Fine-Tuning Language Models with Reward Learning on Policy

Hao Lang   **Fei Huang**   **Yongbin Li** *
Alibaba Group
{hao.lang, f.huang, shuide.lyb}@alibaba-inc.com

Abstract

Reinforcement learning from human feedback (RLHF) has emerged as an effective approach to aligning large language models (LLMs) to human preferences. RLHF contains three steps, i.e., human preference collecting, reward learning, and policy optimization, which are usually performed serially. Despite its popularity, however, (fixed) reward models may suffer from inaccurate off-distribution, since policy optimization continuously shifts LLMs' data distribution. Repeatedly collecting new preference data from the latest LLMs may alleviate this issue, which unfortunately makes the resulting system more complicated and difficult to optimize. In this paper, we propose reward learning on policy (RLP), an unsupervised framework that refines a reward model using policy samples to keep it on-distribution. Specifically, an unsupervised multi-view learning method is introduced to learn robust representations of policy samples. Meanwhile, a synthetic preference generation approach is developed to simulate high-quality preference data with policy outputs. Extensive experiments on three benchmark datasets show that RLP consistently outperforms the state-of-the-art. Our code is available at https://github.com/AlibabaResearch/DAMO-ConvAI/tree/main/rlp.

1 Introduction

Large language models (LLMs) (Brown et al., 2020; Bommasani et al., 2021) have shown great promise in following open-ended user instructions (Askell et al., 2021; Ouyang et al., 2022; Longpre et al., 2023). These capabilities are largely attributed to the fine-tuning of pretrained LLMs using Reinforcement Learning from Human Feedback (RLHF) (Christiano et al., 2017; Bai et al., 2022a), which is a prominent technique to align LLMs with human preferences and greatly en-
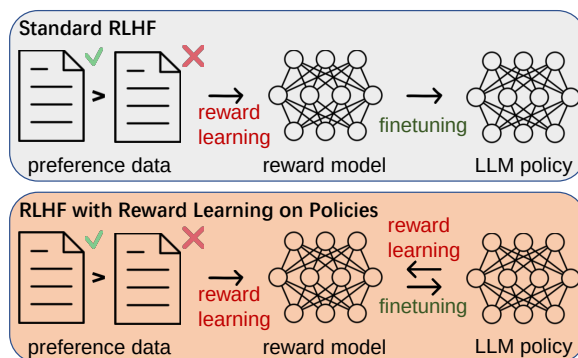

Figure 1: Comparison of standard RLHF (top) and RLHF with reward learning on policies (bottom). Different from (top), which performs reward learning and policy optimization serially, we iteratively train one of the two models with the help of the other.

hance their usability and safety (OpenAI, 2023; Anthropic, 2023; Google, 2023).

A typical RLHF procedure is comprised of three interrelated steps: human preference collecting, reward learning, and policy optimization (Figure 1 top). The reward learning step fits a reward model to the preference data that elicits evaluations from humans. The policy optimization step uses reinforcement learning (RL) to fine-tune a language model to produce outputs assigned high reward.

In practice, the three key steps of RLHF are often performed serially (Casper et al., 2023). Since policy optimization shifts the language model's data distribution during the RL phase, the (fixed) reward model will be inaccurate off-distribution which is trained on offline data (Touvron et al., 2023b). Hence, reward model accuracy can quickly degrade and in turn degenerate the policy that exploits differences between the inferred and true reward (Gao et al., 2023).

The above issue can be mitigated by gathering new human preference data from an up-to-date version of policy (Ziegler et al., 2019). However, the resulting system is significantly more complicated

---

* Corresponding author.

and difficult to optimize, involving iterations of data gathering, reward learning, and RL fine-tuning. Moreover, significant work is required to maintain high data quality over a long time in this setting.

In this paper, we show how to optimize a reward model against the policy to keep it on-distribution, without repeatedly collecting new human preference data. We propose *Reward Learning on Policy (RLP)*, a framework that refines a reward model using policy samples in an unsupervised manner. RLP first trains a reward model and a language model policy from scratch with standard RLHF methods, and then retrains the reward model when exposed to the sample distribution of the trained policy. Finally, RLP retrains the policy on the retrained reward model, which attempts to maintain an accurate reward for the latest policy.

Concretely, RLP uses policy samples to retrain the reward model via two methods: unsupervised multi-view learning (UML) and synthetic preference generation (SPG). RLP-UML constructs two views for an input by generating two responses from the policy (Zhao et al., 2017), then optimizes a multi-view information bottleneck loss (Federici et al., 2020) when fitting the reward model to a dataset of human preferences. This training objective follows the information bottleneck principle (Tishby et al., 2000) and helps learn robust representations of the policy's data distribution.

In addition, RLP-SPG simulates preferences on policy generations to supplement the human preference data. Rather than producing and scoring two outputs with LLMs as in Reinforcement Learning from AI Feedback (RLAIF) (Bai et al., 2022b; Lee et al., 2023), RLP-SPG generates a set of outputs for an instruction. In this way, RLP-SPG can quantify uncertainty and decide when to trust model predictions via measuring the size of the largest semantic equivalence cluster (Kuhn et al., 2023; Si et al., 2023). Thus, RLP-SPG selectively generates pairwise preferences for instructions with low uncertainty (Lin et al., 2023), where the preferred output is sampled from the largest cluster of the output set and the non-preferred one is sampled from the rest clusters. This sampling scheme also conforms to the self-consistency assumption, i.e., the most consistent output is selected as the final prediction (Wang et al., 2023; Chen et al., 2023).

