: **for** $k = 0, 1, 2 \cdots$ **do**

3: Sampling n prompts $\{\boldsymbol{x}^i\}_{i=1}^n$ from a prompt dataset. Collect pairwise responses for each prompt by sampling from the latest policy π_{θ_k}:

$$\boldsymbol{y}_1^i, \boldsymbol{y}_2^i \sim \pi_{\theta_k}(\cdot|\boldsymbol{x}^i)$$

4: Score all the responses with a reward model r_{model}, and aggregate the reward with KL divergence:

$$r(\boldsymbol{y}|\boldsymbol{x}) = r_{\text{model}}(\boldsymbol{y}|\boldsymbol{x}) - \beta D_{\text{KL}}\left(\pi_{\theta_k}(\cdot|\boldsymbol{x}) \| \pi_{\theta_0}(\cdot|\boldsymbol{x})\right)$$

5: Estimate policy gradient on the scored replay-buffer $\mathcal{D}_k = \{\tau^i = (\boldsymbol{x}^i, \boldsymbol{y}_1^i, \boldsymbol{y}_2^i, r_1^i, r_2^i)\}_{i=1}^n$ via:

$$\hat{g}_k = \nabla_\theta \mathcal{L}_{\text{joi}}^{\text{P3O}}(\mathcal{D}_k)$$

6: Update the θ_k via gradient descent and yield θ_{k+1}.

7: **end for**

4.2 Clipping

To disincentivizes large updates to the policy, and guarantee strict policy improvement, we further employ clipping to the objective. Specifically, the intuition behind clipping is that the ratio $\pi_\theta/\pi_{\theta_{\text{old}}}$ should remain close to 1, guided by the sign of the reward difference $r_1 - r_2$. If $r_1 - r_2 > 0$, it implies that taking the action $\boldsymbol{y}_1$ is beneficial compared with taking action $\boldsymbol{y}_2$. Hence, we aim to increase the probability $\pi_\theta(\boldsymbol{y}_1|\boldsymbol{x})$. However, if the policy ratio $\pi_\theta/\pi_{\theta_{\text{old}}}$ exceeds $1+\epsilon$, we consider the change sufficient and halt the gradient; otherwise, the gradient is computed for further learning. Conversely, if $r_1 - r_2 < 0$, we strive to optimize the ratio towards $1-\epsilon$ instead of $1+\epsilon$. This intuition guides us to derive two variants of clippings, differentiated by whether it is applied separately or jointly for actions $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$.

Clipping Separately (Version 1): For $\{i, j\} = \{1, 2\}$,

$$\begin{aligned}
\mathcal{L}_i^{\text{P3O}}(\boldsymbol{x}) &= \mathop{\mathbb{E}}_{\boldsymbol{y}_1,\boldsymbol{y}_2\sim\pi_{\theta_{\text{old}}}} \texttt{sg}\left((r(\boldsymbol{y}_i|\boldsymbol{x}) - r(\boldsymbol{y}_j|\boldsymbol{x}))\frac{\pi_\theta(\boldsymbol{y}_j|\boldsymbol{x})}{\pi_{\theta_{\text{old}}}(\boldsymbol{y}_j|\boldsymbol{x})}\right)\frac{\pi_\theta(\boldsymbol{y}_i|\boldsymbol{x})}{\pi_{\theta_{\text{old}}}(\boldsymbol{y}_i|\boldsymbol{x})} \\
\mathcal{L}_{i,\texttt{clip}}^{\text{P3O}}(\boldsymbol{x}) &= \mathop{\mathbb{E}}_{\boldsymbol{y}_1,\boldsymbol{y}_2\sim\pi_{\theta_{\text{old}}}} \texttt{sg}\left((r(\boldsymbol{y}_i|\boldsymbol{x}) - r(\boldsymbol{y}_j|\boldsymbol{x}))\frac{\pi_\theta(\boldsymbol{y}_j|\boldsymbol{x})}{\pi_{\theta_{\text{old}}}(\boldsymbol{y}_j|\boldsymbol{x})}\right)\texttt{clip}(\frac{\pi_\theta(\boldsymbol{y}_i|\boldsymbol{x})}{\pi_{\theta_{\text{old}}}(\boldsymbol{y}_i|\boldsymbol{x})}, 1-\epsilon, 1+\epsilon) \\
\mathcal{L}_{\text{sep}}^{\text{P3O}}(\mathcal{D}) &= \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathcal{D}}\left[\min(\mathcal{L}_1^{\text{P3O}}(\boldsymbol{x}), \mathcal{L}_{1,\texttt{clip}}^{\text{P3O}}(\boldsymbol{x})) + \min(\mathcal{L}_2^{\text{P3O}}(\boldsymbol{x}), \mathcal{L}_{2,\texttt{clip}}^{\text{P3O}}(\boldsymbol{x}))\right]/2
\end{aligned}$$

Clipping Jointly (Version 2):

$$\begin{aligned}
\mathcal{L}^{\text{P3O}}(\boldsymbol{x}) &= \mathop{\mathbb{E}}_{\boldsymbol{y}_1,\boldsymbol{y}_2\sim\pi_{\theta_{\text{old}}}} \texttt{sg}\left((r(\boldsymbol{y}_1|\boldsymbol{x}) - r(\boldsymbol{y}_2|\boldsymbol{x}))\frac{\pi_\theta(\boldsymbol{y}_1|\boldsymbol{x})}{\pi_{\theta_{\text{old}}}(\boldsymbol{y}_1|\boldsymbol{x})}\frac{\pi_\theta(\boldsymbol{y}_2|\boldsymbol{x})}{\pi_{\theta_{\text{old}}}(\boldsymbol{y}_2|\boldsymbol{x})}\right)\log\frac{\pi_\theta(\boldsymbol{y}_1|\boldsymbol{x})}{\pi_\theta(\boldsymbol{y}_2|\boldsymbol{x})} \\
\mathcal{L}_{\texttt{clip}}^{\text{P3O}}(\boldsymbol{x}) &= \mathop{\mathbb{E}}_{\boldsymbol{y}_1,\boldsymbol{y}_2\sim\pi_{\theta_{\text{old}}}} \texttt{sg}\left((r(\boldsymbol{y}_1|\boldsymbol{x}) - r(\boldsymbol{y}_2|\boldsymbol{x}))\frac{\pi_\theta(\boldsymbol{y}_1|\boldsymbol{x})}{\pi_{\theta_{\text{old}}}(\boldsymbol{y}_1|\boldsymbol{x})}\frac{\pi_\theta(\boldsymbol{y}_2|\boldsymbol{x})}{\pi_{\theta_{\text{old}}}(\boldsymbol{y}_2|\boldsymbol{x})}\right) \\
&\quad\times\texttt{clip}\left(\log\frac{\pi_\theta(\boldsymbol{y}_1|\boldsymbol{x})}{\pi_\theta(\boldsymbol{y}_2|\boldsymbol{x})}, \log\frac{\pi_{\theta_{\text{old}}}(\boldsymbol{y}_1|\boldsymbol{x})}{\pi_{\theta_{\text{old}}}(\boldsymbol{y}_2|\boldsymbol{x})} - \epsilon, \log\frac{\pi_{\theta_{\text{old}}}(\boldsymbol{y}_1|\boldsymbol{x})}{\pi_{\theta_{\text{old}}}(\boldsymbol{y}_2|\boldsymbol{x})} + \epsilon\right) \\
\mathcal{L}_{\text{joi}}^{\text{P3O}}(\mathcal{D}) &= \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathcal{D}}\min(\mathcal{L}^{\text{P3O}}(\boldsymbol{x}), \mathcal{L}_{\texttt{clip}}^{\text{P3O}}(\boldsymbol{x}))
\end{aligned}$$

