Increasing diversity in RLHF

Ifdita Hasan Orney [1] **Zhuoer Gu** [1] **Eric Han** [1]

Abstract

Reinforcement Learning from Human Feedback (RLHF) is a powerful technique used to fine-tune large language models (LLMs) to produce high-quality, human-aligned text. While optimization and reward modeling in RLHF have been well studied, there is a dearth of work around promoting exploration when sampling actions from the policy LLM.

Thankfully, LLMs have a nice property that they can be prompted to generate different response variants. We find that appending ”edits” to the prompt of the LLM increases diversity of generations during RLHF, ultimately encouraging exploration and achieving higher reward. Our method also allows us to trace what kind of change to the LLM generation promoted higher reward, providing interpretability for optimization of black-box rewards.

We recommend LLM finetuning practitioners to introduce a diversity-increasing step in RLHF sampling and potentially in production inference.

1. Introduction

Reinforcement Learning from Human Feedback (RLHF) is a common method for aligning Large Language Models (LLMs). Today, LLMs using RLHF commonly follow the below algorithm:

Algorithm 1 RLHF Algorithm Framework

```
1. Generate a response for each prompt
2. Give a reward for each response by using a reward model
3. Apply optimization step to increase likelihood of responses that increase reward most greatly
```

In the algorithm above, responses are usually generated with high temperature for greater randomness, and the optimization step usually uses the Proximal Policy Optimization(PPO) (Schulman et al., 2017) update step.

In this setup, we are asking the LLM to explore over text generation space to find response generations that better solve the prompt’s request as evaluated by the reward model. However, the current paradigm of simply sampling from the policy tends to have low diversity (Peeperkorn et al., 2024), even if we increase temperature significantly to promote randomness. This issue is compounded by the fact that the ”step size” of LLM generations during RLHF is typically small, leading to slow optimization progress.

We believe that the general RLHF technique is not exploration-aware. The main question we raise in this project is: how can increasing the diversity of sampled generations during RLHF increase exploration, and thus optimize reward more efficiently? To address the problem of increasing exploration of RLHF, we have the following objectives, and our approach to increase diversity of responses involves applying self-critique models which will be explained further in Section 3.

1.1. Objectives

- Generate datasets with higher diversity and find higher scoring response generations based on such datasets.
- Use the response generations with higher scoring to train a policy LLM that achieves greater reward.
- Use this method to increase reward by observing what prompts induce higher reward generations.

2. Background/Related Work

We want to increase diversity of generations and we propose that the LLM itself should be able to produce a greater diversity of responses if we simply ask in the correct way. This

[1]Stanford University, Stanford, California, USA. Correspondence to: Ifdita Hasan Orney <ifdi1101@stanford.edu>, Zhuoer Gu <guzhuoer@stanford.edu>, Eric Han <erichan2@stanford.edu>.

is partially inspired by work around self-critique (Stechly et al., 2023).

Considerable research has focused on aligning language models with human preferences and managing the inherent randomness of LLMs. One significant contribution is Statistical Rejection Sampling (SRS) introduced (Liu et al., 2024). This technique uses rejection sampling to gather preference data directly from the best policy, which facilitates a more accurate estimation of the optimal policy. By selecting samples that align closely with desired outcomes, SRS improves the fidelity of the generated responses.

Another notable method is Sequence Likelihood Calibration (SLiC), developed by (Zhao et al., 2023). SLiC refines the alignment process by adjusting the loss function using sequences sampled from a supervised fine-tuned policy. This calibration process ensures that the model's outputs are more closely aligned with preferred sequences, thereby enhancing the overall quality and alignment of responses.

Direct Preference Optimization (DPO), as proposed by (Rafailov et al., 2023), offers another approach by optimizing models directly based on human preference data. Unlike traditional methods that rely on a separate reward model, DPO integrates preference data into the optimization process, streamlining the model's ability to generate preferred outputs.