Our main contributions are as follows:

- We propose Reward Learning on Policy (RLP), an unsupervised framework that re-

fines a reward model using policy samples to keep it on-distribution for RLHF.

- We optimize a multi-view loss when retraining the reward model to learn representations of the policy's data distribution. We also simulate preferences with a set of policy outputs, which enables selective generation and high-quality data construction.

- Our experiments on three standard benchmark datasets show that RLP outperforms existing methods for learning from human feedback, including PPO-based RLHF.

2 Related Work

Instruction tuning is a procedure to fine-tune pretrained LLMs with instructions and human-written completions (Mishra et al., 2022; Sanh et al., 2022), which increases the usability of LLMs (Chung et al., 2022). Recently, **RLHF** has emerged as the central method for fine-tuning LLMs based on human preferences and further improves their downstream task performance and alignment with user intent (Christiano et al., 2017). Generally, RLHF methods first fit a reward model to human preferences, then fine-tune a language model to maximize the inferred reward using RL algorithms.

Reward models tend to be an imperfect estimate of the true reward due to misspecification (Bıyık et al., 2022) and misgeneralization (Tien et al., 2023), and imperfect in reward models leads to reward hacking (Skalse et al., 2022). Methods with reward ensemble (Coste et al., 2024) and diverse feedback (Yu et al., 2023) are proposed to tackle this issue. Our method retrains the reward model with policy samples to make it on-distribution and generalize to the policy's data distribution.

Human feedback simulation aims to generate additional synthetic preference data using weak human supervision and LLMs (Bai et al., 2022b). RLAIF approaches obtain pairwise preferences by scoring two outputs from a shared prompt (Lee et al., 2023), whereas RLCD generates outputs from two variants of a prompt (Yang et al., 2024). Our method RLP-SPG is the first attempt to simulate human preferences using a set of outputs.

Uncertainty quantification provides confidence scores for generations of LLMs, helping users decide when to trust these generation results (Si et al., 2023). Supervised methods fine-tune

the language model to predict the uncertainty (Kadavath et al., 2022; Lin et al., 2022), while unsupervised methods measure uncertainty by calculating semantic entropy or semantic dispersion amongst generated answers (Kuhn et al., 2023; Lin et al., 2023). In this work, we measure uncertainty to selectively generate preference data.

3 Preliminaries

We start by introducing the instruction following task (Ouyang et al., 2022; Bai et al., 2022a). Given user instructions $x \in \mathcal{X}$ (e.g., "Generate a definition for artificial intelligence"), we aim to develop a model π_θ that generates high-quality responses $y \sim \pi_\theta(y|x)$ as judged by some latent reward model. In this study, we focus on RLHF for this task, due to its central role in instruction-following LLMs (Ouyang et al., 2022). RLHF usually consists of three steps: human preference collecting, reward modeling, and RL policy optimization (Dubois et al., 2024; Rafailov et al., 2024; Casper et al., 2023).

Step 0, SFT: RLHF generally begins with a pre-trained model, which is fine-tuned with supervised learning on instruction-following demonstrations (x, y), to produce a model $\pi^{\mathrm{SFT}}(y|x)$.

Step 1, Human preference collecting: The first step is to produce pairs of responses $(y_1, y_2) \sim \pi^{\mathrm{SFT}}(y|x)$ for the instruction x. These are then presented to humans who express preferences for each response, denoted as $y_w \succ y_l \mid x$ where y_w and y_l denotes the preferred and non-preferred completion amongst (y_1, y_2) respectively.

Step 2, Reward learning: The second step is to fit a reward model $r_\phi(x, y)$ by minimizing the negative log-likelihood loss (Christiano et al., 2017):

$$\mathcal{L}_R = -\mathbb{E}_{(x, y_w, y_l) \sim \mathcal{D}}[\log \sigma(r_\phi(x, y_w) - r_\phi(x, y_l))],$$

where $\mathcal{D} = \{(x, y_w, y_l)\}$ is a dataset of pairwise preferences and σ is the sigmoid function. $r_\phi(x, y)$ is often initialized from $\pi^{\mathrm{SFT}}(y|x)$ with one additional linear layer that infers the reward value.

Step 3, RL policy optimization: The third step is to use the reward model $r_\phi(x, y)$ to fine-tune the language model. The parameters θ of π are trained to maximize

$$\mathbb{E}_{x \sim \mathcal{U}, y \sim \pi_\theta(y|x)}[r_\phi(x, y) - \beta \mathbb{D}_{\mathrm{KL}}(\pi_\theta(y|x)||\pi_{\mathrm{ref}}(y|x))],$$

where $\mathcal{U} = \{x\}$ is an unlabeled instruction dataset, the language model policy $\pi_\theta(y|x)$ is fine-tuned

from the SFT model π^{SFT}, the reference policy π_{ref} is also the SFT model π^{SFT}, and β is a regularization coefficient controlling the deviation from π_{ref}. This objective is typically optimized with RL algorithms such as PPO (Schulman et al., 2017).

4 Reward Learning on Policy

4.1 Overview

In this study, we propose a novel RLHF framework to fine-tune LLMs with human feedback following five steps: **Step 1-3.** Collect a pairwise human preference dataset $\mathcal{D}$, then train a reward model r_ϕ and fine-tune a language model policy π_θ; **Step 4.** Retrain a reward model $\hat{r}_\phi$ using outputs of policy π_θ; **Step 5.** Retrain a policy $\hat{\pi}_\theta$ based on the retrained reward model $\hat{r}_\phi$.

