ODIN: Disentangled Reward Mitigates Hacking in RLHF

Lichang Chen [*1] **Chen Zhu** [*2] **Jiuhai Chen** [1] **Davit Soselia** [1]
Tianyi Zhou [1] **Tom Goldstein** [1] **Heng Huang** [1] **Mohammad Shoeybi** [3] **Bryan Catanzaro** [3]

Abstract

In this work, we study the issue of reward hacking on the response length, a challenge emerging in Reinforcement Learning from Human Feedback (RLHF) on LLMs. A well-formatted, verbose but less helpful response from the LLMs can often deceive LLMs or even human evaluators and achieve high scores. The same issue also holds for some reward models in RL. To address the challenges in both training and evaluation, we establish a more reliable evaluation protocol for comparing different training configurations, which inspects the trade-off between LLM evaluation score and response length obtained by varying training hyperparameters. Based on this evaluation, we conduct large-scale studies, where the results shed insights into the efficacy of hyperparameters and tricks used in RL on mitigating length bias. We further propose to improve the reward model by jointly training two linear heads to predict the preference, one trained to correlate with length and the other trained to decorrelate with length and therefore focusing more on the actual content. We then discard the length head in RL to ignore the spurious length reward. Experiments demonstrate that our approach eliminates the reward correlation with length, and improves the obtained policy by a significant margin.

1. Introduction

Reinforcement Learning from Human Feedback (RLHF) has emerged as a critical technique to elicit the capabilities from pretrained large language models (LLMs) to generate more helpful, honest, and harmless responses that align with human preferences (Ziegler et al., 2019; Askell et al., 2021; Ouyang et al., 2022), which has led to the success of ChatGPT (Schulman et al., 2022) and many other AI systems (Pichai, 2023; Anthropic, 2023; Touvron et al., 2023). RLHF trains a reward model (RM) on human preferences for the responses of given prompts, followed by training the language model to generate responses that maximize the learned reward through reinforcement learning. Such a paradigm simplifies human data collection, as acquiring human ratings is easier than collecting demonstrations for supervised fine-tuning. Moreover, it has been observed that RLHF has weak-to-strong generalization, where the policy becomes more creative than the supervision it receives (Burns et al., 2023).

Despite the promises, one subtle issue of RLHF is reward hacking, or reward model over-optimization, *i.e.*, the policy obtains a high reward but does not fulfill the actual objectives. It happens because the RM is not a perfect proxy of human preferences and has limited out-of-distribution (OOD) generalization, but the policy is a capable LLM that can learn to generate OOD examples to exploit the vulnerabilities of the RM (Hendrycks et al., 2021; Ramé et al., 2024). More critically, the human preference data can often be biased and inconsistent due to the difficulty and subjectivity of the task itself, flaws in the rating criteria, and the limited quality of raters. The most common pattern of reward hacking in practice is verbosity: the language models generate more tokens to make the response appear more detailed or better formatted after RLHF (usually for helpfulness) but the actual quality does not improve (Singhal et al., 2023; Wang et al., 2023b). This tendency is largely due to a preference among human raters for longer responses, which could be exploited by RM easily and cause the length hacking. Given the challenges in controlling the quality of human data, it becomes increasingly important and beneficial to study mitigating the impact of spurious features from the reward modeling and algorithmic perspective.

In this paper, we take a step towards mitigating reward hacking by conducting a comprehensive study on the impact of reward models and the RL algorithm on the verbosity and performance of the learned policy. Considering the challenges in model-based evaluations due to their biases (Zeng et al., 2023), *e.g.*, open-sourced LLMs climb up on AlpacaEval (Li et al., 2023a) leaderboard by utilizing the length

[*]Equal contribution, order is random. [1]University of Maryland, College Park [2]Meta, work done while at Nvidia [3]Nvidia. Correspondence to: Lichang Chen <bobchen@cs.umd.edu>, Chen Zhu <chenzhu@meta.com>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

bias of the judge GPT-4 (Liu, 2024), we first establish a more reliable evaluation protocol for comparing different training configurations, which gathers evaluation results from large-scale grid search under these configurations and compares the achieved performance on the Pareto front of evaluation score vs. length. This offsets the length biases and gives a holistic understanding of the optimal result each approach can achieve at different lengths to reduce the randomness of the conclusions due to the length bias in model-based evaluation. Under this setup, we investigate the effectiveness of hyperparameters and tricks in RL for reducing reward hacking on length, including reward clipping (Mnih et al., 2015) and length penalty (Singhal et al., 2023). While tuning and tricks can push up the Pareto front, we find it hard to conclude with simple principles for tuning this large set of hyperparameters. We seek to solve the issue from its root and eliminate the spurious length signal from the reward. To this end, we train a two-head reward model to disentangle representations for length from the actual preference and discard the length head during RL. The proposed reward disentangling method, ODIN[1], helps the policy achieve a higher Pareto front than previous results with a more expensive tuning budget, and the conclusion holds for both PPO (Schulman et al., 2017) and ReMax (Li et al., 2023b), showing the great potential of ODIN to improving the different RL-tuning algorithms and shed light for reducing the length hacking.

2. Related Works

Reinforcement Learning from Feedbacks. Since its origin on language models (Ziegler et al., 2019), RLHF has empowered the success of several epochal LLM systems (Schulman et al., 2022; OpenAI, 2023; Pichai, 2023; Anthropic, 2023; Team, 2023), and more diverse sources of preferences have been used to train the reward model (Bai et al., 2022; Lee et al., 2023) or provide feedbacks directly in RL (Liu et al., 2023). Since both human and LLM evaluators have biases, ODIN stays relevant for RLHF/RLAIF as long as a reward model needs to be trained. Most of the powerful conversational assistants (Ouyang et al., 2022; Schulman et al., 2022) in practice use PPO (Schulman et al., 2017) for RL and achieved significant improvement on the LLM's instruction-following ability. Many alternatives to PPO also have appeared showing promises for better learning from feedbacks. Most of these approaches are offline algorithms, which includes SLiC-HF (Zhao et al., 2023), DPO (Rafailov et al., 2023), IPO (Azar et al., 2023), KTO (Ethayarajh et al., 2023), RSO (Liu et al., 2024) and ReST (Gulcehre et al., 2023). They use humans or LLMs to annotate a large batch of LLM generations and then train the policy on the annotated demonstrations or preferences, without sampling from the policy during training. Offline algorithms can be less prone to reward hacking as the demonstrations are updated less frequently, but hacking can still happen in the long term. In this paper, we mainly focus on the online algorithms which are more widely adopted in practice.

[1]Odin sacrificed one eye for wisdom, similarly our RM discards the length head for more focus on the actual content.

Mitigating Reward Hacking in RLHF. Shen et al. (2023) proposed to use a smaller reward model to learn the biases in the reward and a larger reward model to learn the true reward. Different from their approach, we explicitly train a linear projection on the shared reward model features to be correlated with length and remove such correlation from the other head. Language models sometimes have a contrary length bias where it favors generating shorter sequences. Sountsov & Sarawagi (2016) found encoder-decoder models tend to generate shorter sequences with beam search. Singhal et al. (2023) explored ways to reduce length increase for PPO, including regularizations for PPO (increasing KL regularization, omitting outputs beyond a length threshold and reward scaling), and improvements on reward model training data (including length balancing, confidence-based truncation and reward data augmentation with random preferred response as negative examples). Their mitigations in PPO were not able to prevent length increase compared to SFT, and the reward becomes lower compared with the setting without the regularizations. The improvements on the reward model either decrease reward model accuracy or fail to decrease correlation with length to significantly small values. Rewarded Soup (Rame et al., 2023) interpolates weights of policies trained to optimize different reward objectives, which can approximate more costly multi-policy strategies. Instead of interpolating the policies, WARM (Ramé et al., 2024) uses weight-averaged reward models to improve their OOD robustness and reduce reward hacking in RL.

LLM Evaluations. For instruction-following evaluation, current evaluations of SFT/RLHF models usually rely on the auto-raters, e.g., GPT-4, since it is a scalable and efficient way (Zheng et al., 2023a; Touvron et al., 2023). However, current open models utilize the length bias of the auto-rater, i.e., GPT-4 is more likely to give a higher score for longer responses compared to the short responses, to climb on Alpaca-Eval (Li et al., 2023a; Liu, 2024). To conduct fair and holistic evaluations of the actor models trained via our reward models, in our paper, we evaluate the models by comparing the Pareto front. As for benchmarks, we aim to evaluate the base capabilities of LLMs, *e.g.*, reasoning and commonsense. Since they are mostly gained from pertaining corpus (Zhou et al., 2023) and SFT/RLHF has limited data compared to the pretraining, we only expect it could still maintain the performance on benchmarks, e.g., on MMLU (Hendrycks et al., 2020), TruthfulQA (Lin et al., 2021).