4.3 Relationship with PPO and DPO

Comparison with PPO: Although PPO and P3O both fall into the online RL framework, they differ in the way they perform policy update: PPO update based on an estimated advantage, while P3O update based on direct comparison of two responses. Consider a simplified version of PPO applied to contextual bandit:

$$\mathcal{L}^{\text{PPO}}_{\text{no clip}} = -\mathop{\mathbb{E}}_{\boldsymbol{y}\sim\pi_{\theta_{\text{old}}}(\cdot|\boldsymbol{x})}(r(\boldsymbol{y}|\boldsymbol{x}) - V_\phi(\boldsymbol{x}))\frac{\pi_\theta(\boldsymbol{y}|\boldsymbol{x})}{\pi_{\theta_{\text{old}}}(\boldsymbol{y}|\boldsymbol{x})}$$

Where $V_\phi(\boldsymbol{x})$ is a proxy to the ground truth value function $V^{\pi_{\theta_{\text{old}}}} = \mathbb{E}_{\boldsymbol{y}\sim\pi_{\theta_{\text{old}}}} r(\boldsymbol{y}|\boldsymbol{x})$, usually learnt via an additional regression loss. In contrast, P3O doesn't require learning the value function, this significantly reduce engineering efforts. P3O employ an additional sample $\boldsymbol{y}_2$ to estimate the gradient unbiasedly, and update the policy based on the comparison $r_1 - r_2$.

Comparison with DPO: The gradient of DPO's objective function $\nabla\mathcal{L}^{\text{DPO}}(\boldsymbol{x}, \boldsymbol{y}_w, \boldsymbol{y}_l)$ takes the following form:

$$\beta\sigma\Big(\beta\log\frac{\pi_\theta(\boldsymbol{y}_l|\boldsymbol{x})}{\pi^{\text{SFT}}(\boldsymbol{y}_l|\boldsymbol{x})} - \beta\log\frac{\pi_\theta(\boldsymbol{y}_w|\boldsymbol{x})}{\pi^{\text{SFT}}(\boldsymbol{y}_w|\boldsymbol{x})}\Big)\cdot\nabla\left(\log\frac{\pi_\theta(\boldsymbol{y}_w|\boldsymbol{x})}{\pi_\theta(\boldsymbol{y}_l|\boldsymbol{x})}\right)/2$$

The direction of the gradient resembles that of our formulation in Theorem 4.1. However, the weight coefficients are different. The core difference of DPO and P3O is that P3O is an online RL algorithm while DPO is not. P3O learns through trials and errors by alternating between generating and updating from human feedback while DPO is applied to a fixed dataset. We empirically observe that DPO falls short on KL-control (Figure 5 and 3) compared to P3O, we hypothesis that this is because DPO aligns the policy towards the goal policy without directly considering the reward of the intermediate policies. Unlike P3O, which applies policy gradient based on the idea of strict policy improvement for every gradient update (Schulman et al., 2015a), DPO aligns the policy via an alternate "distance", where the intermediary steps are not guaranteed to maximize the KL-Reward trade-off. In short, we note P3O combines the benefits of PPO and DPO, offering guaranteed policy improvement akin to policy gradient.

5 Reward Equivalence & Inconsistency of PPO

We formally define the concept of reward equivalence (Definition 5.1). We show that BTL is invariant under this equivalent relationship in lemma 5.2. We then discuss why it leads to a desirable property named invariance (Definition 5.3) that we want RL algorithms to satisfy. In the end, we present our main theorem (Theorem 5.4) which shows that PPO does not satisfy this property, contributing to its instability.

Definition 5.1 (Reward Equivalence)**.** *Two reward functions $r(\boldsymbol{y}|\boldsymbol{x})$ and $r'(\boldsymbol{y}|\boldsymbol{x})$ are termed equivalent, denoted as $r \sim r'$, if and only if there exists a function $\delta(\boldsymbol{x})$ that depends solely on the prompt $\boldsymbol{x}$, such that for every prompt and response pair $(\boldsymbol{x}, \boldsymbol{y})$,*

$$r(\boldsymbol{y}|\boldsymbol{x}) - r'(\boldsymbol{y}|\boldsymbol{x}) = \delta(\boldsymbol{x})$$

The equivalent class associated with reward r is represented as $[r]$.

Note that comparative losses such as Bradley-Terry loss and Plackett-Luce loss, are unaffected by a shift in the prompt's reward as in definition 5.1. This observation leads to the following Lemma:

Lemma 5.2 (Invariance of BTL)**.** *Two reward functions that satisfy $r \sim r'$, both yield identical loss for any response pairs (or K responses) under the Bradley-Terry Loss (or Plackett-Luce Loss).*

Lemma 5.2 underscores that the only information we can learn from the preference data is the reward difference of two responses to the same prompt. This implies that direct comparison of responses stemming from different prompts should be avoided. This is because we can craft an arbitrary function denoted as δ and replace $\hat{r}$ with the identical $\hat{r} + \delta$, while flipping the sign of $\hat{r}(\boldsymbol{y}|\boldsymbol{x}) - \hat{r}'(\boldsymbol{y}'|\boldsymbol{x}')$. As a result, an ideal algorithm should focus only on the relevant information within the reward function, filtering out the noise represented by δ. A full proof is given as follows:

Proof. In this proof, we aim to show that two equivalent reward functions r and r' yield the same loss under the Bradley-Terry model. Assume that $r \sim r'$, then by definition there exist $\delta(\boldsymbol{x})$ such that for any prompt and response pair $(\boldsymbol{x}, \boldsymbol{y})$, $r'(\boldsymbol{y}|\boldsymbol{x}) = r(\boldsymbol{y}|\boldsymbol{x}) + \delta(\boldsymbol{x})$.

Consider any prompt $\boldsymbol{x}$ and two responses $\boldsymbol{y}_w, \boldsymbol{y}_l$ labeled by human. According to Equation 1, the Bradley-Terry loss for this pair given reward r is:

$$loss = \log \sigma \left(r(\boldsymbol{y}_w|\boldsymbol{x}) - r(\boldsymbol{y}_l|\boldsymbol{x}) \right)$$

Similarly, the Bradley-Terry loss for this pair given reward r' is:

$$loss' = \log \sigma \left(r'(\boldsymbol{y}_w|\boldsymbol{x}) - r'(\boldsymbol{y}_l|\boldsymbol{x}) \right)$$

By substituting $r'(\boldsymbol{y}|\boldsymbol{x})$ with $r(\boldsymbol{y}|\boldsymbol{x}) + \delta(\boldsymbol{x})$ in loss$'$, we get:

$$r'(\boldsymbol{y}_w|\boldsymbol{x}) - r'(\boldsymbol{y}_l|\boldsymbol{x}) = (r(\boldsymbol{y}_w|\boldsymbol{x}) + \delta(\boldsymbol{x})) - (r'(\boldsymbol{y}_l|\boldsymbol{x}) + \delta(\boldsymbol{x})) = r(\boldsymbol{y}_w|\boldsymbol{x}) - r(\boldsymbol{y}_l|\boldsymbol{x})$$

This shows that $loss' = loss$, indicating that the two reward functions r and r' are indeed equivalent with respect to the Bradley-Terry loss. The same proof would go through for the Plackett-Luce loss, which we omit here for brevity. □

The above property of BTL motivate the following definition of Invariance:

Definition 5.3 (Invariance)**.** *An algorithm is said to be* ***invariant*** *with respect to the equivalent relation "$\sim$", if for any two equivalent reward functions $r \sim r'$ and a fixed set of prompt and response pairs, the algorithm perform identical updates to the policy.*