Before applying RLP, we assume existing RLHF approaches can be used to train the reward model r_ϕ and the policy π_θ (Ouyang et al., 2022). The sample distribution of the policy π_θ can be quite different from the preference data $\mathcal{D}$ on which the reward model r_ϕ is trained (Touvron et al., 2023b). For example, outputs become increasingly longer after applying RLHF methods as shown in the analysis of AlpacaFarm (Dubois et al., 2024). The average length of SFT outputs is 278 characters and applying PPO increases it to 637 tokens. These distributional differences make the reward model r_ϕ inaccurate off-distribution.

Our goal is to refine the reward model using samples of the policy π_θ and keep it on-distribution. This process is expected to increase the generalization of the retrained reward model $\hat{r}_\phi$ to policy samples. Accordingly, it can maintain an accurate reward during the RL policy optimization phase.

4.2 Reward Retraining

We now describe how to retrain the reward model $\hat{r}_\phi$ in **Step 4** of RLP. We first construct a dataset of policy samples $\mathcal{P} = \{(x, \boldsymbol{y}) \mid x \in \mathcal{U}, \boldsymbol{y} \sim \pi_\theta(y|x)\}$, where $\boldsymbol{y}$ is a set of n outputs from policy π_θ for instruction x. Then, we refine the reward model with policy samples $\mathcal{P}$ in addition to the human preference dataset $\mathcal{D}$. Specifically, we propose two different methods for this purpose: unsupervised multi-view learning (UML) and synthetic preference generation (SPG).

Unsupervised Multi-View Learning attempts to learn robust representations of policy samples. For each pair $(x, \boldsymbol{y}) \in \mathcal{P}$, two semantic invariant

views are constructed: $v_i(x) = (x, y) \mid y \sim \boldsymbol{y}$, $(i = 1, 2)$. These two views preserve the same task-relevant information (Zhao et al., 2017). Then, a multi-view information bottleneck (MIB) loss (Federici et al., 2020) is optimised for unsupervised representation learning, following the information bottleneck principle (Tishby et al., 2000). This optimization process retains task-relevant information in the representations while discarding superficial information.

To facilitate the computation, we parametrize the representation z_i of each view $v_i(x)$ with a factorized Gaussian distribution, i.e., $p_\psi(z|v_i) = \mathcal{N}[\mu(v_i), \Sigma(v_i)]$. Concretely, we estimate $v_i(x)$ with the final transformer layer of the reward model and use two neural networks $\mu(v_i)$ and $\Sigma(v_i)$ to produce the mean and deviation respectively. The following MIB loss is optimized:

$$\mathcal{L}_M = \mathbb{E}_{(x,\boldsymbol{y})\sim\mathcal{P}}[-\mathbb{I}(z_1; z_2) + \mathbb{D}_{\text{SKL}}(p_\psi(z|v_1)||p_\psi(z|v_2))],$$

where $\mathbb{I}$ calculates mutual information of two random variables, and $\mathbb{D}_{\text{SKL}}$ represents the symmetrized KL divergence obtained by averaging the expected value of $\mathbb{D}_{\text{KL}}(p_\psi(z|v_1)||p_\psi(z|v_2))$ and $\mathbb{D}_{\text{KL}}(p_\psi(z|v_2)||p_\psi(z|v_1))$.

Synthetic Preference Generation aims to simulate high-quality preference data with policy samples. For each pair $(x, \boldsymbol{y}) \in \mathcal{P}$, we assume the most frequent item of $\boldsymbol{y}$ as the correct prediction and its frequency as the confidence score (Si et al., 2023), following the self-consistency assumption (Wang et al., 2023). To address semantic equivalence, i.e., different sentences can mean the same thing, we cluster items of $\boldsymbol{y}$ into groups $\mathcal{G}$ with a bi-directional entailment algorithm (Kuhn et al., 2023). Sentences from each group $\boldsymbol{g} \in \mathcal{G}$ are expected to share the same meaning. We estimate the confidence score of $(x, \boldsymbol{y})$ as $\frac{|\tilde{\boldsymbol{g}}|}{|\boldsymbol{y}|}$, where $\tilde{\boldsymbol{g}}$ is the largest group of $\mathcal{G}$ and the operator $|\cdot|$ measures the size of a set. Thus, we can selectively generate a synthetic preference dataset with high confidences $\hat{\mathcal{D}} = \{(x, y_w, y_l) \mid (x, \boldsymbol{y}) \in \mathcal{P}, \frac{|\tilde{\boldsymbol{g}}|}{|\boldsymbol{y}|} \geq \gamma, y_w \sim \tilde{\boldsymbol{g}}, y_l \sim \boldsymbol{y} \setminus \tilde{\boldsymbol{g}}\}$, where γ is the threshold for selective generation, the preferred output y_w is sampled from the largest group $\tilde{\boldsymbol{g}}$ with the largest reward score and the non-preferred one y_l is randomly sampled from the rest groups.

The overall loss that we optimize for the reward model $\hat{r}_\phi$ is:

$$\mathcal{L}_{\hat{R}} = -\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}\cup\hat{\mathcal{D}}}[\log \sigma(\hat{r}_\phi(x, y_w) - \hat{r}_\phi(x, y_l))] + \lambda \mathcal{L}_M, \tag{1}$$

where the coefficient λ controls the weight of the multi-view information bottleneck loss. To simplify computational complexity, we implement two variants: 1. **RLP-UML** removes the synthetic dataset $\hat{\mathcal{D}}$ in Eq. 1 and learns the representations of policy samples when fitting the reward model. 2. **RLP-SPG** removes the MIB loss by setting $\lambda = 0$ in Eq. 1 and fits the reward model with human and synthetic preference data.