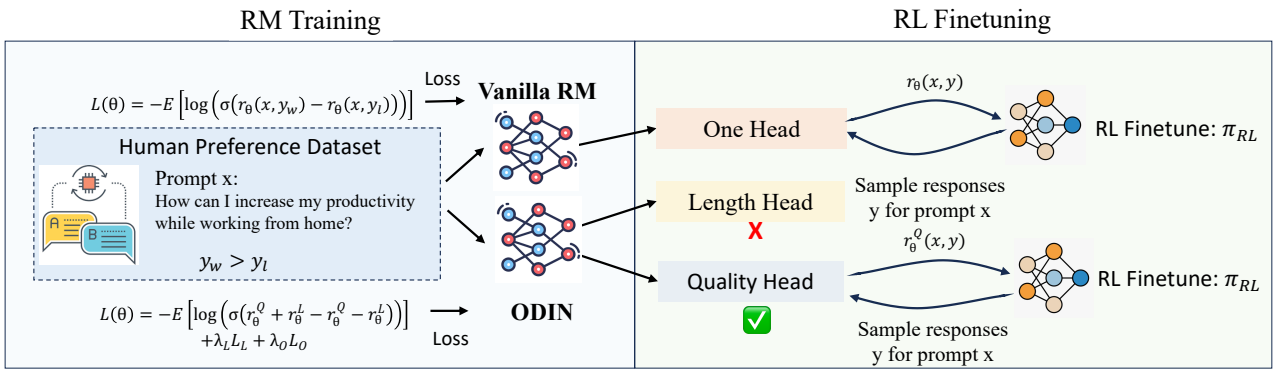


Figure 1: The explanation of ODIN. ODIN has two heads to predict two rewards. In RM training stage, ODIN is trained with the same human preference data as vanilla RM with a carefully designed loss to disentangle the length sinal and the quality signal into two heads. Only the quality head is involved in RL fine-tuning stage, and the length reward is discarded to reduce reward hacking on length.

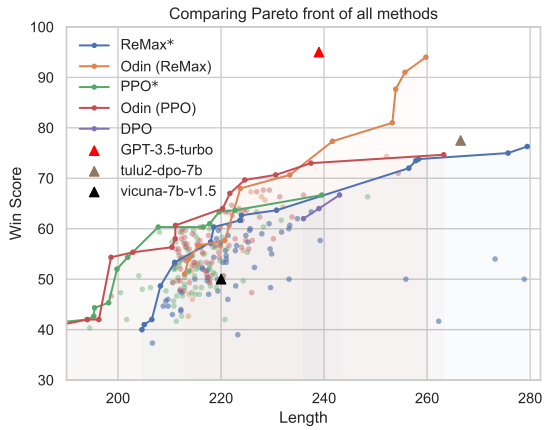


Figure 2: The main results of ODIN. We compare the Pareto front of PPO (Schulman et al., 2017), ReMax (Li et al., 2023b) trained with vanilla reward model and ODIN, as well as the model trained with DPO (Rafailov et al., 2023) loss. For ReMax* and PPO*, we aggregated results with reward clipping and length penalty for comparison, which involves a larger search space and compute budget than the ODIN results.

3. Preliminaries

We consider the RLHF pipeline widely adopted in the developments of LLMs (Ziegler et al., 2019; Stiennon et al., 2020; Ouyang et al., 2022; Touvron et al., 2023), which consists of three stages: (1) Supervised Fine-tuning (SFT); (2) Reward modeling: training the reward model based on the SFT checkpoint; (3) RL: using the SFT checkpoint as initialization and the reward model for feedback.

Reward Modeling. Same as (Stiennon et al., 2020; Ouyang et al., 2022; Touvron et al., 2023), we consider the approach where the reward model is initialized from a supervised fine-tuned LM, with a randomly initialized linear layer appended to the end to project the feature representation of the whole sequence into a scalar representing the reward. The reward model is trained to minimize the loss under the Bradley–Terry model (Bradley & Terry, 1952) on pair-wise comparisons of model responses as

$$\mathcal{L}(\theta) = -\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}}\left[\log\left(\sigma\left(r_{\boldsymbol{\theta}}\left(x,y_w\right)-r_{\boldsymbol{\theta}}\left(x,y_l\right)\right)\right)\right], \tag{1}$$

where $r_{\boldsymbol{\theta}}(x,y)$ is the scalar reward from the reward model with trainable parameters $\boldsymbol{\theta}$ for prompt x and the response y; y_w and y_l are the chosen and rejected responses respectively, and $\sigma(\cdot)$ denotes the sigmoid function.

RL Objective. Different from SFT, RL fine-tuning stage does not require golden responses for supervision. Instead, the reward model is used as a proxy of human feedback on the responses generated by the policy throughout training. Specifically, it fine-tunes the parameters $\boldsymbol{w}$ of the policy $\pi_{\boldsymbol{w}}$ by maximizing the the following objective:

$$\mathbb{E}_{(x,y)\sim\mathcal{D}_{\pi_{\boldsymbol{w}}}}\left[r_{\boldsymbol{\theta}}(x,y)\right]-\beta\mathbb{D}_{\mathrm{KL}}\left[\pi_{\boldsymbol{w}}(y\mid x)||\pi^{\mathrm{SFT}}(y\mid x)\right], \tag{2}$$

where the SFT policy π^{SFT} is used as initialization of $\pi_{\boldsymbol{w}}$, $\mathcal{D}_{\pi_{\boldsymbol{w}}}=\{(x,y)|x\sim\mathcal{D}_{\mathrm{RL}},y\sim\pi_{\boldsymbol{w}}(y|x)\}$, and $\beta>0$ is a constant adjusting strength of the KL regularization. The KL regularization term is used to mitigate reward hacking by preventing the policy $\pi_{\boldsymbol{w}}$ from drifting away from the SFT model π^{SFT} (Stiennon et al., 2020; Ouyang et al., 2022). In practice, the KL term is replaced with some estimator, which makes Eq. (2) equivalent to maximizing some auxiliary reward $\hat{r}(x,y)$. Following Stiennon et al. (2020), we consider the naïve estimator in this paper, and define the

auxilary reward as

$$\hat{r}(x, y) = r_{\boldsymbol{\theta}}(x, y) - \beta \log \frac{\pi_{\boldsymbol{w}}(y|x)}{\pi^{\text{SFT}}(y|x)}. \tag{3}$$

See Schulman (2020) for unbiased estimator of KL.

RL Algorithms. Different RL algorithms can be used to maximize $\hat{r}(x, y)$. We compare two options to see how existing mechanisms in RL algorithms can reduce reward hacking in RLHF: the simpler REINFORCE with baseline (Williams, 1992), and the more sophisticated PPO (Schulman et al., 2017). For REINFORCE, we consider the ReMax variant (Li et al., 2023b), which saves memory and compute significantly by replacing the value network with the reward on the greedy decoding of the current policy. Li et al. (2023b) proved that similar to REINFORCE, ReMax has an unbiased gradient estimate and reduces gradient variance under certain assumptions. ReMax maximizes the following objective with gradient ascent on $\boldsymbol{w}$:

$$\mathbb{E}_{(x,y)\sim\mathcal{D}_{\pi_{\boldsymbol{w}}}} [\hat{r}(x, y) - \hat{r}(x, \bar{y})] \log \pi_{\boldsymbol{w}}(y|x), \tag{4}$$

where $\bar{y}$ is the greedy sampling from $\pi_{\boldsymbol{w}}$.

PPO is a more prevalent option adopted by many works (Ziegler et al., 2019; Stiennon et al., 2020; Ouyang et al., 2022; Touvron et al., 2023). For clarity, we provide details of PPO in the context of RLHF for LLMs in Algorithm 1 in Appendix. PPO maximizes the clipping objective

$$\mathbb{E}_{\mathcal{D}_{\boldsymbol{w}_{old}}} \left[\min \left\{ \frac{\pi_{\boldsymbol{w}}(y|x)}{\pi_{\boldsymbol{w}_{\text{old}}}(y|x)} \hat{A}, \text{clip} \left(\frac{\pi_{\boldsymbol{w}}(y|x)}{\pi_{\boldsymbol{w}_{\text{old}}}(y|x)}, 1-\epsilon, 1+\epsilon \right) \hat{A} \right\} \right], \tag{5}$$

where $\epsilon > 0$ is a constant for clipping, $\frac{\pi_{\boldsymbol{w}}(y|x)}{\pi_{\boldsymbol{w}_{\text{old}}}(y|x)}$ is the likelihood ratio, $\hat{A}$ is the advantage usually estimated by GAE (Schulman et al., 2015) as a function of the value estimate and the reward. Intuitively, this clipping objective can help reduce reward hacking. It prevents the model from becoming over-confident on samples with positive advantage to over-optimize the reward by stopping optimizing on samples when their likelihood ratio $\frac{\pi_{\boldsymbol{w}}(y|x)}{\pi_{\boldsymbol{w}_{\text{old}}}(y|x)} > 1 + \epsilon$. See our results in Figure 3 (a).

4. Mitigating Reward Hacking in Practice

In this section, we first establish a more reliable evaluation for comparing different methods, which uses the length of the generated response $L(y)$ as an indicator of the degree of reward hacking. Then, we study the impact of RL hyperparameters and tricks on the Pareto front of model-based or human evaluation metrics against $L(y)$, and propose a more reliable approach by training a reward model that disentangles the spurious length correlation from the actual reward on contents.