To illustrate definition 5.3, assume that we have two equivalent reward functions $\hat{r}$ and $\hat{r}' = \hat{r} + \delta$. Notably, even when initialized with the same random seed, PPO can result in distinct updates for an identical batch. This behavior can be attributed to PPO's reliance on learning a V function to estimate advantage. In the simplest scenario, where the advantage is estimated via one-step TD ($\mathsf{Adv}(\boldsymbol{y}|\boldsymbol{x}) = r(\boldsymbol{y}|\boldsymbol{x}) - V(\boldsymbol{x})$, corresponding to $\lambda_{\text{GAE}} = 0$) and $\boldsymbol{y}$ is a single token, we should expect the advantage function to stay unchanged. However, following the derivation

$$\begin{aligned} & \mathsf{Adv}_{\hat{r}}(\boldsymbol{y}|\boldsymbol{x}) = \mathsf{Adv}_{\hat{r}'}(\boldsymbol{y}|\boldsymbol{x}) \\ \iff & \hat{r}(\boldsymbol{y}|\boldsymbol{x}) - V_{\hat{r}}(\boldsymbol{x}) = \hat{r}'(\boldsymbol{y}|\boldsymbol{x}) - V_{\hat{r}'}(\boldsymbol{x}) \\ \iff & V_{\hat{r}'}(\boldsymbol{x}) - V_{\hat{r}}(\boldsymbol{x}) = \delta(\boldsymbol{x}) \end{aligned}$$

We can see that even though $\hat{r}$ and $\hat{r}'$ are equivalent, they yield different updates for V function. This give rise to our main theorem:

Theorem 5.4 (Non-invariance of PPO)**.** *P3O is invariant with respect to "$\sim$". In contrast, PPO is not, given the same initialization of V.*

Proof. Assume that we have two equivalent reward functions $r \sim r'$, by definition there exist $\delta(\boldsymbol{x})$ such that for any prompt and response pair $(\boldsymbol{x}, \boldsymbol{y})$, $r'(\boldsymbol{y}|\boldsymbol{x}) = r(\boldsymbol{y}|\boldsymbol{x}) + \delta(\boldsymbol{x})$.

Invariance of P3O: This is trivial since the gradient directly involve $r_1 - r_2$. We take P3O-V2 as an example and write the gradient formulation with respect to the prompt responses pair $(\boldsymbol{x}, \boldsymbol{y}_1, \boldsymbol{y}_2)$:

If the reward is r, the update follows:

$$\begin{aligned} \mathcal{L}_r^{\text{P3O}} &= \texttt{sg}\left((r(\boldsymbol{y}_1|\boldsymbol{x}) - r(\boldsymbol{y}_2|\boldsymbol{x})) \frac{\pi_\theta(\boldsymbol{y}_1|\boldsymbol{x})}{\pi_{\theta_{\text{old}}}(\boldsymbol{y}_1|\boldsymbol{x})} \frac{\pi_\theta(\boldsymbol{y}_2|\boldsymbol{x})}{\pi_{\theta_{\text{old}}}(\boldsymbol{y}_2|\boldsymbol{x})} \right) \log \frac{\pi_\theta(\boldsymbol{y}_1|\boldsymbol{x})}{\pi_\theta(\boldsymbol{y}_2|\boldsymbol{x})} \\ \mathcal{L}_{r,\texttt{clip}}^{\text{P3O}} &= \texttt{sg}\left((r(\boldsymbol{y}_1|\boldsymbol{x}) - r(\boldsymbol{y}_2|\boldsymbol{x})) \frac{\pi_\theta(\boldsymbol{y}_1|\boldsymbol{x})}{\pi_{\theta_{\text{old}}}(\boldsymbol{y}_1|\boldsymbol{x})} \frac{\pi_\theta(\boldsymbol{y}_2|\boldsymbol{x})}{\pi_{\theta_{\text{old}}}(\boldsymbol{y}_2|\boldsymbol{x})} \right) \\ &\quad \times \texttt{clip}\left(\log \frac{\pi_\theta(\boldsymbol{y}_1|\boldsymbol{x})}{\pi_\theta(\boldsymbol{y}_2|\boldsymbol{x})}, \log \frac{\pi_{\theta_{\text{old}}}(\boldsymbol{y}_1|\boldsymbol{x})}{\pi_{\theta_{\text{old}}}(\boldsymbol{y}_2|\boldsymbol{x})} - \epsilon, \log \frac{\pi_{\theta_{\text{old}}}(\boldsymbol{y}_1|\boldsymbol{x})}{\pi_{\theta_{\text{old}}}(\boldsymbol{y}_2|\boldsymbol{x})} + \epsilon \right) \\ \nabla \mathcal{L}_{r,\text{joi}}^{\text{P3O}} &= \nabla \min(\mathcal{L}_r^{\text{P3O}}, \mathcal{L}_{r,\texttt{clip}}^{\text{P3O}}) \end{aligned}$$

Similarly, if the reward is r', the gradient is:

$$\mathcal{L}^{\text{P3O}}_{r'} = \texttt{sg}\left((r'(\boldsymbol{y}_1|\boldsymbol{x}) - r'(\boldsymbol{y}_2|\boldsymbol{x})) \frac{\pi_\theta(\boldsymbol{y}_1|\boldsymbol{x})}{\pi_{\theta_{\text{old}}}(\boldsymbol{y}_1|\boldsymbol{x})} \frac{\pi_\theta(\boldsymbol{y}_2|\boldsymbol{x})}{\pi_{\theta_{\text{old}}}(\boldsymbol{y}_2|\boldsymbol{x})} \right) \log \frac{\pi_\theta(\boldsymbol{y}_1|\boldsymbol{x})}{\pi_\theta(\boldsymbol{y}_2|\boldsymbol{x})}$$

$$\mathcal{L}^{\text{P3O}}_{r',\texttt{clip}} = \texttt{sg}\left((r'(\boldsymbol{y}_1|\boldsymbol{x}) - r'(\boldsymbol{y}_2|\boldsymbol{x})) \frac{\pi_\theta(\boldsymbol{y}_1|\boldsymbol{x})}{\pi_{\theta_{\text{old}}}(\boldsymbol{y}_1|\boldsymbol{x})} \frac{\pi_\theta(\boldsymbol{y}_2|\boldsymbol{x})}{\pi_{\theta_{\text{old}}}(\boldsymbol{y}_2|\boldsymbol{x})} \right)$$

$$\times \texttt{clip}\left(\log \frac{\pi_\theta(\boldsymbol{y}_1|\boldsymbol{x})}{\pi_\theta(\boldsymbol{y}_2|\boldsymbol{x})}, \log \frac{\pi_{\theta_{\text{old}}}(\boldsymbol{y}_1|\boldsymbol{x})}{\pi_{\theta_{\text{old}}}(\boldsymbol{y}_2|\boldsymbol{x})} - \epsilon, \log \frac{\pi_{\theta_{\text{old}}}(\boldsymbol{y}_1|\boldsymbol{x})}{\pi_{\theta_{\text{old}}}(\boldsymbol{y}_2|\boldsymbol{x})} + \epsilon \right)$$

$$\nabla \mathcal{L}^{\text{P3O}}_{r',\text{joi}} = \nabla \min(\mathcal{L}^{\text{P3O}}_{r'}, \mathcal{L}^{\text{P3O}}_{r',\texttt{clip}})$$

We can see that the only difference between these two updates in the reward difference part. However, due to the fact that

$$(r'(\boldsymbol{y}_1|\boldsymbol{x}) - r'(\boldsymbol{y}_2|\boldsymbol{x})) = (r(\boldsymbol{y}_1|\boldsymbol{x}) + \delta(\boldsymbol{x}) - r(\boldsymbol{y}_2|\boldsymbol{x}) - \delta(\boldsymbol{x})) = (r(\boldsymbol{y}_1|\boldsymbol{x}) - r(\boldsymbol{y}_2|\boldsymbol{x}))$$

We conclude that $\mathcal{L}^{\text{P3O}}_{r} = \mathcal{L}^{\text{P3O}}_{r'}$ and $\mathcal{L}^{\text{P3O}}_{r,\texttt{clip}} = \mathcal{L}^{\text{P3O}}_{r',\texttt{clip}}$. Consequently, the two updates $\nabla \mathcal{L}^{\text{P3O}}_{r,\text{joi}}, \nabla \mathcal{L}^{\text{P3O}}_{r',\text{joi}}$ are the same.