4.3 Policy Retraining

We finally retrain the policy $\hat{\pi}_\theta$ using $\hat{r}_\phi$ in **Step 5** of RLP. Specifically, we optimize $\hat{\pi}_\theta$ to maximize

$$\mathbb{E}_{x\sim\mathcal{U},y\sim\hat{\pi}_\theta(y|x)}[\hat{r}_\phi(x, y) - \beta\mathbb{D}_{\text{KL}}(\hat{\pi}_\theta(y|x)||\pi_{\text{ref}}(y|x))].$$

Our approach RLP is summarized in Algorithm 1.

Algorithm 1: RLP: RLHF with Reward Learning on Policy

Input: SFT model π^{SFT}, unlabeled data $\mathcal{U}$.
Output: A language model policy $\hat{\pi}_\theta$.

1 Collect a human preference dataset $\mathcal{D}$.
2 Train a reward model r_ϕ using $\mathcal{D}$.
3 Fine-tune a language model π_θ from π^{SFT} using $\mathcal{U}$ and r_ϕ.
4 Retrain a reward model $\hat{r}_\phi$ using $\mathcal{L}_{\hat{R}}$ (Eq. 1).
5 Fine-tune $\hat{\pi}_\theta$ from π^{SFT} using $\mathcal{U}$ and $\hat{r}_\phi$.

	Dataset	#Sample
	SFT dataset	10k
Training data	Preference dataset $\mathcal{D}$	10k
	Unlabeled dataset $\mathcal{U}$	20k
	AlpacaFarm	805
Evaluation data	LLMBar	100
	Vicuna	80

Table 1: Dataset statistics.

5 Experiments

5.1 Datasets

We run experiments on the instruction following task (Ouyang et al., 2022; Bai et al., 2022a), which remains a challenging task for the strongest LLMs today (Wu et al., 2023; Li et al., 2023).

Method	AlpacaFarm		LLMBar	Vicuna
	Simulated Win-Rate	Human Win-Rate	Simulated Win-Rate	Simulated Win-Rate
GPT-4	79.0	69.8	74.0	85.0
ChatGPT	61.4	52.9	59.0	63.7
PPO	46.8	55.1	47.5	57.5
Best-of-n	45.0	50.7	43.4	52.5
SFT	36.7	44.3	42.4	50.0
LLaMA-7B	11.3	6.5	12.5	12.8
RLP-UML (ours)	49.1	56.5	48.5	61.3
RLP-SPG (ours)	**50.2**	**57.4**	**50.5**	**62.5**

Table 2: The win-rate (%) performance of RLP and baselines. Win-rates are computed against reference model `text-davinci-003`. Baseline results in AlpacaFarm come from Dubois et al. (2024). **Bold** numbers are superior results among the implemented LLMs. We omitted LLMBar and Vicuna for human evaluation because the simulated method rankings consistently correlate with the human method rankings in AlpacaFarm.
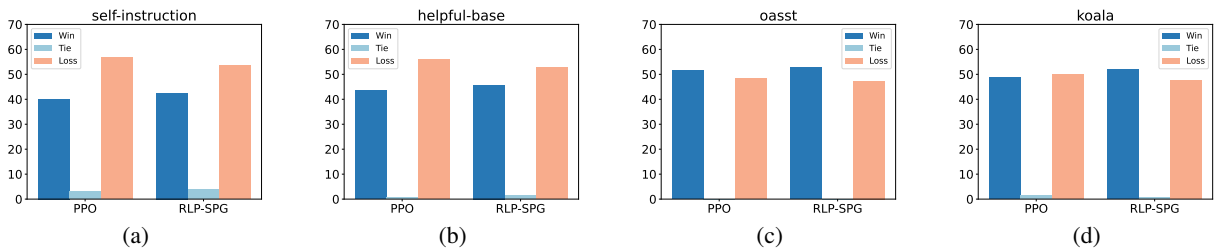

Figure 2: The simulated win-rate (%) performance of RLP-SPG compared to PPO on various subsets of AlpacaFarm. Win-rates are computed against reference model `text-davinci-003`.

Training data of the RLHF procedure come from Alpaca data, which consists of 52k instruction-following demonstrations (x, y) (Taori et al., 2023). Following the data splits of AlpacaFarm (Dubois et al., 2024), three splits are used: 1. SFT split are 10k data for fine-tuning the SFT model π^{SFT}; 2. Preference split are 10k instructions on which we collect pairwise preference dataset $\mathcal{D}$; 3. Unlabeled split are 20k unlabeled instructions $\mathcal{U}$ used in PPO. Concretely, two variants of preference dataset $\mathcal{D}$ are curated: simulated $\mathcal{D}_{\text{sim}}$ are constructed with AlpacaFarm simulated annotators by prompting API LLMs, and human $\mathcal{D}_{\text{human}}$ are constructed with human annotators.

Evaluation data of the trained LLMs include three benchmarks: 1. **AlpacaFarm** consists of 805 instructions, which includes instructions from Self-instruct evaluation (Wang et al., 2022), OASST evaluation, Anthropic helpful evaluation (Bai et al., 2022a), Vicuna evaluation (Chiang and Lee, 2023), and Koala evaluation (Geng et al., 2023); 2. **LLM-Bar** (Zeng et al., 2024) contains 100 instructions with objective preferences. We use the natural set to evaluate performance in real-world distributions; 3. **Vicuna** (Chiang et al., 2023) contains 80 instruc-

tions to evaluate different skills of LLMs. Table 1 shows the statistics of these datasets.