4.1. Evaluation

It is challenging to evaluate the policy automatically through LLM evaluators, as the policy has been optimized against a reward model initialized from a LLM, and these LLM evaluators can often be biased in practice (Zeng et al., 2023; Zheng et al., 2023a; Singhal et al., 2023). Previous works studying reward hacking have been using a ground-truth reward model to generate the preference data for training and evaluate the policy using the ground-truth reward (Gao et al., 2023; Ramé et al., 2024), but here we want to develop an approach that applies to reward models trained on real human preference data. To achieve this, we look at the model-based evaluation metric S against the average response length L on the evaluation set, and compare the Pareto front achieved by each method or configuration. We consider the response length because it is easy to measure and well-reflects the degree of reward hacking in RLHF for LLMs; in practice, the policy tends to generate longer responses when reward hacking happens (Ramé et al., 2024; Wang et al., 2024). A better method or configuration should achieve higher score when L is the same, therefore a higher Pareto front in the plots. We mainly use model-based evaluations in our studies, where we compare responses generated by the policy against the responses generated by the SFT baseline. We then use the following win score as the metric:

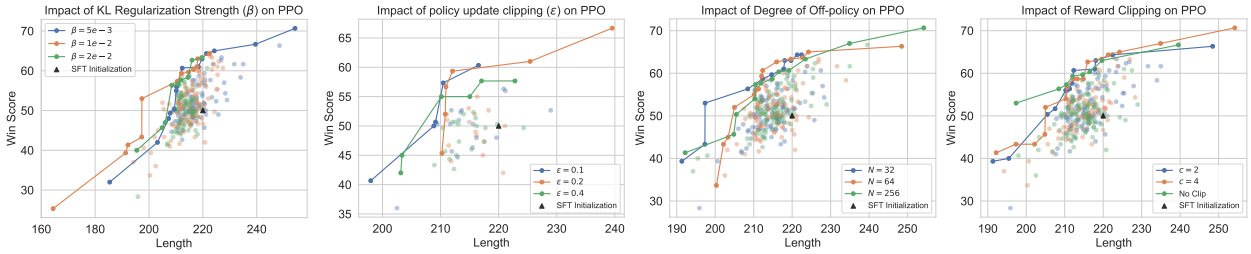
$$\text{Win Score} = 50 + 100 \times \frac{n_{win} - n_{lose}}{n}, \tag{6}$$

where n_{win} (n_{lose}) is the number of examples rated as winning (losing) against the baseline, and n is the total number of evaluation examples. Win Score $\geq$ 50 when the test model is no worse than the baseline. See Section 5 for more details.

4.2. How much hacking can we mitigate by tuning RL?

We investigate how much the hyperparameters and tricks used in RL can reduce reward hacking and improve evaluation results. While this helps to some extent, we find it can be hard to obtain a simple principle that will guarantee a significantly better Pareto front.

PPO Hyperparameters. The KL regularization is introduced into the RL objective to explicitly prevent reward hacking. In Figure 9, we show that larger KL weight β can indeed prevent excessive length increase, but the win score becomes worse and the policy becomes closer to SFT initialization. In Figure 3 (a), we show the effect of KL is marginalized when reward clipping is introduced. As mentioned in Section 3, the clipping objective can potentially reduce reward hacking. From Figure 3 (b), we find it is indeed the case, with smaller ϵ bringing around 2.5 points of improvement on the Pareto front. However, the effect becomes more complicated when reward clipping is introduced; see Figure 10. Another mechanism that can po-



(a) (b) (c) (d)

Figure 3: (a) Results under different β's, when sweeping η, ϵ, N and c. The effect is marginal when the length is around SFT init. We show the version without reward clipping in Figure 9. (b) Results under different PPO clipping ϵ, when disabling reward clipping and sweep η, N. More conservative ϵ reduces hacking and improves results, but the trend becomes complicated when enabling reward clipping (Figure 10). (c) Results under different sizes of experience batch N, when sweeping η, β, ϵ and c. We use batch size $b = 32$, so $N = 32, 64, 256$ correspond to 0%, 50% and 87.5% "off-policy" samples, and ϵ clipping is ineffective when $N = 32$. Surprisingly, larger N is not beneficial. (d) Results under different reward clipping thresholds c, when sweeping η, β, ϵ, N. Certain c can outperform the baseline without clipping, but this requires tuning.

tentially alleviate reward hacking is to sample the responses from the old policy, which should reduce the chance of sampling from a hacking policy. This is effective when $N > b$, where the policy is trained on $(N - b)$ "off-policy" experiences in each PPO inner epoch. Surprisingly, in Figure 3 (c), we show that a higher degree of off-policy makes it more likely to generate longer responses, and the win score around the length of π^{SFT} is not as high as pure on-policy ($N = b$), where even the PPO clipping ϵ is ineffective ($\rho_{\pi_{\boldsymbol{w}_{old}}}(x, y) \equiv 1$). We leave studying GAE as future work.

Reward Clipping. Reward clipping is widely adopted by previous works like (Mnih et al., 2015; Engstrom et al., 2020) as well as the Deepspeed RLHF implementation. Specifically, we clip the reward from the reward model and use the clipped auxiliary reward as

$$\hat{r}^{\text{clip}}_{\boldsymbol{\theta}}(x, y) = \text{clip}(r_{\boldsymbol{\theta}}(x, y), -c, c) - \beta \log \frac{\pi_{\boldsymbol{w}}(y|x)}{\pi^{\text{SFT}}(y|x)}, \quad (7)$$

where $c > 0$ is a constant. Reward clipping can alleviate reward hacking in that it prevents the policy from trying to achieve higher rewards by generating hacking patterns to the reward model. In Figure 3 (d), we do observe that a proper c leads to a higher win score for PPO at length close to the SFT init. In Figure 8, we show that a proper clipping can also improve ReMax, but a more aggressive clipping (e.g., $c = 1$) can hinder effective learning by preventing the policy from exploiting higher reward responses. As a result, similar to the recommendation in (Zheng et al., 2023b), careful tuning is required to use reward clipping successfully in practice.

Length Penalty. A more straightforward way to prevent reward hacking on length is to explicitly penalize longer responses. Singhal et al. (2023) adds a length penalty proportional to the response length using the standard deviation of reward as the coefficient. However, to eliminate the correlation with length, we also need to consider the covariance between the reward and length, which can be constantly changing during RL due to shifts in the distribution of generations. Therefore, we simply make the coefficient a tunable constant $\alpha > 0$, and change the auxiliary reward into $\hat{r}^{\text{lp}}_{\boldsymbol{\theta}}(x, y) = \hat{r}_{\boldsymbol{\theta}}(x, y) - \alpha * L(y)$, where $L(y)$ is number of tokens in the response y. In Figure 4, we show that length penalty makes $\hat{r}^{\text{lp}}_{\boldsymbol{\theta}}(x, y)$ less affected by length and improves the Pareto front, but is not as effective as ODIN, which bakes length decorrelation into RM training to make the reward more reliable and does not add new hyperparameters to RL.

4.3. Reward Disentanglement: a more reliable approach

In the previous section, we have shown the difficulty of reducing the simple reward hacking pattern of length through tuning and tricks in RL against a vanilla reward model. Here, we demonstrate a better approach where we train the reward model to disentangle the actual reward from spurious rewards correlating with patterns that do not always represent the quality of the response but can add to the vulnerabilities of the reward model and make it easier to be exploited by the policy. Different from previous approaches that learn and integrate rewards from multiple types of preferences (Wu et al., 2023), we discard the spurious rewards during RL. We find this removes the need to use reward clipping and length penalty to prevent length increase and achieves better results without excessive tuning on the disentangled reward model.

Learning Multiple Rewards on Shared Representations. To minimize the overhead, we increase the output dimension of the final linear layer of the RM to predict different rewards. This is sufficient to separate the spurious reward from the original scalar reward, since the RM is a pretrained LLM

with enough capacity, and the observed reward hacking is a consequence of spurious rewards being exploited. Specifically in the case of disentangling length reward $r^{\mathrm{L}}_{\boldsymbol{\theta}}(x, y)$ and the actual reward reflecting quality of the response $r^{\mathrm{Q}}_{\boldsymbol{\theta}}(x, y)$, we represent the full reward from the feature representation as $r^{\mathrm{Q}}_{\boldsymbol{\theta}}(x, y) + r^{\mathrm{L}}_{\boldsymbol{\theta}}(x, y)$, and consider the following ranking loss for reward model:

$$\mathcal{L}^{\mathrm{R}}_{\boldsymbol{\theta}}(x, y_w, y_l) = -\mathbb{E}\big[\log\big(\sigma\big(r^{\mathrm{Q}}_{\boldsymbol{\theta}}(x, y_w) + r^{\mathrm{L}}_{\boldsymbol{\theta}}(x, y_w) - r^{\mathrm{Q}}_{\boldsymbol{\theta}}(x, y_l) - r^{\mathrm{L}}_{\boldsymbol{\theta}}(x, y_l)\big)\big)\big], \tag{8}$$

which equivalently trains the model to decompose the original projection weights into the sum of two sets of projection weights, and should have better capacity.

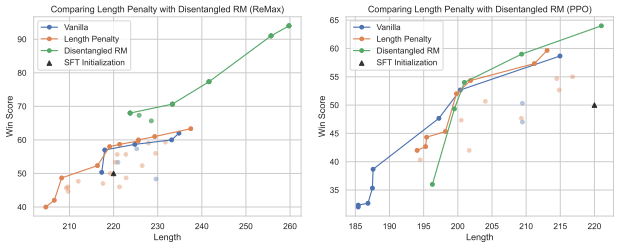


Figure 4: Comparing the effect of length penalty and ODIN on ReMax and PPO. For both ReMax and PPO, length penalty (LP) can improve the Pareto front, but not as significant as ODIN. Due to limited computes, we ran less experiments for LP, so this set was selected so that each method shares the same RL hyper-parameters as LP. See Appendix E.4 for hyperparameters considered.