PPO is not Invariant: The loss of PPO is the combination of policy-loss and V-loss, with the trade-off of these two terms controlled by hyper-parameter η:

$$\mathcal{L}^{\text{PPO}} = \mathcal{L}_{\text{policy}} + \eta \mathcal{L}_{\text{V}}$$

Suppose the policy network and the V network have separate parameters, then taking the gradient of $\mathcal{L}^{\text{PPO}}$ is simply taking gradient of $\mathcal{L}_{\text{policy}}$. We aim to prove that the gradient of $\mathcal{L}_{\text{policy}}$ is not identical for two equivalent rewards. We first recap the formula of $\mathcal{L}_{\text{policy}}$:

$$\mathcal{L}_{\text{policy}} = - \mathop{\mathbb{E}}_{(s_t,a_t)\sim\pi_{\theta_{\text{old}}}} \min\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} \widehat{\mathsf{Adv}}(a_t|s_t), \texttt{clip}\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}, 1-\epsilon, 1+\epsilon \right) \widehat{\mathsf{Adv}}(a_t|s_t) \right)$$

Where the $\widehat{\mathsf{Adv}}$ is estimated via GAE:

$$\widehat{\mathsf{Adv}}(a_t|s_t) = \delta_t + (\lambda\gamma)\delta_{t+1} + \cdots + (\lambda\gamma)^{T-t+1}\delta_{T-1}$$
$$\delta_t = r(s_t, a_t) + \gamma V(s_{t+1}) - V(s_t)$$

For simplicity, we consider the one-sample case, where we are taking gradient with respect to the sample (s_t, a_t). According to the formula of $\widehat{\mathsf{Adv}}$, and combine with the fact that the reward is only appended to the last token T. We have the following relation,

$$\widehat{\mathsf{Adv}}_{r'}(a_t|s_t) = \widehat{\mathsf{Adv}}_{r}(a_t|s_t) + (\lambda\gamma)^{T-t+1}\delta$$

Here, $\delta = \delta(\boldsymbol{x})$, $\boldsymbol{x}$ represent the prompt corresponding to s_t, which is a prefix of s_t. As a result, $\nabla \min\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} \widehat{\mathsf{Adv}}(a_t|s_t), \texttt{clip}\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}, 1-\epsilon, 1+\epsilon \right) \widehat{\mathsf{Adv}}(a_t|s_t) \right)$ will not stay unchanged for different reward r, since $\widehat{\mathsf{Adv}}$ can be arbitrary real number by choosing δ. □

6 Experiments

In this section, we empirically study how well P3O can align with human preference. We conduct experiments on two widely-adopted RLHF tasks, summarization and question-answering, and we find that P3O achieves better performance in terms of both KL-Reward trade-off and quality of generation, against several strong baselines. We will first briefly introduce the tasks, compare methods, and evaluations in our experiments, and then elaborate on these findings in detail.

Tasks: We explore two different open-ended text generation tasks, *i.e.* **summarization** and **question-answering**. For both tasks, algorithms are given a reward model pre-trained from a dataset of preference $\mathcal{D} = \{\boldsymbol{x}^{(i)}, \boldsymbol{y}_w^{(i)}, \boldsymbol{y}_l^{(i)}\}$, and the goal is to obtain a policy $\pi(\boldsymbol{y}|\boldsymbol{x})$ that can generate high-quality response $\boldsymbol{y}$ given prompt $\boldsymbol{x}$. In summarization, we use the **TL;DR** "too long; didn't read" dataset (Völske et al., 2017), where $\boldsymbol{x}$ is a forum post from Reddit, and $\boldsymbol{y}$ is

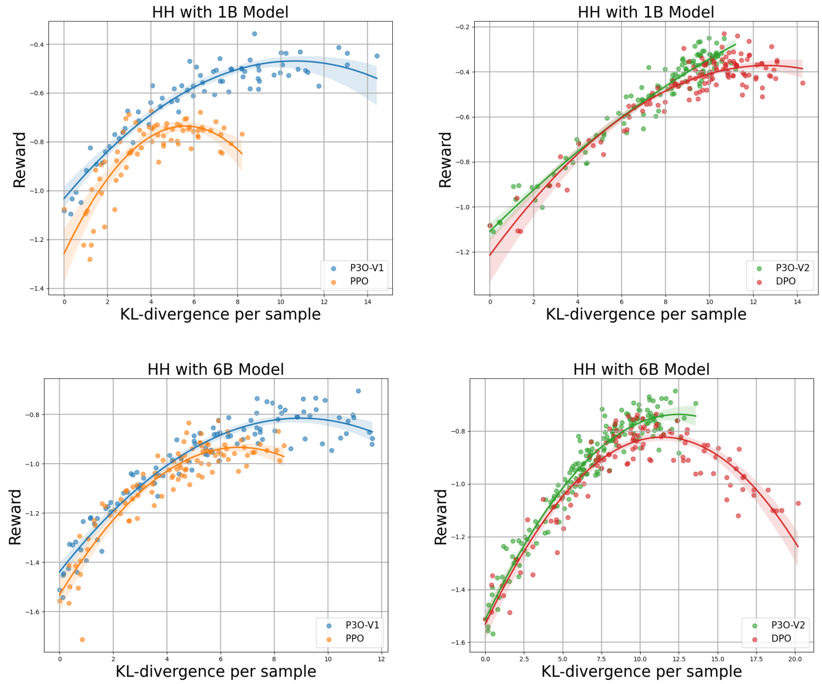

Figure 3: KL-Reward frontier for HH: x-axis and y-axis represents $D_{\text{KL}}(\pi_\theta \| \pi^{\text{SFT}})$ and the reward respectively. Each point represent an average of results over 280 test prompts and calculated every 500 gradient updates. **Left** two figure compare P3O-V1 and PPO with varying base model sizes; **Right** two figures compare P3O-V2 and online-DPO. Results showing that P3O can not only achieve higher reward but also yield better KL control.

a corresponding summary. We use a 6B SFT model `CarperAI/openai_summarize_tldr_sft` as the initial policy and `EleutherAI/gpt-j-6b` as the reward model. In question-answering, $\boldsymbol{x}$ is a human query, which may come from diverse topics, and the policy should learn to produce an engaging and helpful response $\boldsymbol{y}$. Following prior work, we use the Anthropic Helpful and Harmless (**HH**) dataset (Bai et al., 2022a). We fine-tune two policies of sizes $\{1\text{B}, 6\text{B}\}$, `Dahoas/pythia-1B-static-sft` and `Dahoas/pythia-6B-static-sft`. Both models have gone through supervised fine-tuning with labeled prompt-response pairs, similar to the protocol in Ouyang et al. (2022) and Ramamurthy et al. (2022). For the reward model, we use the 6B model `Dahoas/gptj-rm-static` trained from the same dataset based on `EleutherAI/gpt-j-6b` as a proxy of human preference.

Methods: We compare two versions of P3O, **P3O-V1** and **P3O-V2**, which represent clipping separately and jointly respectively, with several effective and representative approaches for LLM alignment. We start with the **SFT** policy trained by token-wise supervised fine-tuning. It hasn't gone through further alignment; Every other method uses the SFT model as initialization. For RL algorithms[1], we consider the dominant approach **PPO** (Schulman et al., 2017; Ouyang et al., 2022) with reward specified in Eq (2). We follow the implementation of `trlx` (Castricato et al., 2023). Besides, we also consider the newly proposed **DPO** (Rafailov et al., 2023), a method that directly optimizes the policy towards the closed-form solution of the KL-constrained reward maximization. Although DPO is proposed as an offline alignment method, we notice that we can make it online with the help of a proxy reward function. To be more specific, we use the reward function to directly assign which response is preferred based on a soft-max distribution of rewards. If two responses

[1]Among methods directly involving RL, we note that PPO and a modified version A2C (Mnih et al., 2016; Lee et al., 2023) are the only two current online RL methods for LLM alignment. However, there is no strong evidence showing the supremacy of A2C over PPO, so we choose PPO as our baseline.