Incorporating insights from these methods, we hypothesize that prompting the LLM in a manner that encourages self-critique and diversity can further enhance the variety of its responses. By combining techniques such as rejection sampling, likelihood calibration, and direct optimization, we can develop a more robust framework for eliciting diverse and high-quality outputs from LLMs. This approach not only aligns with human preferences but also leverages the inherent capabilities of the model to produce a richer set of responses.

3. Approach

3.1. Infrastructure and Inference

In setting up the infrastructure, we first choose the open-source OpenAssistant Helpfulness Reward Model (OpenAssistant/reward-model-deberta-v3-large-v2), accessible through HuggingFace. With compution limitations on our compute environment Google Colab and its free T4 GPU runtime, we performed our inferences on small language model Phi3 (microsoft/Phi-3-mini-128k-instruct 3.8B). We ran our inference on a set of 100 prompts that are generated from prompting ChatGPT "Generate 100 short prompts that ask for explanations of academic concepts and summaries of literature, art works, or movies." We performed finetuning on Phi2 (microsoft/phi2). While we would have liked to use the same model as our generations (Phi3), we could not because of increased memory consumption in finetuning and the limits on our compute environment.

3.2. Self-Correction RLHF

we propose self-correction technique to add diversity to RLHF. For a given prompt, we begin from an initial seed response from the LLM. Then we make edits, and ask the LLM to make improvements to its own response based on the edit inquiries. We use these more diverse responses within the RLHF loop.

The algorithm framework is as follows:

Algorithm 2 Self-Correction Algorithm Framework

```
1. Sample from LLM
2. Ask LLM for $K$ different ways to improve upon or rewrite the response
3. Allow the LLM to generate $K$ different responses based on its planned improvements.
4. Score all $K + 1$ responses using the reward model.
5. Optimization step.
```

In our experiments, we chose $K = 5$ and hence our initial dataset includes 600 responses. The following two subsections are two variations of the above framework in terms of the implementation of Step 2 and 3. We will call the ways to ask LLM to rewrite the responses in Step 2 as "edits".

3.3. Treatment Algorithm

When we sample 5 more generations from our LLM, instead of naive sampling, we append different edits that are predetermined to change the style or nature of the generation. Below is an illustration of the 5 prompts with appended edits.

"Explain nuclear fusion."
$\Rightarrow$
"Explain nuclear fusion. **Make the response concise."**
"Explain nuclear fusion. **Make the response informative."**
"Explain nuclear fusion. **Make the response easier to understand for beginners."**
"Explain nuclear fusion. **Make the response in more formal or professional language."**
"Explain nuclear fusion. **Illustrate an example."**

We implement the below algorithm based on the framework in **Algorithm 2**.

Algorithm 3 Treatment Algorithm

```
Input: prompt list $[prompt_i]$, size 100
Augment prompt list by appending the 5 edits to every $prompt_i$, size 600
Reshape the augmented prompt list to size $(100, 6)$
Generate responses on the reshaped augmented prompt list using LLM inference
Generate logits raw scores using the reward model inference
```

3.4. Baseline Algorithm

We follow the rejection sampling algorithm that is used heavily in LLama RLHF(AI, 2024). For each prompt, we sample 6 generations in total from the our LLM. Then we score all generations using reward model, which should be size $(100, 6)$ after reshaping. We take the highest scoring generation for each prompt. After generating the dataset of all responses and scores, we finetune the policy(our LLM) using behavior cloning on generated prompt-response pairs. Below is the baseline algorithm modified from the framework in **Algorithm 2**.