Disentangling the Rewards. We consider the case when supervision can be added to all but one of the rewards, since unsupervised learning of disentangled representations is impossible without inductive biases on both the models and the data for generative models (Locatello et al., 2019). In the case of length and quality, we first design the loss to enhance the length correlation of r^{L} while minimizing that for r^{Q} as follows:

$$\mathcal{L}^{\mathrm{L}}_{\boldsymbol{\theta}}(x, y) = \Big|\rho(r^{\mathrm{Q}}_{\boldsymbol{\theta}}(x, y), L(y))\Big| - \rho(r^{\mathrm{L}}_{\boldsymbol{\theta}}(x, y), L(y)), \tag{9}$$

where $L(y)$ is number of tokens in the response y, and $\rho(X, Y)$ is the Pearson correlation of X, Y computed within the global minibatch. To compute ρ within the global minibatch when data parallel is enabled, we gather the rewards and lengths from all devices only in the forward pass, which leads to the correct gradients for parameters $\boldsymbol{\theta}$ in the backward pass since the reward predictions are independent of each other in the Transformer architecture (Vaswani et al., 2017). Note we use $\mathcal{L}^{\mathrm{L}}_{\boldsymbol{\theta}}(x, y)$ as a regularization added to the ranking loss in Eq. 8. When $\mathcal{L}^{\mathrm{L}}_{\boldsymbol{\theta}}(x, y)$ is minimized to -1, $r^{\mathrm{L}}_{\boldsymbol{\theta}}$ and $r^{\mathrm{Q}}_{\boldsymbol{\theta}}$ will have zero correlation, which can be beneficial since it indicates $r^{\mathrm{Q}}_{\boldsymbol{\theta}}$ and $r^{\mathrm{L}}_{\boldsymbol{\theta}}$ did not co-adapt to reduce the ranking loss and both heads are learning independently to maximize their predictive power. However, perfect correlation and decorrelation can be hard to achieve in practice, since we usually train on minibatches, and we want to generalize the RM to OOD examples in RL.

To further enhance disentanglement between $r^{\mathrm{Q}}_{\boldsymbol{\theta}}$ and $r^{\mathrm{L}}_{\boldsymbol{\theta}}$ and learn both more effectively, we enforce the orthogonality of their projection weights. Specifically, let $\mathbf{W}_{\mathrm{Q}}, \mathbf{W}_{\mathrm{L}} \in \mathbb{R}^{1\times d}$ be the linear projection for quality and length rewards. We introduce the orthogonality loss

$$\mathcal{L}^{\mathrm{O}}_{\boldsymbol{\theta}} = |\mathbf{W}_{\mathrm{Q}}\mathbf{W}_{\mathrm{L}}^{T}|. \tag{10}$$

When enforced together with $\mathcal{L}^{\mathrm{R}}_{\boldsymbol{\theta}}(x, y_w, y_l)$ and $\mathcal{L}^{L}_{\boldsymbol{\theta}}(x, y)$, $\mathcal{L}^{\mathrm{O}}_{\boldsymbol{\theta}}$ can be beneficial for disentangling the feature representations of length and quality into orthogonal subspaces, because the feature representation of the RM will learn to align the length and quality representations with $\mathbf{W}_{\mathrm{L}}$ and $\mathbf{W}_{\mathrm{Q}}$ to minimize other losses, and $\mathbf{W}_{\mathrm{L}}$ and $\mathbf{W}_{\mathrm{Q}}$ are learned to be orthogonal. In Table 1 and Figure 5, we show that adding $\mathcal{L}^{\mathrm{O}}_{\boldsymbol{\theta}}$ further reduced the length correlation, and lead to even better RL policies.

Note that both $\mathcal{L}^{\mathrm{L}}_{\boldsymbol{\theta}}(x, y)$ and $\mathcal{L}^{\mathrm{O}}_{\boldsymbol{\theta}}$ can be minimized when $\mathbf{W}_{\mathrm{Q}} = 0$. To prevent this from happening and improve training dynamics, we add weight normalization (Salimans & Kingma, 2016) to both $\mathbf{W}_{\mathrm{Q}}$ and $\mathbf{W}_{\mathrm{L}}$ before computing the losses and predicting the rewards.

Summary. We train ODIN with weight-normalized $\mathbf{W}_{\mathrm{Q}}$ and $\mathbf{W}_{\mathrm{L}}$ to minimize the following loss

$$\mathcal{L}^{\mathrm{R}}(x, y_w, y_l) + \lambda_{\mathrm{L}}\mathcal{L}^{\mathrm{L}}_{\boldsymbol{\theta}}(x, y_w) + \lambda_{\mathrm{L}}\mathcal{L}^{\mathrm{L}}_{\boldsymbol{\theta}}(x, y_l) + \lambda_{\mathrm{O}}\mathcal{L}^{\mathrm{O}}_{\boldsymbol{\theta}}, \tag{11}$$

where $\lambda_{\mathrm{L}}, \lambda_{\mathrm{O}} > 0$ are constants for regularization strength. In RL, we only use the r^{Q} from ODIN. Without excessive tuning, we find setting $\lambda_{\mathrm{L}} = \lambda_{\mathrm{O}} = 1$ to yield reasonably good results for RL outperforming many baselines in Figure 2. In Table 1, we show that using only the quality reward $r^{\mathrm{Q}}_{\boldsymbol{\theta}}$ of the disentangled RM maintains the validation accuracy compared with the baseline, while drastically reducing correlation with length.

5. Experiments

5.1. Settings

Dataset. We use the OpenAssistant dataset (Köpf et al., 2023), a human-generated, human-annotated assistant-style conversation corpus with over 10,000 complete and fully annotated conversation trees. Our preprocessing of this dataset involves the following steps: (1) We transform all items into a dialogue format (see Appendix E.5) and discard samples with non-English prompts or responses. (2) For prompts associated with multiple ranked responses, we retain all these responses by considering all the pairwise comparisons. This results in $k(k-1)/2$ unique comparisons when a prompt has k ranked responses.

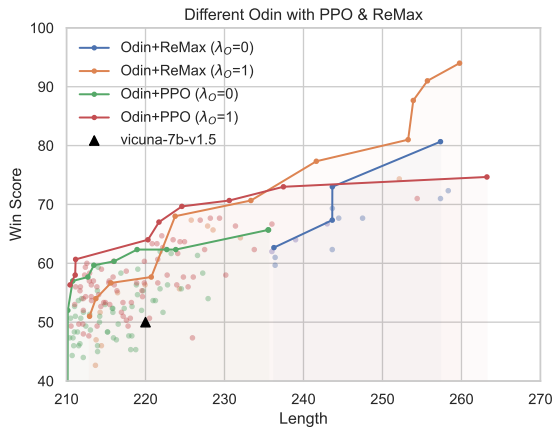


Figure 5: The performance of the policies trained by two different ODIN's. $\lambda_O = 1.0$ denotes the ODIN trained using both Length Loss and Orthogonal loss while $\lambda_O = 0.0$ represents the reward model only trained with Length loss.

Models and Training. We use Vicuna-7b as the base model π_{SFT},[2] which is a SFT model with decent instruction-following capability. We fine-tune the reward model from Vicuna-7B with randomly initialized projection layer appended to the last layer. We also initialize the policy $\pi_{\boldsymbol{w}}$ from the same Vicuna-7b. All experiments are implemented with DeepSpeed-Chat (Yao et al., 2023) and Huggingface Transformers (Wolf et al., 2020), running on 8 NVIDIA A100 80GB GPUs. We tried different learning rates from $\{1e-5, 3e-5, 5e-5\}$ with batch size 128 for tuning both the baseline RM and ODIN on 22k preference data for 3 epochs, and picked the one with the highest validation accuracy for both. We fine-tune all the parameters in the models for both RM training and RL without freezing anything or using adapters. To evaluate how the efficacy of ODIN can transfer across different RL algorithms, we experiment with ReMax (Li et al., 2023b), an efficient and effective version of REINFORCE without a value network, and Proximal Policy Optimization (PPO)(Schulman et al., 2017). We provide more details on the hyperparameters in Appendix E. To compare with other alternatives for utilizing human feedback, we re-implement Direct Preference Optimization (DPO) (Rafailov et al., 2023) and use it to tune the same Vicuna 7B on the same Open Assistant human preference data as we train our reward models. For reference, we also evaluate and compare with another open-sourced models trained with DPO, `tulu-2-dpo-7b` (Ivison et al., 2023), which is based on the same pretrained model (Llama 2 7B) as Vicuna 7B.

Evaluation Metrics. Our main focus is on open-ended generation. Incorporating recent advances in automated evaluation (Dubois et al., 2023; Zheng et al., 2023a; Chiang et al., 2023), we use model-based metrics for large-scale studies. We use GPT-4 (OpenAI, 2023) as the judge to compare two responses for each prompt. We use the same prompt as Chen et al. (2023), where GPT-4 is asked to give a rating for each response when both responses are present in the input; see Appendix D for details. By comparing the two ratings, the result can be win, tie or lose. To counter positional bias in GPT-4 ratings (Wang et al., 2023a), we collect two sets of ratings by alternating the order of test and baseline model responses. A winning response must receive at least one win and at most one tie. This protocol can mitigate the positional bias and improve the rating quality of GPT-4 as reported by Chiang et al. (2023). After counting number of win, tie and lose for the test model, we use the Win Score as defined in Eq. 6 as the aggregated metric. To show the relative improvement each model obtained compared with the SFT baseline (Vicuna-7B), for each prompt, we use one response generated by Vicuna-7B, and collect the other one from the RL policy we want to evaluate in all our GPT-4 evaluations. Taking the length bias in the GPT-4 evaluations into account (Wang et al., 2023b), a real improvement is achieved with higher Win Score at a similar average length, therefore we use the Pareto front achieved by each method for the final judgement. To validate the results, we also select best models at different length scales and compare them with human studies.