5.2 Metrics and Experimental Setups

Following Dubois et al. (2024); Touvron et al. (2023b), we use win-rate to evaluate the performance of an LLM π_θ, i.e., the percentage of times π_θ is preferred to a reference model π_{ref} for their instruction-following outputs. Following the experimental setups of AlpacaFarm (Dubois et al., 2024), we use simulated win-rate to evaluate methods trained on simulated $\mathcal{D}_{\text{sim}}$ with simulated annotators by prompting API LLMs. In parallel, we use human win-rate to evaluate methods trained on human $\mathcal{D}_{\text{human}}$ with human annotators.

5.3 Implementation Details

Our implementations of reward learning, policy optimization, and simulated annotators are based on the AlpacaFarm codebase and its default hyper-parameters (Dubois et al., 2024). Specifically, we use RL algorithm PPO (Schulman et al., 2017) to fine-tune the language model during policy optimization. All reward models and language models are based on LLaMA-7B (Touvron et al., 2023a). We sample $n = 10$ outputs for each in-

Instruction:	
What are different drawers I should have for clothes?	

Instruction:
What are different drawers I should have for clothes?

SFT Output:
Different drawers for clothes can include a top drawer for socks and underwear, a middle drawer for shorts, pants, and skirts, a lower drawer for long-sleeved shirts, a separate drawer for t-shirts and tank tops, a drawer for sweaters and cardigans, a drawer for jackets, and a drawer for shoes.

PPO Output:
Different drawers for clothes can include a master drawer for all items, such as pants, shirts, and jackets; a drawer for socks and underwear; a drawer for delicates, such as lingerie and fine clothing; a drawer for seasonal clothing, such as winter coats and swimwear; a drawer for work clothes, such as dress shirts and skirts; a drawer for shoes; and a drawer for accessories, such as belts, hats, and purses.

RLP-UML Output:
Different drawers for clothes can include:
- A top drawer for long-sleeved shirts and blouses.
- A middle drawer for short-sleeved shirts and blouses.
- A bottom drawer for sweaters, sweaters, and jackets.
- A separate drawer for pants, shorts, and skirts.
- A drawer for socks and underwear.
- A drawer for accessories such as belts, scarves, and hats.
- A separate drawer for seasonal clothing, such as winter coats and summer dresses.

RLP-SPG Output:
Different drawers for clothes can include:
- A master drawer for folded clothing, such as shirts, pants, skirts, and shorts.
- A drawer for seasonal items, like sweaters and jackets.
- A drawer for delicate items, like lingerie and fine fabrics.
- A drawer for formal wear, like suits and dresses.
- A drawer for accessories, like belts, scarves, and jewelry.
- A drawer for uniforms, like work clothes and school uniforms.
- A drawer for out-of-season items, like swimwear and winter coats.
- A drawer for shoes, organized by type and use.
- A drawer for cleaning supplies, like a laundry bin or ironing board.
- A drawer for miscellaneous items, like first-aid kits and sewing supplies.

Table 3: Outputs generated by RLP and baselines for an example from AlpacaFarm. RLP-UML and RLP-SPG produce more comprehensive outputs than SFT and PPO in this case.

struction to construct a dataset of policy samples $\mathcal{P}$. For unsupervised multi-view learning, we implement $\mu(v_i)$ and $\Sigma(v_i)$ as three-layer MLPs, and use Jensen-Shannon mutual information estimator (Hjelm et al., 2018) to estimate mutual information $\mathbb{I}$ in the MIB loss. For synthetic preference generation, we implement a bidirectional entailment clustering algorithm using Deberta-large model (He et al., 2021) and set the threshold $\gamma = 0.5$ for selective generation. We set $\lambda = 0.5$ in Eq. 1 for RLP-UML. At training and inference time, we use a sampling temperature of 1.0 and 0.7, respectively. All experiments are performed on a single $8 \times A100$ machine.

5.4 Baselines

We compare RLP with competitive baselines: **1. LLaMA-7B** (Touvron et al., 2023a) directly generates outputs using the base unaligned LLaMA-7B; **2. SFT** (Taori et al., 2023) is a LLaMA-7B model supervised fine-tuned on 10k Alpaca instruction-

following data; **3. Best-of-**n (Stiennon et al., 2020) samples n i.i.d. responses from the SFT model and returns the response with the highest inferred reward; **4. PPO** (Schulman et al., 2017) is a reinforcement learning algorithm that maximizes surrogate reward, subject to a KL penalty keeping parameters near the SFT model; **5. ChatGPT** uses OpenAI API LLM `gpt-3.5-turbo-0301`; **6. GPT-4** uses OpenAI API LLM `gpt-4-0314`.

5.5 Main Results

We compare the win-rate performance of our method RLP and all baselines on three standard benchmarks to assess their instruction-following ability in Table 2. It can be seen that API LLM GPT-4 significantly outperforms all other models due to its obvious advantages. Among the implemented LLMs, RLP-SPG performs the best in both the simulator and human preference data, and achieves SOTA results on all three benchmarks. Compared with the implemented best-performing

baseline PPO, RLP-SPG brings up from a simulator win-rate of 46.8% to 50.2% in AlpacaFarm, 47.5% to 50.5% in LLMBar, and 57.5% to 62.5% in Vicuna. RLP-SPG also brings up from a human win-rate of 55.1% to 57.4% in AlpacaFarm.