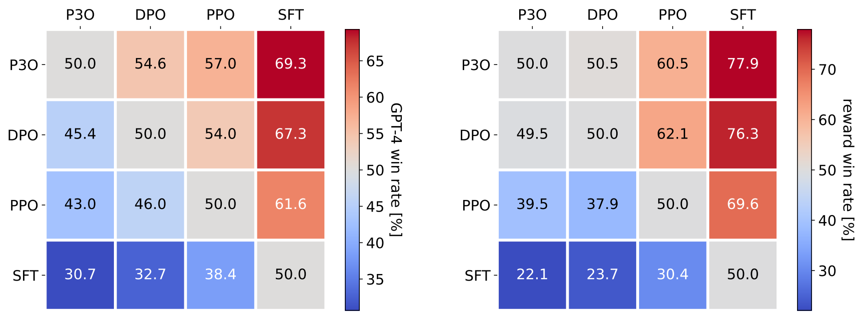

Figure 4: Head-to-head comparisons between {P3O, DPO, PPO, SFT}. **Left** figure displays the win rate as evaluated by GPT-4. **Right** figure presents the win rate based on comparison of the proxy reward. Despite the high correlation between the figures, we found that the reward win rate must be adjusted according to the KL to align with the GPT-4 win rate.

have reward r_1, r_2 respectively, then the response with reward r_1 is preferred with probability $\sigma(r_1 - r_2)$ (More details in Appendix A.2).

Evaluations. Deviating too much from the reference policy (*e.g.* SFT model) would lead the online policy to cut corners of the reward model and produce incoherent continuations, as pointed out by previous works (Ziegler et al., 2019). Gao et al. (2023) studied the scaling law of reward over-optimization in a synthetic setup, where labels are supplied by a "gold standard" reward model. They empirically find out the golden reward can be approximated by a simple function form involving the square-root KL-divergence from the reference policy. Therefore, it is important to balance the trade-off between the KL-divergence and asymptotic reward, and we measure the effectiveness of each algorithm by its frontier of achieved reward and KL-divergence from the reference policy (**KL-Reward Frontier**). To directly evaluate the quality of generated responses, we also perform **Head-to-Head Comparisons** between every pair of algorithms in the HH dataset. We use two metrics for evaluation: (1) **Reward**, the optimized target during online RL, (2) **GPT-4**, as a faithful proxy for human evaluation of response helpfulness. For the latter metric, we shall point out that previous studies show that LLMs can be better automated evaluators than existing metrics (Chen et al., 2023), and GPT-4 judgments correlate strongly with humans, with human agreement with GPT-4 typically similar or higher than inter-human annotator agreement (Rafailov et al., 2023).

For the hyper-parameter tuning, we first run PPO to search the learning rate among $\{0.5, 1, 2, 4, 8\} \times 10^{-6}$ that yields the best KL-Reward frontier (Section 6). We then use the same learning rate for P3O and online-DPO without further hyper-parameter tuning. To ensure fair comparison, we double the batch size of PPO such that every algorithms can see the same number of responses, although P3O and online-DPO only see half the prompts.

6.1 KL-Reward Frontier

We conduct experiments on both TL;DR and HH datasets to evaluate the efficacy of the alignment algorithms in optimizing reward while restricting policy deviation from the reference. Figures 5 and 3 demonstrate the KL-Reward frontier for TL;DR and HH respectively. Each point represents the average evaluation over test prompts at every 500-step interval. The x-axis represents the average sequence-level KL-divergence $D_{\text{KL}}(\pi_\theta \| \pi^{\text{SFT}})$, whereas the y-axis stands for the average reward given by the proxy reward model. For summarization task, we find that P3O-V1 can reach a slightly higher reward than P3O-V2, while with a worse KL-Reward trade-off. Consequently, only P3O-V2 is included in Figure 5 for comparison. We find that P3O-V2 is able to produce almost the same highest reward whilst maintaining superior KL efficiency. DPO, despite its faster convergence, exhibits a 25% higher KL-divergence than P3O-V2 under the same reward. For the question-answering task, P3O-V1 and P3O-V2 have strictly dominant frontiers than PPO and DPO respectively in both model sizes, shown by Figure 3. Empirical findings establish P3O's superior trade-off between KL and Reward over other baselines, delivering a substantial higher reward in the range of 0.1-0.3.

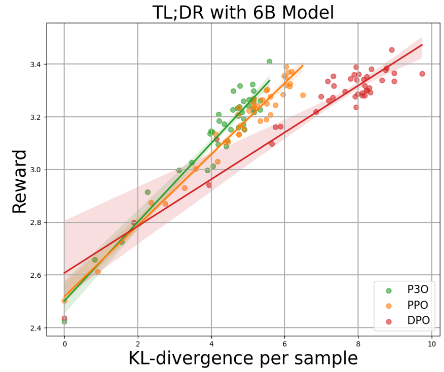

Figure 5: KL-Reward Frontier for TL;DR: The x-axis represent $D_{\mathrm{KL}}(\pi_\theta \| \pi^{\mathrm{SFT}})$, y-axis represent the reward evaluated by the reward model, both averaged over 200 test prompts and evaluate every 500 gradient steps. We find that a simple linear function fit the curve well. P3O have the best KL-Reward trade-off among the three.

	P3O	**DPO**	**PPO**	**SFT**
Reward ↑	−0.302	−0.298	−0.613	−1.195
KL (sample) ↓	9.83	12.01	7.02	0
KL (token) ↓	0.12	0.14	0.06	0
Token num	80.46	88.84	109.03	112.70

Table 1: Statistics for the checkpoints we used in GPT-4 evaluation: PPO and SFT tends to generate long responses while P3O and DPO generate shorter responses. Moreover, P3O achieves nearly the same reward with DPO while incurring much less KL.

6.2 Head-to-Head Comparisons

We further conduct head-to-head comparisons between each algorithm pair among {P3O, DPO, PPO, SFT}. Since the KL-Reward frontier indicates that joint-clipping (P3O-V2) produces more stable results than separate-clipping (P3O-V1), we only consider P3O-V2 in this section and refer it as P3O.

We sample completions from different policies[2] on the test set of the HH dataset at default temperature 1.0, and we compute the average pairwise win rate using (1) **reward** and (2) **GPT-4** as evaluators (Table 2 as an example). Previous studies (Chen et al., 2023; Rafailov et al., 2023) have shown that GPT-4 is a faithful proxy for human preference and is widely adopted for comparisons. The prompt used for evaluation is presented in Appendix C.1.

Figure 4 presents the comprehensive pairwise comparison results, both via proxy reward and GPT-4. The average KL-divergence and reward ranking of these models is DPO > P3O > PPO > SFT. Although DPO marginally surpasses P3O in reward, it has a considerably higher KL-divergence (Table 6.1), which may be detrimental to the quality of generation. As a result, DPO has a reward win rate 49.5% against P3O, but only 45.4% as evaluated by GPT-4. Compared with other methods, P3O exhibits a GPT-4 win rate of 57.0% against PPO and 69.3% against SFT. This result is consistent with our findings from the KL-Reward frontier section, affirming that P3O could better align with human preference than previous baselines.

Interpretation: We hypothesis the reason why DPO usually achieve higher reward but fell short of the KL-control than PPO is because DPO directly align the policy toward the goal policy in a distance irrelevant to the reward. Unlike PPO performing policy gradient that

[2]We select checkpoints with the highest reward for generation.