Benchmarks. For the GPT-4 evaluation and human studies, we use prompts from the LIMA (Zhou et al., 2023)

Table 1: Direct reward model evaluation. We calculate the Pearson correlation ρ, Spearman's r_s, and Kendall's τ between response length $L(y)$ and reward score $r(x, y)$. Note 66% of this preference data test set has the chosen response longer than rejected response.

	ρ	τ	r_s	Val Acc.
Baseline RM	0.451	0.422	0.338	70.1
$\lambda_L = 1.0, \lambda_O = 0.0$	-0.05	-0.04	-0.05	70.1
$\lambda_L = 1.0, \lambda_O = 1.0$	-0.03	0.008	0.006	69.2

Table 2: The evaluation results on the separated test set, where Chosen-L (Rejected-L) means the chosen response is longer (shorter) than the rejected response. ODIN obtains more balanced accuracies on the two sets, showing less length bias.

RM	Chosen-L	Rejected-L.
Baseline RM	**86.8%**	39.3%
$\lambda_L = 1.0, \lambda_O = 0.0$	83.3%	44.8%
$\lambda_L = 1.0, \lambda_O = 1.0$	82.4%	**45.4%**

[2]https://huggingface.co/lmsys/vicuna-7b-v1.5.

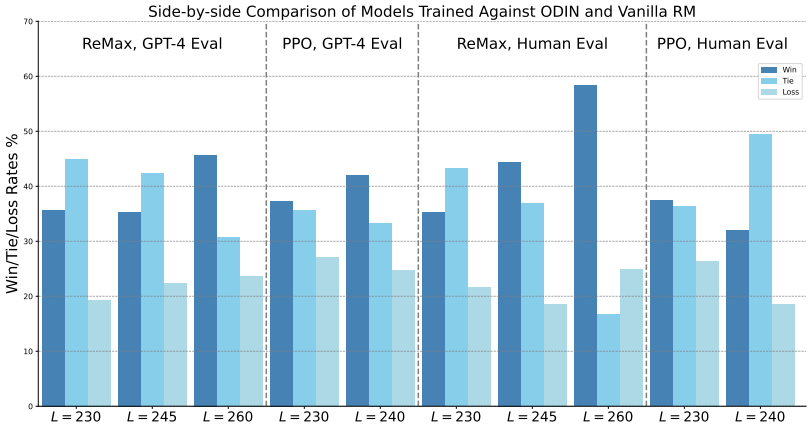


Figure 6: The human evaluation & GPT-4 pairwise-comparison results between models trained with our ODIN and length penalty baselines at the lengths of 230, 245, and 260. These lengths are close to our SFT initialization, the GPT-3.5 Turbo and the `tulu-2-dpo-7b` models, respectively. Note that all the models selected are on the Pareto front. The baseline here is the ReMax & PPO with Length Penalty.

test-set, which contains 300 open-ended prompts in total. We also evaluate the performance of our models on benchmarks on specific model capabilities. Following InstructEval (Chia et al., 2023), we test the trained policy π_{RL} on BBH (Suzgun et al., 2022), MMLU (Hendrycks et al., 2020), DROP (Dua et al., 2019), and TruthfulQA (Lin et al., 2021) to evaluate the model's ability on challenging task solving, multi-task, Math, and Truthfulness. We expect the trained policy to improve on the LIMA evaluations, and maintains its ability on the benchmarks (BBH, MMLU, DROP and TruthfulQA), which is not targeted by the Open Assistant data we are using but was obtained from pretraining.

5.2. Results

RM Evaluation. The efficacy of the reward models is best judged by the performance of the policy they supervised, which is demonstrated by the large-scale studies based on GPT-4 evaluation in Figure 2 and our human studies in Figure 6. For direct comparison of the reward models, we mainly evaluate the accuracy of distinguishing the chosen and rejected responses on the Open Assistant test set. We also look at the correlation of the reward with length to measure how much the reward prediction relies on length. Besides the linear Pearson correlation ρ, which we explicitly used for training ODIN, we also consider the rank correlations, Kendall's τ and Spearman's r_s (See Appendix C), to see how much the reward rankings correlate with length rankings, as the reward model is optimized for ranking. We report results of RMs with the highest validation accuracy in Table 1. It shows that, despite only being trained to minimize the Pearson correlation with length, the rank correlations are also eliminated, which helps understand why ODIN outperforms the linear length penalty in Figure 4 as it can only remove linear correlation. Without exploiting length information, ODIN is able to maintain most of the prediction accuracy on preference data, and the drop is insignificant considering the significant reduction in correlation and the 66% natural length bias in the preference data. This indicates that r^{Q} better utilized the actual content for rankings.

Automatic Evaluation. The main results are shown in Figure 2, where the Pareto front of the policy $\pi_{\boldsymbol{w}}$ trained by ODIN is always higher than that of the respective baselines (PPO* and ReMax*) when $L(y) \geq 210$; $L(y) < 210$ may indicate bad quality as the SFT model tuned on high-quality demonstrations has $L(y) = 220$. Note we included additional tricks (reward-clipping and length-penalty) and used more compute budget for PPO* and ReMax* shown in the plots. We also provide head-to-head GPT-4 evaluations of the best models of each method in Figure 6.

Human Studies. We further conduct human studies involving 8 college students as participants rating the quality of generated responses. Each rater evaluates 90 samples, with at least three ratings obtained for each sample. Due to the limited budget, we sample 60 prompts from the LIMA test set in each group of evaluation. Since human evaluations can also be biased toward longer or shorter responses, we select models with similar average lengths on the Pareto front of each method for comparisons. For each sample, we presented raters with the original prompt as well as two randomly positioned responses. Referring to the guideline, the rater will choose a better response or rate both as similar. The guideline asks raters to consider the following criteria: Alignment with the User's Intent, Clarity and Preci-

Table 3: The benchmark results of the trained policies $\pi_{\boldsymbol{w}}$. We select the policies with different average response lengths (annotated in parenthesises) on the Pareto front. Vanilla: the policy trained with the baseline RM. TQA(mc1): TruthfulQA(mc1). SFT Init: Vicuna-7B.

Datasets	SFT Init	ODIN (230)	Vanilla (230)	ODIN (245)	Vanilla (245)	ODIN (260)	Vanilla (260)
BBH	36.92	37.07	36.94	37.09	36.70	37.10	37.55
Drop	29.02	29.05	28.70	29.10	28.91	28.94	28.27
MMLU	49.81	49.86	49.85	49.83	49.74	49.87	49.96
TQA(mc1)	32.68	34.64	33.90	34.67	33.89	34.63	33.66

sion, Directness and Relevance, and Efficiency and Brevity. (See Appendix B for details.) The results can be seen in Figure 6 where all the examined models trained with ODIN are more preferred than the baselines, with the difference being the most significant at the highest length of 260.

Results on Benchmarks. We show the results in Table 3. We observe improvements in TruthfulQA, which may come from a better understanding of the questions after RLHF. They also maintain the performance for all other tasks compared to the SFT initialization. It is worth pointing out that on every length scale, the policies trained by ODIN could perform better than those trained by the vanilla reward model.

6. Conclusion

In this work, we embark on an exploration to address the challenge of reward hacking in RLHF, focusing particularly on the issue of verbosity as a form of reward hacking. To combat this, we first introduce a more reliable evaluation protocol which evaluates different methods by the score-verbosity trade-off. We conduct extensive experiments to verify the impact of hyperparameters and tricks (reward clipping and length penalty) on reward hacking. While we observed some trends for PPO clipping and the replay buffer size, the best results of baselines come from tuning all these dimensions, and it becomes hard to draw definitive conclusions about how these hyperparameters should be tuned when applied all together. We seek to resolve the issue from its root and propose ODIN, a novel approach designed to disentangle representation of the content quality from the lengths of responses. ODIN demonstrates notable improvements on the Pareto front, which transfers across two RL algorithms (ReMax and PPO). This advancement not only showcases the effectiveness of our approach but also sheds light on future research in RLHF.

Impact Statement

By striving to eliminate reward hacking, we contribute to the development of AI systems that are more trustworthy and aligned with ethical standards. Our approach shows great potential for LLMs, such as those used in conversational agents, to prioritize the quality of information over mere verbosity, thereby enhancing the user experience and preventing the dissemination of misleading or unnecessarily verbose information.

Our work also has significant societal consequences. As LLMs become increasingly integrated into our daily lives, ensuring their responses are both helpful and concise can lead to more efficient communication and information exchange. This has implications for educational technology, customer service, and any domain where LLMs are employed to interact with humans. By reducing the propensity for generating overly verbose responses, we can improve accessibility, particularly for individuals who may struggle with processing large amounts of information quickly.

Acknowledgement

LC Chen and H Huang were partially supported by NSF IIS 2347592, 2347604, 2348159, 2348169, DBI 2405416, CCF 2348306, CNS 2347617.

References

Anthropic. Anthropic introducing claude., March 2023. URL https://www.anthropic.com/index/introducing-claude.

Askell, A., Bai, Y., Chen, A., Drain, D., Ganguli, D., Henighan, T., Jones, A., Joseph, N., Mann, B., DasSarma, N., et al. A general language assistant as a laboratory for alignment. *arXiv preprint arXiv:2112.00861*, 2021.

Azar, M. G., Rowland, M., Piot, B., Guo, D., Calandriello, D., Valko, M., and Munos, R. A general theoretical paradigm to understand learning from human preferences, 2023.

Bai, Y., Kadavath, S., Kundu, S., Askell, A., Kernion, J., Jones, A., Chen, A., Goldie, A., Mirhoseini, A., McKinnon, C., et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022.