We can also observe that: **1.** Both the two variants of RLP, namely, RLP-UML and RLP-SPG, outperform all implemented baselines that do not train reward models using policy samples. The performance gain demonstrates the advantage of considering policy for reward learning, which can help keep the reward model on-distribution. **2.** RLP-SPG generally outperforms RLP-UML under all circumstances. It demonstrates that synthetic preference generation leads to better performance, which simulates pairwise preference data with policy samples that can be leveraged for optimizing a reward model directly.

To provide a clearer perspective on RLP's superiority over other baselines, we illustrate the simulated win-rate of our best method RLP-SPG compared to the best-performing baseline PPO on various subsets of AlpacaFarm in Figure 2. Instructions from these subsets show diverse coverage over realistic interactions, allowing for an intricate analysis of the proficiency attained through language model fine-tuning (Dubois et al., 2024). Notably, RLP-SPG outperforms PPO across all subsets, including Self-instruct evaluation, Anthropic helpful evaluation, OASST evaluation, and Koala evaluation. It further indicates that reward learning on policy leads to a comprehensive enhancement in the capabilities of the LLMs. Meanwhile, RLP also outperforms these baselines on knowledge intensive benchmarks such as MMLU (Hendrycks et al., 2021) (See Appendix A.).

The difference between RLP and baselines can be observed qualitatively as well. For example, the case shown in Table 3 makes it sufficiently clear why RLP is so strongly preferred over our baselines from AlpacaFarm. Compared to RLP-UML, RLP-SPG generates even longer and more comprehensive outputs.

5.6 Ablation Studies

This section provides comprehensive ablation studies to understand the efficacy of RLP. For consistency, all ablations are conducted using metric simulated win-rate that is computed against reference model `text-davinci-003`.

Method	AlpacaFarm Win-Rate	LLMBar Win-Rate
InfoMax	44.4	46.5
MVI	48.1	47.5
CL	48.2	46.5
RLP-UML	**49.1**	**48.5**

Table 4: Ablation study on the representation learning loss for RLP-UML.

Sampling	Method	Preference Acc.	AlpacaFarm Win-Rate
A Pair	RLAIF	52	46.0
	Reward	56	48.9
A Set	Select-All	55	48.7
	RLP-SPG	**59**	**50.2**

Table 5: Ablation on different synthetic preference generation approaches.

Information Bottleneck Loss We demonstrate the effectiveness of our multi-view information bottleneck loss $\mathcal{L}_M$ by replacing $\mathcal{L}_M$ in Eq. 1 with other alternatives of representation learning: **1. InfoMax** (Poole et al., 2019) maximizes the mutual information between an input $v(x)$ and its representation z, i.e., $\mathbb{I}(v; z)$; **2. MVI** (Bachman et al., 2019) is similar to InfoMax except that it maximizes the mutual information between its two views $\mathbb{I}(z_1; z_2)$; Note that neither InfoMax nor MVI attempts to remove superficial information from representations. **3. CL** (Caron et al., 2020) uses a contrastive learning loss. Positive pairs in this variant are obtained using our multi-view construction approach.

Results in Table 4 show that the information bottleneck loss used in RLP-UML performs better than all other variants. We also want to highlight that the approach of explicitly removing superficial information in RLP-UML makes it outperform InfoMax and MVI by 4.7% and 1.0% in AlpacaFarm, and 2.0% and 1.0% in LLMBar, respectively. This validates our claim that the multi-view information bottleneck loss used in RLP-UML effectively removes the superficial information.

Synthetic Preference Generation We compare RLP with two types of synthetic preference generation approaches: **I.** sampling a pair of responses for each instruction and then labeling its preference with LLMs; **II.** sampling a set of outputs for each instruction and then selecting a preferred and a non-preferred one to construct pairwise preference data.

	Instruction:

Instruction:
Generate a definition for the term "artificial intelligence".

RLAIF Simulated Preference Data:
y_w: Artificial intelligence (AI) is a branch of computer science that enables machines to learn from experience, adjust to new input, and mimic human decision-making processes. AI technology is used to develop systems that can interact with their environment and achieve tasks such as robotic navigation, natural language processing, and computer vision.
y_l: Artificial intelligence (AI) is a field of study focused on developing computers and machines that are capable of performing tasks that are typically done by humans, such as reasoning, problem solving, and decision-making. AI technology is used in a variety of applications, including robotics, games, speech recognition, natural language processing, and automation.

RLP Simulated Preference Data:
y_w: Artificial intelligence (AI) is a branch of computer science that involves the use of sophisticated algorithms and technology to enable machines to think, learn, and act like humans. AI systems can be programmed to perform a variety of tasks, from simple to complex, such as recognizing speech, understanding natural language, automating tasks, and more. AI technology is used in a wide range of applications, from self-driving cars to virtual assistants to automated customer service. In short, AI is the science and technology of making computers and machines smart and capable of doing things traditionally done by humans.
y_l: Artificial intelligence (AI) is a field of study focused on developing computers and machines that can think and act like humans. AI technology allows machines to interact with their environment and to learn from their mistakes, just like humans do.

Table 6: An example instruction and corresponding simulated preference data from RLAIF and RLP-SPG.