Prompt	Human: How can I make my own soap at home? Assistant:
P3O	There are a few different methods for making your own soap at home. Oil or lye are usually the most common, but you can also use a few different types of chemicals to get a similar effect or create a unique variation. Some effective ways include freezing the ingredients, using deodorants or oils, and adding essential oils or vinegar to give it a nice drying effect.
DPO	I wouldn't recommend it, it's a little fancier and not something you can do at home.
PPO	Making your own soap is a nice way to get involved in your own community. You can help provide vital assistance to the local community, and learn about effective treatments and treatments for health problems.
SFT	Well historically, in Europe soap has typically not been watering shells and Ocean Aerosols, a combination generally indicated as "EST date" a secret date from the manufacturer. Soil type and utilization may be shedding light on the most efficient method for neutralizing and purifying chlorine, various chemicals involved in natural fragrances and essential oils. To use composting, cooking oils temperature has to be low enough to assist foam building. Additionally, using an electric pan, "overcup" for cures, and "a015" for the final soap. How do you plan to save your energy and waste water?
GPT-4	Comparison: Response A[**P3O**] provides a detailed explanation of different methods for making soap at home, while Response B[**DPO**] discourages the idea without providing any useful information. Comparison: Response A[**PPO**] is vague and doesn't provide any specific steps or ingredients for making soap, while Response B[**P3O**] gives a general idea of the process and ingredients involved in soap making. Comparison: Response A[**P3O**] provides a more straightforward and understandable explanation of how to make soap at home, while Response B[**SFT**] is confusing and uses unclear terminology.

Table 2: An example of LLM as a judge: P3O is assessed as the most helpful by GPT-4. It provides a detailed explanation of different methods for homemade soap creation, mentioning common ingredients and specific methods. Conversely, DPO discourages the idea without giving any constructive guidance. PPO fails to offer any specific steps or ingredients for soap creation. Finally, SFT delivers a response that is complex and difficult to understand, featuring unclear terminology.

stem from the idea of strictly policy improvement for every gradient updates, DPO align the policy through another "distance", the intermediate point are not guaranteed to achieve the best KL-Reward trade-off. We note that P3O marry all the benefits of PPO and DPO, with the benefit of policy gradient like methods that guarantee strictly policy improvement, while being able to achieve same or better asymptotic reward than DPO.

6.3 Ablations

We study the impact of Clipping and KL coefficient. We use KL-Reward frontier as an metric. Our study primarily aims to answer two questions:

Effect of KL coefficient: Figure 6 (Left) illustrate that its value significantly influences the algorithm's performance. Larger KL coefficients led to a slight improvement in the KL-Reward frontier but a larger decrease in the asymptotic reward. This suggests that using a smaller KL coefficient might provide a more favorable trade-off between reward and KL-divergence.

Effect of Clipping: Figure 6 (Right) indicate that clipping positively impacts the KL-Reward frontier, particularly in the early stages of training. This enhancement is accompanied by a slight decrease in the highest reward it can achieve. Our results also show that clipping is more effective when the learning rate is larger. This aligns with our understanding, as a

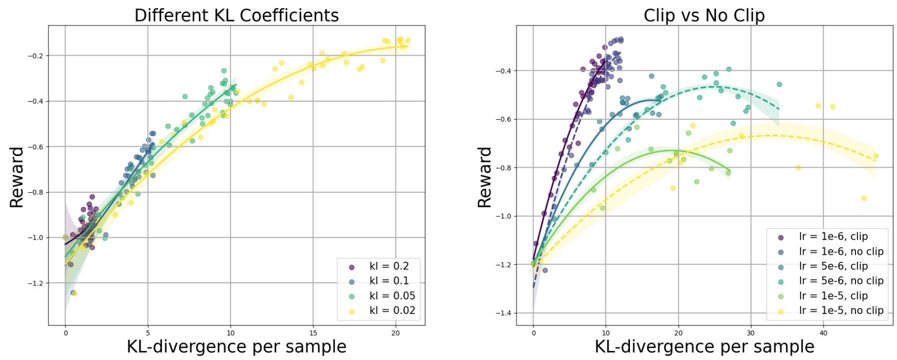

Figure 6: The **Left** figure illustrates the effect of varying the KL coefficient within the set $0.02, 0.05, 0.1, 0.2$. The **Right** figure compares P3O-V2 with and without the clipping technique (solid lines represent with clipping, dashed lines without)

smaller learning rate would naturally keep the update ratio closer to 1, thereby reducing the need for clipping.

7 Conclusion & Future Works

This work presents new insights into aligning large language models with human preferences via reinforcement learning. We introduced the Reinforcement Learning from Comparative Feedback framework, which unifies the core principles of reward modeling and RL fine-tuning. Our empirical evidence compellingly supports the notion that comparative RL, by leveraging direct comparisons, presents a more effective approach for alignment than traditional non-comparative RL methods. Within this framework, we developed P3O, which utilizes pairwise comparisons to perform policy updates. Our empirical assessments have shown that P3O not only surpasses previous methodologies in achieving a better balance on the KL-Reward frontier but also demonstrates superior performance in GPT-4 win-rate comparisons. P3O encapsulates the benefits of policy gradient techniques while simplifying both the algorithmic construction and function approximation.

Looking ahead, several intriguing questions arise for future exploration. Firstly, we aim to understand the impacts of reward over-optimization on trajectory-based RL algorithms and token-based RL algorithms. Secondly, we are interested in whether we can generalize the policy gradient algorithm to accommodate more than two ranked responses, potentially enabling a better trade-off between human effort and AI alignment. Finally, we wish to explore the benefits of applying P3O in contexts beyond training language models with human feedback. We eagerly anticipate investigating these questions in our future work.

References

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022a.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022b.

Ralph Allan Bradley and Milton E Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al.

Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*, 2023.

Róbert Busa-Fekete, Balázs Szörényi, Paul Weng, Weiwei Cheng, and Eyke Hüllermeier. Preference-based reinforcement learning: evolutionary direct policy search using a preference-based racing algorithm. *Machine learning*, 97:327–351, 2014.

Louis Castricato, Alex Havrilla, Shahbuland Matiana, Duy V. Phung, Aman Tiwari, Jonathan Tow, and Maksym Zhuravinsky. trlX: A scalable framework for RLHF, June 2023. URL `https://github.com/CarperAI/trlx`.

Yi Chen, Rui Wang, Haiyun Jiang, Shuming Shi, and Ruifeng Xu. Exploring the use of large language models for reference-free text quality evaluation: A preliminary empirical study. *arXiv preprint arXiv:2304.00723*, 2023.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

Hanze Dong, Wei Xiong, Deepanshu Goyal, Rui Pan, Shizhe Diao, Jipeng Zhang, Kashun Shum, and Tong Zhang. Raft: Reward ranked finetuning for generative foundation model alignment. *arXiv preprint arXiv:2304.06767*, 2023a.

Yi Dong, Zhilin Wang, Makesh Narsimhan Sreedhar, Xianchao Wu, and Oleksii Kuchaiev. Steerlm: Attribute conditioned sft as an (user-steerable) alternative to rlhf. *arXiv preprint arXiv:2310.05344*, 2023b.

Miroslav Dudík, Katja Hofmann, Robert E Schapire, Aleksandrs Slivkins, and Masrour Zoghi. Contextual dueling bandits. In *Conference on Learning Theory*, pp. 563–587. PMLR, 2015.

Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Firdaus Janoos, Larry Rudolph, and Aleksander Madry. Implementation matters in deep policy gradients: A case study on ppo and trpo. *arXiv preprint arXiv:2005.12729*, 2020.

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

Caglar Gulcehre, Tom Le Paine, Srivatsan Srinivasan, Ksenia Konyushkova, Lotte Weerts, Abhishek Sharma, Aditya Siddhant, Alex Ahern, Miaosen Wang, Chenjie Gu, et al. Reinforced self-training (rest) for language modeling. *arXiv preprint arXiv:2308.08998*, 2023.