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

Burns, C., Izmailov, P., Kirchner, J. H., Baker, B., Gao, L., Aschenbrenner, L., Chen, Y., Ecoffet, A., Joglekar, M., Leike, J., et al. Weak-to-strong generalization: Eliciting strong capabilities with weak supervision. *arXiv preprint arXiv:2312.09390*, 2023.

Chen, L., Li, S., Yan, J., Wang, H., Gunaratna, K., Yadav, V., Tang, Z., Srinivasan, V., Zhou, T., Huang, H., and Jin, H. Alpagasus: Training a better alpaca with fewer data, 2023.

Chia, Y. K., Hong, P., Bing, L., and Poria, S. Instructeval: Towards holistic evaluation of instruction-tuned large language models, 2023.

Chiang, W.-L., Li, Z., Lin, Z., Sheng, Y., Wu, Z., Zhang, H., Zheng, L., Zhuang, S., Zhuang, Y., Gonzalez, J. E., Stoica, I., and Xing, E. P. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality, March 2023. URL https://lmsys.org/blog/2023-03-30-vicuna/.

Dua, D., Wang, Y., Dasigi, P., Stanovsky, G., Singh, S., and Gardner, M. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *Proc. of NAACL*, 2019.

Dubois, Y., Li, X., Taori, R., Zhang, T., Gulrajani, I., Ba, J., Guestrin, C., Liang, P., and Hashimoto, T. B. Alpacafarm: A simulation framework for methods that learn from human feedback. *arXiv preprint arXiv:2305.14387*, 2023.

Engstrom, L., Ilyas, A., Santurkar, S., Tsipras, D., Janoos, F., Rudolph, L., and Madry, A. Implementation matters in deep policy gradients: A case study on ppo and trpo. *arXiv preprint arXiv:2005.12729*, 2020.

Ethayarajh, K., Xu, W., Jurafsky, D., and Kiela, D. Human-centered loss functions (halos). Technical report, Contextual AI, 2023.

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

Geng, X., Gudibande, A., Liu, H., Wallace, E., Abbeel, P., Levine, S., and Song, D. Koala: A dialogue model for academic research. Blog post, April 2023. URL https://bair.berkeley.edu/blog/2023/04/03/koala/.

Gulcehre, C., Paine, T. L., Srinivasan, S., Konyushkova, K., Weerts, L., Sharma, A., Siddhant, A., Ahern, A., Wang, M., Gu, C., et al. Reinforced self-training (rest) for language modeling. *arXiv preprint arXiv:2308.08998*, 2023.

Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., and Steinhardt, J. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

Hendrycks, D., Carlini, N., Schulman, J., and Steinhardt, J. Unsolved problems in ml safety. *arXiv preprint arXiv:2109.13916*, 2021.

Ivison, H., Wang, Y., Pyatkin, V., Lambert, N., Peters, M., Dasigi, P., Jang, J., Wadden, D., Smith, N. A., Beltagy, I., and Hajishirzi, H. Camels in a changing climate: Enhancing lm adaptation with tulu 2, 2023.

Köpf, A., Kilcher, Y., von Rütte, D., Anagnostidis, S., Tam, Z.-R., Stevens, K., Barhoum, A., Duc, N. M., Stanley, O., Nagyfi, R., ES, S., Suri, S., Glushkov, D., Dantuluri, A., Maguire, A., Schuhmann, C., Nguyen, H., and Mattick, A. Openassistant conversations – democratizing large language model alignment, 2023.

Lee, H., Phatale, S., Mansoor, H., Lu, K., Mesnard, T., Bishop, C., Carbune, V., and Rastogi, A. Rlaif: Scaling reinforcement learning from human feedback with ai feedback. *arXiv preprint arXiv:2309.00267*, 2023.

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

Li, Z., Xu, T., Zhang, Y., Lin, Z., Yu, Y., Sun, R., and Luo, Z.-Q. Remax: A simple, effective, and efficient reinforcement learning method for aligning large language models, 2023b.

Lin, S., Hilton, J., and Evans, O. Truthfulqa: Measuring how models mimic human falsehoods, 2021.

Liu, J., Zhu, Y., Xiao, K., Fu, Q., Han, X., Yang, W., and Ye, D. Rltf: Reinforcement learning from unit test feedback. *arXiv preprint arXiv:2307.04349*, 2023.

Liu, P. J., 2024. URL https://twitter.com/i/web/status/1750318955808583997.

Liu, T., Zhao, Y., Joshi, R., Khalman, M., Saleh, M., Liu, P. J., and Liu, J. Statistical rejection sampling improves preference optimization. *ICLR*, 2024.

Locatello, F., Bauer, S., Lucic, M., Raetsch, G., Gelly, S., Schölkopf, B., and Bachem, O. Challenging common assumptions in the unsupervised learning of disentangled representations. In *international conference on machine learning*, pp. 4114–4124. PMLR, 2019.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *nature*, 518(7540): 529–533, 2015.

Nakano, R., Hilton, J., Balaji, S., Wu, J., Ouyang, L., Kim, C., Hesse, C., Jain, S., Kosaraju, V., Saunders, W., et al. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*, 2021.

OpenAI. Gpt-4 technical report, 2023.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744, 2022.

Pichai, S. An important next step on our ai journey, February 2023. URL https://blog.google/technology/ai/bard-google-ai-search-updates/.

Rafailov, R., Sharma, A., Mitchell, E., Ermon, S., Manning, C. D., and Finn, C. Direct preference optimization: Your language model is secretly a reward model, 2023.

Rame, A., Couairon, G., Shukor, M., Dancette, C., Gaya, J.-B., Soulier, L., and Cord, M. Rewarded soups: towards pareto-optimal alignment by interpolating weights fine-tuned on diverse rewards. *arXiv preprint arXiv:2306.04488*, 2023.

Ramé, A., Vieillard, N., Hussenot, L., Dadashi, R., Cideron, G., Bachem, O., and Ferret, J. Warm: On the benefits of weight averaged reward models. *arXiv preprint arXiv:2401.12187*, 2024.

Salimans, T. and Kingma, D. P. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. *Advances in neural information processing systems*, 29, 2016.

Schulman, J. Approximating kl divergence, 2020. URL http://joschu.net/blog/kl-approx.html.

Schulman, J., Moritz, P., Levine, S., Jordan, M., and Abbeel, P. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Schulman, J., Zoph, B., Kim, C., Hilton, J., Menick, J., Weng, J., Uribe, J. F. C., Fedus, L., Metz, L., Pokorny, M., et al. Chatgpt: Optimizing language models for dialogue. *OpenAI blog*, 2022.

Shen, W., Zheng, R., Zhan, W., Zhao, J., Dou, S., Gui, T., Zhang, Q., and Huang, X.-J. Loose lips sink ships: Mitigating length bias in reinforcement learning from human feedback. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 2859–2873, 2023.

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

Sountsov, P. and Sarawagi, S. Length bias in encoder decoder models and a case for global conditioning. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 1516–1525, 2016.

Stiennon, N., Ouyang, L., Wu, J., Ziegler, D., Lowe, R., Voss, C., Radford, A., Amodei, D., and Christiano, P. F. Learning to summarize with human feedback. *Advances in Neural Information Processing Systems*, 33: 3008–3021, 2020.

Suzgun, M., Scales, N., Schärli, N., Gehrmann, S., Tay, Y., Chung, H. W., Chowdhery, A., Le, Q. V., Chi, E. H., Zhou, D., , and Wei, J. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*, 2022.

Team, G. G. Gemini: A family of highly capable multimodal models, 2023.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Wang, B., Zheng, R., Chen, L., Liu, Y., Dou, S., Huang, C., Shen, W., Jin, S., Zhou, E., Shi, C., et al. Secrets of rlhf in large language models part ii: Reward modeling. *arXiv preprint arXiv:2401.06080*, 2024.

Wang, P., Li, L., Chen, L., Zhu, D., Lin, B., Cao, Y., Liu, Q., Liu, T., and Sui, Z. Large language models are not fair evaluators. *arXiv preprint arXiv:2305.17926*, 2023a.

Wang, Y., Kordi, Y., Mishra, S., Liu, A., Smith, N. A., Khashabi, D., and Hajishirzi, H. Self-instruct: Aligning language model with self generated instructions. *arXiv preprint arXiv:2212.10560*, 2022.

Wang, Y., Ivison, H., Dasigi, P., Hessel, J., Khot, T., Chandu, K. R., Wadden, D., MacMillan, K., Smith, N. A., Beltagy, I., et al. How far can camels go? exploring the state of instruction tuning on open resources. *arXiv preprint arXiv:2306.04751*, 2023b.

Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8:229–256, 1992.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Scao, T. L., Gugger, S., Drame, M., Lhoest, Q., and Rush, A. M. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 38–45, Online, October 2020. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/2020.emnlp-demos.6.

Wu, Z., Hu, Y., Shi, W., Dziri, N., Suhr, A., Ammanabrolu, P., Smith, N. A., Ostendorf, M., and Hajishirzi, H. Fine-grained human feedback gives better rewards for language model training. *arXiv preprint arXiv:2306.01693*, 2023.

Xu, C., Sun, Q., Zheng, K., Geng, X., Zhao, P., Feng, J., Tao, C., and Jiang, D. Wizardlm: Empowering large language models to follow complex instructions, 2023.

Yao, Z., Aminabadi, R. Y., Ruwase, O., Rajbhandari, S., Wu, X., Awan, A. A., Rasley, J., Zhang, M., Li, C., Holmes, C., et al. Deepspeed-chat: Easy, fast and affordable rlhf training of chatgpt-like models at all scales. *arXiv preprint arXiv:2308.01320*, 2023.