Decision	Instruction	Confidence
Reject	Describe the life and reign of King Charles II.	0.1
	What type of fruit would be a good addition to a fruit salad?	0.2
	Research about a famous scientist and provide a short biography about him/her.	0.2
	Compose a five word sentence describing your day.	0.4
Accept	Write a scientific explanation for why the sky is blue.	0.6
	Find the synonyms of the following word: 'Tenacious'.	0.7
	Find the main idea of the following passage.	0.8
	Create a tweet summarizing the following news article in 140 characters or less.	0.9

Table 7: Cases of rejected and accepted instructions for selective synthetic preference generation by RLP-SPG.

For type **I**, we implement **RLAIF** (Lee et al., 2023) that labels preferences with policy π_θ, and **Reward** that rank the two outputs with reward model r_ϕ and assume the top ranked output as the preferred one. For type **II**, we study a variant of RLP, **Select-All**, that sets $\gamma = 0$ for selective generation and no longer rejects low confidence data.

Table 5 shows the accuracy of generated preference data and the win-rate of the corresponding LLMs. Golden preferences are labeled with AlpacaFarm simulated annotator. These results indicate that RLP-SPG outperforms all ablation variants in terms of synthetic preference quality and LLM performance. We can also observe that: 1. Sampling a set of outputs rather than a pair for each instruction helps encourage output diversity and leads to high-quality preference generation. 2. The confidence score based on multiple sampling can be used for selective generation and further improve preference quality. 3. LMMs trained with

more accurate preference data generally perform better and obtain higher win-rate scores.

5.7 Further Analysis

Here we present further analysis of intermediate results during LLM training. Table 6 shows an example of simulated preference data by RLAIF and RLP-SPG, respectively. The two outputs (y_w and y_l) of RLAIF for this case look quite similar. However, y_w is preferred by RLAIF which would bring in noises in the training process. On the contrary, y_w of RLP-SPG is more comprehensive than y_l of RLP-SPG, resulting in more accurate labels. We also find a major difference in the length distributions of RLP-SPG outputs, with preferred outputs y_w (510 characters on average) significantly longer than non-preferred outputs y_l (449 characters on average).

Table 7 also demonstrates cases of rejected and accepted instructions for selective synthetic preference generation by RLP-SPG (rejecting low confi-

dence generations). It can be observed that open-ended instructions (e.g., more subjective and creative) tend to have low confidences and be rejected.

6 Conclusion

In this paper, we propose reward learning on policy (RLP), a novel framework to align LLMs with human preferences. RLP learns robust representations of the policy's data distribution via optimizing a multi-view information bottleneck loss. RLP also simulates preferences with a set of policy outputs, which enables confidence estimation and selective generation. Extensive experiments demonstrate that RLP outperforms SOTA baselines.

Limitations

While we have carefully studied the effectiveness of RLP compared to several baselines on three benchmark datasets for LLaMA-7B, we have not yet empirically verified our conclusions when aligning larger pretrained LLMs. It would also be interesting to align new SOTA pretrained LLMs such as LLaMA 2 (Touvron et al., 2023b) and test other methods for fitting preference data like DPO (Rafailov et al., 2024).

Meanwhile, all of our training and evaluation data are in English, and we have not tested in other languages. Performance may degenerate especially in low-resource languages when pretrained LLMs have not been trained with these data.

Ethics Statement

This work does not raise any direct ethical issues. In the proposed work, we seek to develop a novel RLHF framework to align large language models (LLMs) with human preferences. Concretely, we propose to learn reward models on policy to keep it on-distribution. We believe this work can benefit the field of LLMs, with the potential to benefit other fields requiring NLP models. All experiments are conducted on open datasets.

References

Anthropic. 2023. Introducing claude.

Amanda Askell, Yuntao Bai, Anna Chen, Dawn Drain, Deep Ganguli, Tom Henighan, Andy Jones, Nicholas Joseph, Ben Mann, Nova DasSarma, et al. 2021. A general language assistant as a laboratory for alignment. *arXiv preprint arXiv:2112.00861*.

Philip Bachman, R Devon Hjelm, and William Buchwalter. 2019. Learning representations by maximizing mutual information across views. In *NeurIPS*.

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

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

Erdem Bıyık, Dylan P Losey, Malayandi Palan, Nicholas C Landolfi, Gleb Shevchuk, and Dorsa Sadigh. 2022. Learning reward functions from diverse sources of human feedback: Optimally integrating demonstrations and preferences. *The International Journal of Robotics Research*.

Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. 2021. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. In *NeurIPS*.

Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. 2020. Unsupervised learning of visual features by contrasting cluster assignments. In *NeurIPS*.

Stephen Casper, Xander Davies, Claudia Shi, Thomas Krendl Gilbert, Jérémy Scheurer, Javier Rando, Rachel Freedman, Tomasz Korbak, David Lindner, Pedro Freire, et al. 2023. Open problems and fundamental limitations of reinforcement learning from human feedback. *TMLR*.

Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. 2023. Teaching large language models to self-debug. *arXiv preprint arXiv:2304.05128*.

Yew Ken Chia, Pengfei Hong, Lidong Bing, and Soujanya Poria. 2023. Instructeval: Towards holistic evaluation of instruction-tuned large language models. *arXiv preprint arXiv:2306.04757*.

Cheng-Han Chiang and Hung-yi Lee. 2023. Can large language models be an alternative to human evaluations? In *ACL*.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E Gonzalez, et al.

2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality. *See https://vicuna. lmsys. org (accessed 14 April 2023).*

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep reinforcement learning from human preferences. In *NeurIPS*.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2022. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416.*

Thomas Coste, Usman Anwar, Robert Kirk, and David Krueger. 2024. Reward model ensembles help mitigate overoptimization. In *ICLR*.