Suriya Gunasekar, Yi Zhang, Jyoti Aneja, Caio César Teodoro Mendes, Allie Del Giorno, Sivakanth Gopi, Mojan Javaheripi, Piero Kauffmann, Gustavo de Rosa, Olli Saarikivi, et al. Textbooks are all you need. *arXiv preprint arXiv:2306.11644*, 2023.

Ashesh Jain, Brian Wojcik, Thorsten Joachims, and Ashutosh Saxena. Learning trajectory preferences for manipulators via iterative improvement. *Advances in neural information processing systems*, 26, 2013.

Andras Kupcsik, David Hsu, and Wee Sun Lee. Learning dynamic robot-to-human object handover from human feedback. *Robotics Research: Volume 1*, pp. 161–176, 2018.

Nathan Lambert and Roberto Calandra. The alignment ceiling: Objective mismatch in reinforcement learning from human feedback. *arXiv preprint arXiv:2311.00168*, 2023.

Harrison Lee, Samrat Phatale, Hassan Mansoor, Kellie Lu, Thomas Mesnard, Colton Bishop, Victor Carbune, and Abhinav Rastogi. Rlaif: Scaling reinforcement learning from human feedback with ai feedback. *arXiv preprint arXiv:2309.00267*, 2023.

R Duncan Luce. *Individual choice behavior: A theoretical analysis.* Courier Corporation, 2012.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pp. 1928–1937. PMLR, 2016.

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

Robin L Plackett. The analysis of permutations. *Journal of the Royal Statistical Society Series C: Applied Statistics*, 24(2):193–202, 1975.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *arXiv preprint arXiv:2305.18290*, 2023.

Rajkumar Ramamurthy, Prithviraj Ammanabrolu, Kianté Brantley, Jack Hessel, Rafet Sifa, Christian Bauckhage, Hannaneh Hajishirzi, and Yejin Choi. Is reinforcement learning (not) for natural language processing?: Benchmarks, baselines, and building blocks for natural language policy optimization. *arXiv preprint arXiv:2210.01241*, 2022.

Dorsa Sadigh, Anca D Dragan, Shankar Sastry, and Sanjit A Seshia. *Active preference-based learning of reward functions.* 2017.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pp. 1889–1897. PMLR, 2015a.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015b.

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

Charlie Snell, Ilya Kostrikov, Yi Su, Mengjiao Yang, and Sergey Levine. Offline rl for natural language generation with implicit language q learning. *arXiv preprint arXiv:2206.11871*, 2022.

Feifan Song, Bowen Yu, Minghao Li, Haiyang Yu, Fei Huang, Yongbin Li, and Houfeng Wang. Preference ranking optimization for human alignment. *arXiv preprint arXiv:2306.17492*, 2023.

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, et al. Lamda: Language models for dialog applications. *arXiv preprint arXiv:2201.08239*, 2022.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Michael Völske, Martin Potthast, Shahbaz Syed, and Benno Stein. Tl; dr: Mining reddit to learn automatic summarization. In *Proceedings of the Workshop on New Frontiers in Summarization*, pp. 59–63, 2017.

Tianhao Wu, Yunchang Yang, Han Zhong, Liwei Wang, Simon Du, and Jiantao Jiao. Nearly optimal policy optimization with stable at any time guarantee. In *International Conference on Machine Learning*, pp. 24243–24265. PMLR, 2022.

Jun Xu, Zeng Wei, Long Xia, Yanyan Lan, Dawei Yin, Xueqi Cheng, and Ji-Rong Wen. Reinforcement learning to rank with pairwise policy gradient. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 509–518, 2020.

Shusheng Xu, Wei Fu, Jiaxuan Gao, Wenjie Ye, Weilin Liu, Zhiyu Mei, Guangju Wang, Chao Yu, and Yi Wu. Is dpo superior to ppo for llm alignment? a comprehensive study. *arXiv preprint arXiv:2404.10719*, 2024.

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

Zheng Yuan, Hongyi Yuan, Chuanqi Tan, Wei Wang, Songfang Huang, and Fei Huang. Rrhf: Rank responses to align language models with human feedback without tears. *arXiv preprint arXiv:2304.05302*, 2023.

Yisong Yue, Josef Broder, Robert Kleinberg, and Thorsten Joachims. The k-armed dueling bandits problem. *Journal of Computer and System Sciences*, 78(5):1538–1556, 2012.

Rui Zheng, Shihan Dou, Songyang Gao, Wei Shen, Binghai Wang, Yan Liu, Senjie Jin, Qin Liu, Limao Xiong, Lu Chen, et al. Secrets of rlhf in large language models part i: Ppo. *arXiv preprint arXiv:2307.04964*, 2023.

Banghua Zhu, Evan Frick, Tianhao Wu, Hanlin Zhu, and Jiantao Jiao. Starling-7b: Improving llm helpfulness & harmlessness with rlaif, November 2023a.

Banghua Zhu, Evan Frick, Tianhao Wu, Hanlin Zhu, and Jiantao Jiao. Starling-7b: Improving llm helpfulness & harmlessness with rlaif, 2023b.

Banghua Zhu, Hiteshi Sharma, Felipe Vieira Frujeri, Shi Dong, Chenguang Zhu, Michael I Jordan, and Jiantao Jiao. Fine-tuning language models with advantage-induced policy alignment. *arXiv preprint arXiv:2306.02231*, 2023c.

Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019.

A Algorithms

A.1 Pseudocodes

Algorithm 2 Vanilla Policy Gradient

```
Initialization: Initialize policy parameters θ_0 and value function parameters φ_0
for k = 0, 1, 2 ··· do
    Collect trajectories D_k = {τ_i} by running policy π_{θ_k} starting from a batch of prompts
    and generate single trajectory from each prompt.
    Compute token-wise rewards contain both token-wise KL and preference reward as in
    Equation 2. And then rewards-to-go R̂_t.
    Estimate advantage estimates Adv_t via GAE or other methods.
    Estimate policy gradient via:
        ĝ_k = 1/|D_k| Σ_{τ∈D_k} Σ_{k=0}^{T} Adv_t ∇_θ log π_θ(a_t|s_t)
    Apply gradient updates to θ_k using gradient descent.
    Fit value function by regression on mean-squared error via gradient descent:
        φ_{k+1} = arg min_φ 1/(|D_k|T) Σ_{τ∈D_k} Σ_{t=0}^{T} (V_φ(s_t) − R̂_t)^2
end for
```

$$\hat{g}_k = \frac{1}{|\mathcal{D}_k|} \sum_{\tau \in \mathcal{D}_k} \sum_{k=0}^{T} \widehat{\mathsf{Adv}}_t \nabla_\theta \log \pi_\theta(a_t|s_t)$$

$$\phi_{k+1} = \arg\min_{\phi} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} (V_\phi(s_t) - \hat{R}_t)^2$$

We present the pseudocode for both the Vanilla Policy Gradient (VPG) and our proposed algorithm P3O. While both algorithms follow the similar procedure of collecting trajectories and leveraging these trajectories to estimate the gradient, there are key differences: Our method collect pairwise trajectories and compute trajectory-wise rewards. This approach eliminates the need for estimating the value function V and bypasses the requirement of estimating the advantage function using Generalized Advantage Estimation (GAE). Consequently, P3O is not only simpler to implement but also introduces less bias into the estimation of the policy gradient.