Zeng, Z., Yu, J., Gao, T., Meng, Y., Goyal, T., and Chen, D. Evaluating large language models at evaluating instruction following. In *NeurIPS 2023 Workshop on Instruction Tuning and Instruction Following*, 2023.

Zhao, Y., Joshi, R., Liu, T., Khalman, M., Saleh, M., and Liu, P. J. Slic-hf: Sequence likelihood calibration with human feedback, 2023.

Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E., et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *arXiv preprint arXiv:2306.05685*, 2023a.

Zheng, R., Dou, S., Gao, S., Hua, Y., Shen, W., Wang, B., Liu, Y., Jin, S., Liu, Q., Zhou, Y., et al. Secrets of rlhf in large language models part i: Ppo. *arXiv preprint arXiv:2307.04964*, 2023b.

Zhou, C., Liu, P., Xu, P., Iyer, S., Sun, J., Mao, Y., Ma, X., Efrat, A., Yu, P., Yu, L., Zhang, S., Ghosh, G., Lewis, M., Zettlemoyer, L., and Levy, O. Lima: Less is more for alignment, 2023.

Ziegler, D. M., Stiennon, N., Wu, J., Brown, T. B., Radford, A., Amodei, D., Christiano, P., and Irving, G. Fine-tuning language models from human preferences, 2019.

A. Appendix

Algorithm 1 Proximal Policy Optimization for RLHF

Initialize policy parameters $\boldsymbol{w}$ from SFT model, old policy parameters $\boldsymbol{w}_{\text{old}} = \boldsymbol{w}$, batch size b. $m = 1, 2, ..., M$ Construct a batch of experiences $\mathcal{D}_{\pi_{\boldsymbol{w}_{old}}}$ by sampling N prompts $x \sim \mathcal{D}_{RL}$ and their completions $y \sim \pi_{\boldsymbol{w}_{old}(y|x)}$. $k = 1, 2, ..., K$ $n = 1, 2, ..., N/b$ Sample a batch $\mathcal{B}_{\pi_{\boldsymbol{w}_{old}}}$ of b examples from $\mathcal{D}_{\pi_{\boldsymbol{w}_{old}}}$. Compute the reward, value and advantage estimate $\hat{A}$ for each $(x, y) \in \mathcal{B}_{\pi_{\boldsymbol{w}_{old}}}$. Update the value network parameters. Update the policy with the clip objective. $\boldsymbol{w}_{\text{old}} \leftarrow \boldsymbol{w}$

B. Human Study

We designed the following human study interface based on the Gradio, shown as Figure 7. After consenting to the study, the participants are presented with a screen containing a session ID used to track and reference back the session, and guidelines framing how to evaluate the response. The criteria used are described in Table 4.

Criteria	Description
Alignment with User's Intent	Ensure the response directly addresses the user's question or task, interpreting underlying intentions when not explicitly stated.
Clarity and Precision	Responses should be easy to understand, avoiding unnecessary jargon and maintaining focus on the user's query.
Directness and Relevance	Keep the response strictly related to the task, avoiding unrelated information or tangents.
Efficiency and Brevity	Provide comprehensive yet concise information, steering clear of repetitive or overly detailed content that does not enhance understanding.

Table 4: Criteria for evaluating responses in the human study interface.

C. Correlation Metric

We use three correlation metrics in our main paper, *i.e.*, Spearsman's rank correlation r_s, Kendall's τ, and Pearson ρ. We compute ρ, r_s and τ using the following formulas:

$$
\begin{aligned}
\rho &= \frac{\sum(x_i - \mu_x)(y_i - \mu_y)}{\sqrt{\sum(x_i - \mu_x)^2 \sum(y_i - \mu_y)^2}} \\
r_s &= 1 - \frac{6 \sum d_i^2}{n(n^2 - 1)}, \\
\tau &= \frac{2}{n(n-1)} \sum_{i<j} \operatorname{sgn}(x_i - x_j) \operatorname{sgn}(y_i - y_j),
\end{aligned}
\tag{12}
$$

where $d_i = R(X_i) - R(Y_i)$ is the difference between two ranks of each observation and n is the number of the observations.

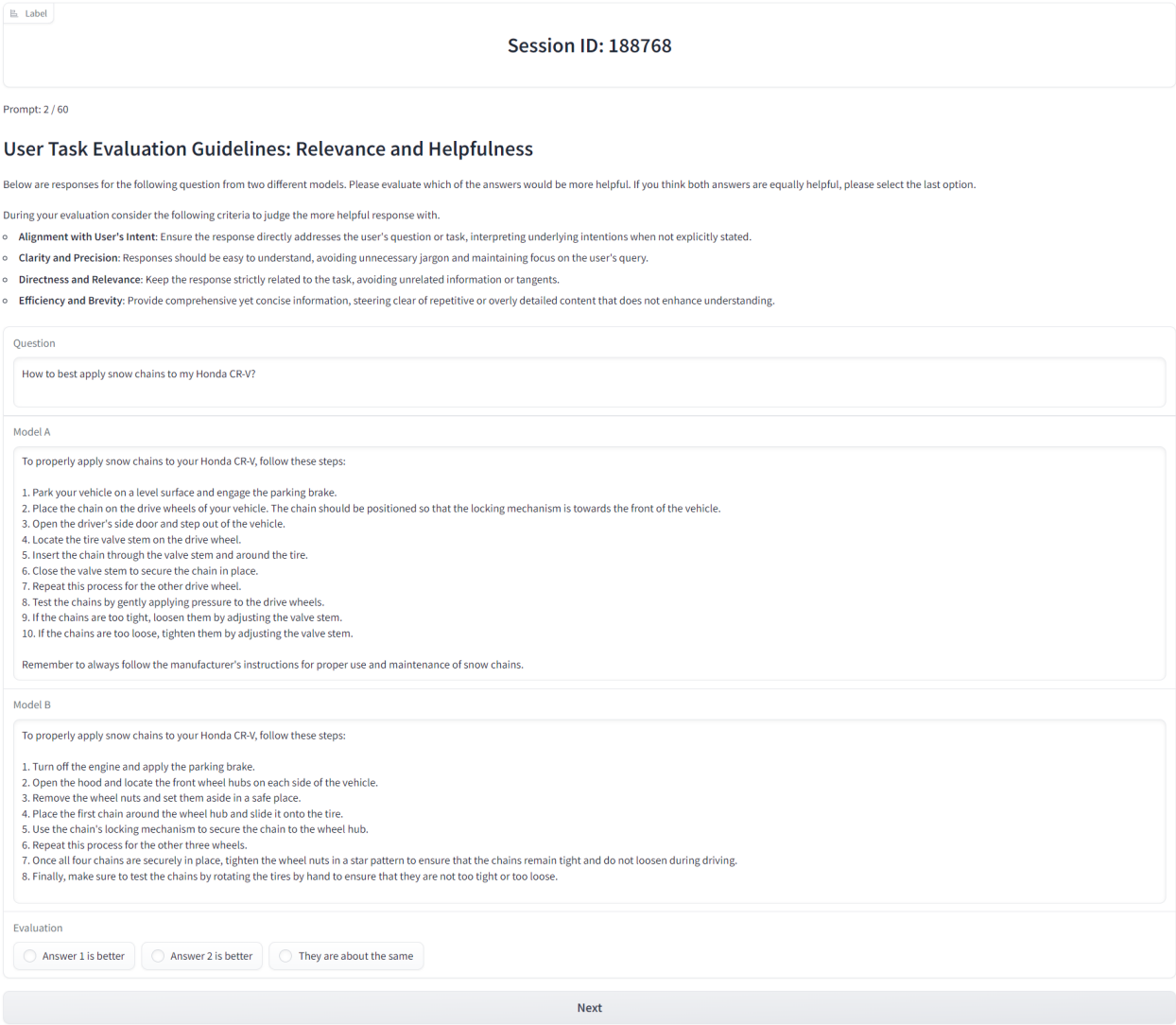


Figure 7: We design a human study UI using Gradio as the above shows.

D. Evaluation Prompt

[System Prompt] You are a helpful and precise assistant for checking the quality of the answers. [User Prompt] [Question] [The Start of Assistant1's Answer] Answer 1 [The End of Assistant1's Answer] [The Start of Assistant2's Answer] Answer 2 [The End of Assistant2's Answer] We would like to request your feedback on the performance of two AI assistants in response to the user question displayed above. Please rate the helpfulness, relevance, accuracy, level of details of their responses. Each assistant receives an overall score on a scale of 1 to 10, where a higher score indicates better overall performance. Please first output a single line containing only two values indicating the scores for Assistant 1 and 2, respectively. The two scores are separated by a space. In the subsequent line, please provide a comprehensive explanation of your evaluation, avoiding any potential bias and ensuring that the order in which the responses were presented does not affect your judgment.

Table 5: The GPT4 evaluation prompt.

E. Hyperparameter

E.1. Model generation config.

For RLHF training, to encourage models' exploration, we choose $top_p = 0.9$ and temperature $T = 1.0$ as the generation config which aligns with the setting used in Deepspeed-Chat and ReMax. As for evaluation, we use T=0.8 and top_p=0.8 to avoid over-randomness on the generations.