Yann Dubois, Xuechen Li, Rohan Taori, Tianyi Zhang, Ishaan Gulrajani, Jimmy Ba, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. 2024. Alpacafarm: A simulation framework for methods that learn from human feedback. In *NeurIPS*.

Marco Federici, Anjan Dutta, Patrick Forré, Nate Kushman, and Zeynep Akata. 2020. Learning robust representations via multi-view information bottleneck. In *ICLR*.

Leo Gao, John Schulman, and Jacob Hilton. 2023. Scaling laws for reward model overoptimization. In *ICML*.

Xinyang Geng, Arnav Gudibande, Hao Liu, Eric Wallace, Pieter Abbeel, Sergey Levine, and Dawn Song. 2023. Koala: A dialogue model for academic research. *Blog post, April*, 1.

Google. 2023. Bard.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. Deberta: Decoding-enhanced bert with disentangled attention. In *ICLR*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. In *ICLR*.

R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. 2018. Learning deep representations by mutual information estimation and maximization. In *ICLR*.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, et al. 2022. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221.*

Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. 2023. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. In *ICLR*.

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

Shiyang Li, Jun Yan, Hai Wang, Zheng Tang, Xiang Ren, Vijay Srinivasan, and Hongxia Jin. 2023. Instruction-following evaluation through verbalizer manipulation. *arXiv preprint arXiv:2307.10558.*

Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. Teaching models to express their uncertainty in words. *arXiv preprint arXiv:2205.14334.*

Zhen Lin, Shubhendu Trivedi, and Jimeng Sun. 2023. Generating with confidence: Uncertainty quantification for black-box large language models. *arXiv preprint arXiv:2305.19187.*

Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V Le, Barret Zoph, Jason Wei, et al. 2023. The flan collection: Designing data and methods for effective instruction tuning. In *ICML*.

Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. 2022. Cross-task generalization via natural language crowdsourcing instructions. In *ACL*.

OpenAI. 2023. Gpt-4 technical report.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. In *NeurIPS*.

Ben Poole, Sherjil Ozair, Aaron Van Den Oord, Alex Alemi, and George Tucker. 2019. On variational bounds of mutual information. In *ICML*.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D Manning, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model. In *NeurIPS*.

Victor Sanh, Albert Webson, Colin Raffel, Stephen H Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, et al. 2022. Multitask prompted training enables zero-shot task generalization. In *ICLR*.

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

Chenglei Si, Zhe Gan, Zhengyuan Yang, Shuohang Wang, Jianfeng Wang, Jordan Boyd-Graber, and Lijuan Wang. 2023. Prompting gpt-3 to be reliable. In *ICLR*.

Joar Skalse, Nikolaus Howe, Dmitrii Krasheninnikov, and David Krueger. 2022. Defining and characterizing reward gaming. In *NeurIPS*.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. 2020. Learning to summarize with human feedback. In *NeurIPS*.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. 2023. Stanford alpaca: An instruction-following llama model.

Jeremy Tien, Jerry Zhi-Yang He, Zackory Erickson, Anca D Dragan, and Daniel S Brown. 2023. Causal confusion and reward misidentification in preference-based reward learning. In *ICLR*.

Naftali Tishby, Fernando C Pereira, and William Bialek. 2000. The information bottleneck method. *arXiv preprint physics/0004057*.

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

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

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *ICLR*.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Anjana Arunkumar, Arjun Ashok, Arut Selvan Dhanasekaran, Atharva Naik, David Stap, et al. 2022. Super-naturalinstructions: Generalization via declarative instructions on 1600+ nlp tasks. *arXiv preprint arXiv:2204.07705*.

Zhaofeng Wu, Linlu Qiu, Alexis Ross, Ekin Akyürek, Boyuan Chen, Bailin Wang, Najoung Kim, Jacob Andreas, and Yoon Kim. 2023. Reasoning or reciting? exploring the capabilities and limitations of language models through counterfactual tasks. *arXiv preprint arXiv:2307.02477*.

Kevin Yang, Dan Klein, Asli Celikyilmaz, Nanyun Peng, and Yuandong Tian. 2024. Rlcd: Reinforcement learning from contrast distillation for language model alignment. In *ICLR*.

Tianshu Yu, Ting-En Lin, Yuchuan Wu, Min Yang, Fei Huang, and Yongbin Li. 2023. Constructive large language models alignment with diverse feedback. *arXiv preprint arXiv:2310.06450*.

Zhiyuan Zeng, Jiatong Yu, Tianyu Gao, Yu Meng, Tanya Goyal, and Danqi Chen. 2024. Evaluating large language models at evaluating instruction following. In *ICLR*.

Jing Zhao, Xijiong Xie, Xin Xu, and Shiliang Sun. 2017. Multi-view learning overview: Recent progress and new challenges. *Information Fusion*.

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

A Evaluation on Knowledge Intensive Benchmark

We also evaluate the performance of RLP on knowledge intensive benchmark MMLU (Hendrycks et al., 2021), which includes exam questions from 57 tasks such as mathematics, history, law, and medicine. Specifically, we use InstructEval (Chia et al., 2023) to perform the evaluation. As shown in Table 8, both RLP-UML and RLP-SPG outperform PPO.

Method	MMLU
GPT-4	86.4
ChatGPT	70.0
PPO	36.9
LLaMA-7B	35.2
RLP-UML (ours)	**37.6**
RLP-SPG (ours)	37.3

Table 8: Performance of RLP and baselines on the MMLU benchmark. The scores are obtained by running InstructEval.