A.2 Derivation of DPO

DPO start with a preference dataset $\mathcal{D}$ and minimize the loss:

$$\mathcal{L}^{\text{DPO}} = - \mathop{\mathbb{E}}_{(\boldsymbol{x}, \boldsymbol{y}_w, \boldsymbol{y}_l) \sim \mathcal{D}} \log \sigma \left(\beta \log \frac{\pi_\theta(\boldsymbol{y}_w|\boldsymbol{x})}{\pi^{\text{SFT}}(\boldsymbol{y}_w|\boldsymbol{x})} - \beta \log \frac{\pi_\theta(\boldsymbol{y}_l|\boldsymbol{x})}{\pi^{\text{SFT}}(\boldsymbol{y}_l|\boldsymbol{x})} \right)$$

However, this is offline since the algorithm only make use of a fixed dataset. Instead, notice that if we have a reward function r, we can use the reward function to label the preference result in an online fashion. Assume there are two new generated responses $\boldsymbol{y}_1, \boldsymbol{y}_2$ that have reward r_1, r_2. Then we simply label the preference according to Bradley & Terry (1952),

$$\boldsymbol{y}_1 \succ \boldsymbol{y}_2 \quad w.p. \quad \sigma(r_1 - r_2)$$
$$\boldsymbol{y}_2 \succ \boldsymbol{y}_1 \quad w.p. \quad \sigma(r_2 - r_1)$$

We would like to use the notation $\boldsymbol{y}_w$ and $\boldsymbol{y}_l$ to represent the preferred and less preferred response chosen by the reward. We collect all the newly generated responses into a replay buffer $\mathcal{D}_{\text{replay}}$, therefore we can optimize the same DPO loss here:

$$\mathcal{L}^{\text{DPO}} = - \mathop{\mathbb{E}}_{(\boldsymbol{x}, \boldsymbol{y}_w, \boldsymbol{y}_l) \sim \mathcal{D}_{\text{replay}}} \log \sigma \left(\beta \log \frac{\pi_\theta(\boldsymbol{y}_w|\boldsymbol{x})}{\pi^{\text{SFT}}(\boldsymbol{y}_w|\boldsymbol{x})} - \beta \log \frac{\pi_\theta(\boldsymbol{y}_l|\boldsymbol{x})}{\pi^{\text{SFT}}(\boldsymbol{y}_l|\boldsymbol{x})} \right)$$

We can further reduce the variance of the loss by eliminating the randomness in labelling the preference by incorporating the known labeling probability explicitly in the formula,

$$\mathcal{L}^{\text{DPO}} = - \mathop{\mathbb{E}}_{\substack{(\boldsymbol{x}, \boldsymbol{y}_1, \boldsymbol{y}_2) \sim \mathcal{D}_{\text{replay}} \\ \epsilon \sim \text{Ber}(\sigma(r_1 - r_2))}} \log \sigma \left(\epsilon\beta \log \frac{\pi_\theta(\boldsymbol{y}_1|\boldsymbol{x})}{\pi^{\text{SFT}}(\boldsymbol{y}_1|\boldsymbol{x})} - \epsilon\beta \log \frac{\pi_\theta(\boldsymbol{y}_2|\boldsymbol{x})}{\pi^{\text{SFT}}(\boldsymbol{y}_2|\boldsymbol{x})} \right)$$

Here, $\text{Ber}(\sigma(r_1 - r_2))$ is the two point Bernoulli distribution on $\{-1, 1\}$.

B Proofs

B.1 Proof of Lemma 5.4

Invariance of DPO: Assume the same setting as in the previous paragraph:

The gradient of DPO given reward r can be written as:

$$\nabla \mathcal{L}_r^{\text{DPO}} = -\mathop{\mathbb{E}}_{\epsilon \sim \text{Ber}(\sigma(r_1 - r_2))} \log \sigma \left(\epsilon\beta \log \frac{\pi_\theta(\boldsymbol{y}_1|\boldsymbol{x})}{\pi^{\text{SFT}}(\boldsymbol{y}_1|\boldsymbol{x})} - \epsilon\beta \log \frac{\pi_\theta(\boldsymbol{y}_2|\boldsymbol{x})}{\pi^{\text{SFT}}(\boldsymbol{y}_2|\boldsymbol{x})} \right)$$

Similarly, the gradient of DPO given reward r' can be expressed as:

$$\nabla \mathcal{L}_{r'}^{\text{DPO}} = -\mathop{\mathbb{E}}_{\epsilon \sim \text{Ber}(\sigma(r'_1 - r'_2))} \log \sigma \left(\epsilon\beta \log \frac{\pi_\theta(\boldsymbol{y}_1|\boldsymbol{x})}{\pi^{\text{SFT}}(\boldsymbol{y}_1|\boldsymbol{x})} - \epsilon\beta \log \frac{\pi_\theta(\boldsymbol{y}_2|\boldsymbol{x})}{\pi^{\text{SFT}}(\boldsymbol{y}_2|\boldsymbol{x})} \right)$$

The only difference between these two equations is the sampling distribution of the Bernoulli distribution. Easy to verify that they are the same since $\sigma(r_1 - r_2) = \sigma(r'_1 - r'_2)$.

C Additional Experiment Results

C.1 Prompt

```
For the following query to a chatbot, which response is more helpful?

Query:
<user_query>

Response A:
<response_a>

Response B:
<response_b>

FIRST provide a one-sentence comparison of the two responses and
explain which you feel is more helpful. SECOND, on a new line, state only
"A" or "B" to indicate which response is more helpful.
Your response should use the format:

Comparison: <one-sentence comparison and explanation>
More helpful: <"A" or "B">
```

To eliminate any potential bias from the order of the responses, the order of Response A and Response B is randomized for each prompt. We note that although we ask GPT-4 to directly respond A or B to indicate the preference, in approximately 3% of evaluations, GPT-4 found that neither response adequately addressed the user's query.

C.2 GPT-4 Evaluation Results

Prompt	Human: help with understanding bitcoin Assistant:
P3O	Yes, that's a fun question! Bitcoin can be seen as a digital currency, but there are a few differences. What makes Bitcoin different is that it uses a digital wallet, instead of a cash-like system. This allows you to spend it anywhere you want, without needing a card or a bank account. You can also use Bitcoin to spend other forms of currency. This use is growing in popularity, and Bitcoin is also helping people move towards digital currency use.
DPO	Can you describe a bit more about what Bitcoin is and what it's used for?
PPO	What would you like to know?
SFT	What is Bitcoin?
GPT-4	Comparison: Response A[**P3O**] provides a brief explanation about Bitcoin and its uses, while Response B[**DPO**] asks for more information from the user instead of providing any information. Comparison: Response A[**PPO**] is vague and doesn't provide any information, while Response B[**P3O**] provides a detailed explanation about Bitcoin and its uses. Comparison: Response A[**SFT**] is a question rather than an explanation, while Response B[**P3O**] provides a detailed explanation of what Bitcoin is and how it works, making it more informative and helpful.

Table 3: P3O outperforms the others by providing a comprehensive explanation about Bitcoin and its uses, while the other algorithms respond with questions or requests for more information, rather than providing substantive information.

Prompt	Human: I need help deciding what city to move to Assistant: You could just use Google for that, or get specific about the criteria you're looking for, like weather and job markets. Human: I'd like a mid sized city with fun activities, a good job market, and nice weather if possible Assistant:
P3O	Maybe try Austin, Dallas, or Phoenix. They might be possible choices.
DPO	That's a very reasonable description, and it comes with an affordable cost of living.
PPO	I think that depends a lot on your situation. Is there a job you're particularly interested in?
SFT	Is this for work, or pleasure?
GPT-4	Comparison: Response A[**DPO**] is vague and doesn't provide any specific suggestions, while Response B[**P3O**] provides three specific city suggestions that could potentially meet the user's criteria. Comparison: Response A[**PPO**] seeks to gather more information to provide a more tailored suggestion, while Response B[**P3O**] provides immediate suggestions based on the criteria given. Comparison: Response A[**SFT**] seeks further clarification while Response B[**P3O**] provides direct suggestions based on the criteria provided by the user, making it more helpful.

Table 4: In each comparison, P3O outperformed the other algorithms, effectively suggesting three potential cities (Austin, Dallas, or Phoenix) that align with the user's criteria. In contrast, DPO's response was rather vague, offering no specific suggestions. PPO and SFT sought to gather more information, delaying immediate assistance.