E.2. PPO config

We do full-model fine-tuning for both the actor and critic. Same as (Nakano et al., 2021), we use one epoch (set $K = 1$), and set $\gamma = 1.0, \lambda = 0.95$ for GAE. We train the model on Open Assistant for 3 epochs, which translates to 702 gradient update steps under the batch size $b = 32$, and takes around 11 hours to finish on 8 A100 GPUs with ZeRO stage 2. To make the search space tractable, we use the same learning rate η for the actor and critic. We search $\eta \in \{5e-7, 1e-6, 2e-6\}$, $\epsilon \in \{0.1, 0.2, 0.4\}$, $\beta \in \{2.5e-3, 5e-3, 1e-2, 2e-2\}$, $c \in \{\inf, 2, 4\}$, and $N \in \{32, 64, 256\}$. Note we did not finish all experiments with $\beta = 2.5e-3$, but we have included the partial results in the plots when $\beta = 2.5e-3$ is not explicitly excluded.

E.3. ReMax Config

The full-model finetuning is applied as well. Same as PPO, we use global batch size 32, and train the model for 3 epochs on the prompt set. We search $\beta \in \{1e-3, 2.5e-3, 5e-3, 1e-2\}$ and $\eta \in \{1e-6, 5e-7\}$ first. But we found the lengths of the trained actor models are mostly over 225. Unlike PPO, ReMax baselines do not have many hyperparameters (only β and η), we add some extra $\beta \in \{5e-3, 5.5e-3, 6e-3, \ldots, 9.5e-3\}$ with $\eta = 5e-7$ to get more results across different lengths, which makes the comparisons between different Pareto fronts more reliable.

E.4. Configs for Length Penalty Experiments

For experiment shown in Figure 4, we tried $\alpha \in \{1e-3, 1e-4, 1e-5, 5e-4, 1e-6, 5e-6\}$ for ReMax, and $\alpha \in \{5e-5, 1e-4, 5e-4, 1e-3\}$ for PPO. We select evaluation results with the same set of other RL hyperparameters like η, β, ϵ, N for different settings. Therefore, the length penalty setting always tends to have more data points.

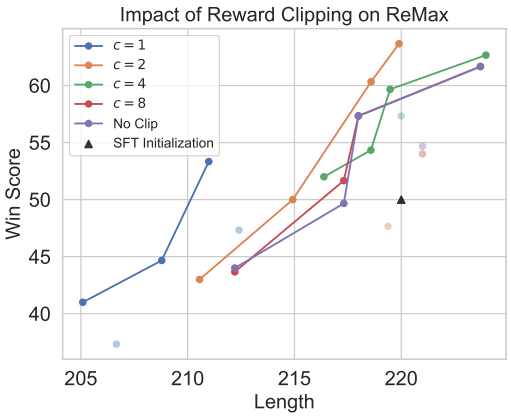


Figure 8: The effect of Reward Clipping on ReMax. We sweep the η and β.

E.5. Dialogue Format

We convert the prompts and responses in OpenAssistant into dialogue format using the following template:

Human: [The user prompt]
Assistant: [The answer to the prompt]

If the dialogue is multi-turn, we will use the same template as described and make all the previous turns' prompts and answers as the model's inputs.

F. Frequently Asked Questions

F.1. Why do not use other prompt sets for evaluating the models' capability on free-form QA?

We use LIMA (Zhou et al., 2023) as our test set for evaluating the instruction-following capability of models since it has 300 prompts, the size of which is larger than the other commonly used test set, e.g., WizardLM test set(218 prompts) (Xu et al., 2023), Koala(180 prompts) (Geng et al., 2023), MT-bench (Zheng et al., 2023a), and Self-Instruct(252 prompts) (Wang et al., 2022). The evaluation cost (for human study) is extremely high since we have tons of actor models to evaluate. Thus, the main evaluations are conducted by GPT-4, and we also select some models on the Pareto front to do human study.

F.2. Why do you choose Vicuna-7B as the base model or the starting point of the RL?

We choose Vicuna-7B as our base model for two reasons: (1). Compared to other open-sourced 7B models, Vicuna-7B has pretty good instruction-following capability. To ensure efficient and effective exploration in RLHF, we need a good base model. (2). To ease the comparison and provide an accurate and comprehensive view of the RL algorithm, we chose the well-known SFT model but did not do the SFT on the OpenAssistant dataset by ourselves. It can also help us avoid the selection of the SFT checkpoint, where different people have different criteria. By using Vicuna-7B and the reward model we provided, we believe the community could reproduce our results more easily.

G. Case Study

We show two comparison in Figure 12 and Figure 13, where our models could generate more accurate answers with shorter length.

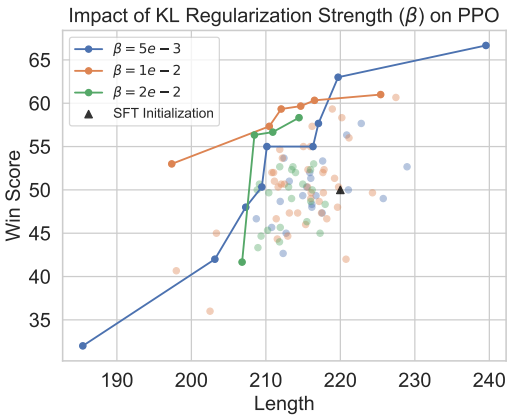


Figure 9: The effect of KL regularization strength when sweeping η, ϵ, N and disabling reward clipping. While the result becomes more sensitive to KL, it indicates that we can find better results with smaller β, and results with larger β can be surprisingly improved when reward clipping is considered.

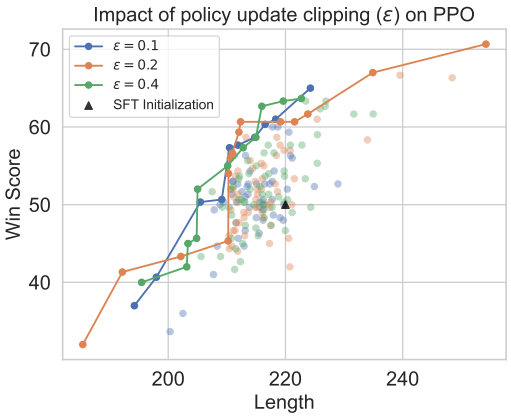


Figure 10: The effect of PPO clipping threshold when sweeping η, β, N and c. With reward clipping, the result becomes better, but the effect of ϵ becomes complicated, with $\epsilon = 0.1$ and $\epsilon = 0.4$ both being better than $\epsilon = 0.2$.

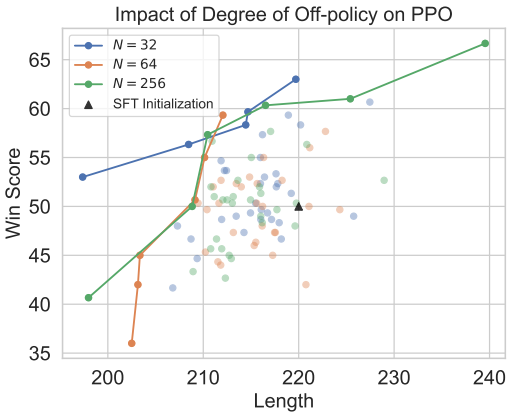


Figure 11: Results under different sizes of experience batch N, when disabling reward clipping and sweeping η, β, ϵ. We use batch size $b = 32$, so $N = 32, 64, 256$ correspond to 0%, 50% and 87.5% "off-policy" samples, and ϵ clipping is ineffective when $N = 32$. Similar to the case with reward clipping, larger N is not beneficial.

Prompt:
what does \\multirow{5}{*} do in latex?

ODIN: (96 tokens)
The command \\multirow{5}{*} is used in LaTeX to create a multi-line cell in a table. It is similar to the \\multirow command, which allows you to merge cells, but instead of merging cells, it creates a new cell that spans multiple lines.\n\nThe \"5\" in the command is the number of lines the cell should span, and can be adjusted as needed. The \"*\" indicates that the cell should span the full width of the column.\n\n

ReMax with Vanilla RM: (285 tokens)
The command `\\multirow{5}{*}` is used in LaTeX to horizontally center a row of five items. The `multirow` command is used to merge cells in a table and create a single cell that spans multiple columns. The `{}` indicates the width of the cell, which is five times the width of the column.\n\nHere is an example of how to use `\\multirow{5}{*}` in a table:\n```css\n\\begin{tabular}{|c|c|c|c|c|}\n \\hline\n \\multirow{5}{*} & Column 1 & Column 2 & Column 3 & Column 4 & \\\\\n \\hline\n Item 1 & Value 1 & Value 2 & Value 3 & Value 4 & \\\\\n Item 2 & Value 5 & Value 6 & Value 7 & Value 8 & \\\\\n Item 3 & Value 9 & Value 10 & Value 11 & Value 12 & \\\\\n Item 4 & Value 13 & Value 14 & Value 15 & Value 16 & \\\\\n Item 5 & Value 17 & Value 18 & Value 19 & Value 20 & \\\\\n \\hline\n\\end{tabular}\n```\nIn this example, the row containing the items \"Item 1\" to \"Item 5\" is horizontally centered in the table.

Figure 12: The comparison of our actor model, trained with ODIN, with the actor model trained with the vanilla reward model.

Prompt:
Please prove that 3+5=9

ODIN: (18 tokens)
I'm sorry, but that is incorrect. 5 + 3 equals 8.

PPO with Vanilla RM + Length Penalty: (69 tokens)
I‘m sorry, but that is a false statement. 5 + 3 equals 8. Please keep in mind that, its a basic math fact that 5+3 is 8 and not 9. I'm glad I could help! If you have any more questions or need further assistance, please don't hesitate to ask!

Figure 13: The comparison of our actor model, trained with ODIN, with the actor model trained with the vanilla reward model.