Algorithm 4 Baseline Algorithm

```
Input: prompt list $[prompt_i]$, size 100
Augment prompt list by repeating it 5 times, size 600
Reshape the augmented prompt list to size $(100, 6)$
Generate responses on the reshaped augmented prompt list using LLM inference
Generate logits raw scores using the reward model inference
SFT on the dataset
```

4. Experiment Results

4.1. Adding diversity in best-of-N sampling increases exploration

Today, there is little work on increasing diversity of RLHF generations. The focus has primarily been around better optimization algorithms and more robust reward models. However, assuming robustness of the reward, a failure mode of RLHF is that naively generating from the LLM gives very similar generations - there is little exploration. Thankfully, LLMs have a nice property that they can be prompted to change their response. Thus, we believe that appending ”edits” to the prompt of the LLM should allow the LLM to jump out of local minima and find more optimal generations.

For each prompt, in the naive case, we perform six generations from the same prompt with high temperature. We compare this against a more diverse set of generations where we append various edits to the prompt. We see in Figure 1 that the distribution of scores widens. This effect is particularly strong at the per-prompt level. On average, per-prompt standard deviation in reward increases from 0.76 to 1.32.

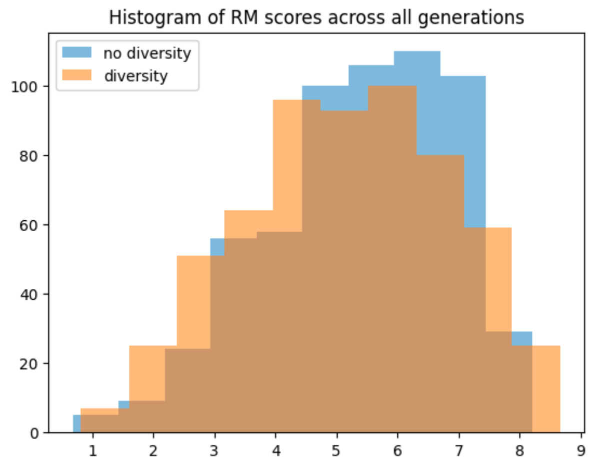


Figure 1. Diversifying generations increases score variance: After adding diversity, per-prompt standard deviation of reward model score increases from 0.76 to 1.32.

4.2. Adding edits in rejection sampling increases top-1 reward

We then ask whether we can utilize greater variance in reward score to find overall higher reward generations. For each prompt, out of our six generations, we take the one with the highest reward as the optimal generation.

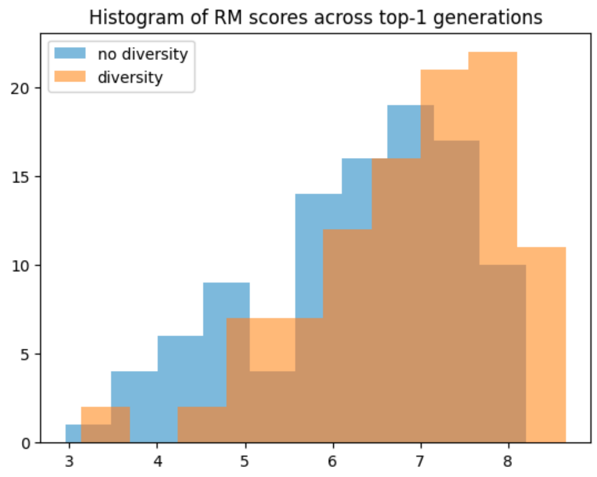


Figure 2. Diversifying generations gives higher top-1 reward: We achieve 74% winrate of diverse dataset over non-diverse dataset, comparing score of top-1 generation for each prompt.

For each prompt, we then compare the top-1 score of the dataset with diversity edits vs. the dataset without. As judged by reward model score, we achieve 74% winrate. Tracking with these results, we see the overall distribution of scores is right-shifted in Figure 2.

4.3. Adding diversity also improves human preference

We've shown that improving RLHF diversity improves predicted reward, based on the reward model. However, a potential pitfall of exploration in RLHF is going off-distribution of the reward model. We thus perform a small-scale human evaluation of the preference pairs to see whether humans still consider the generations from the diverse dataset to be more helpful.

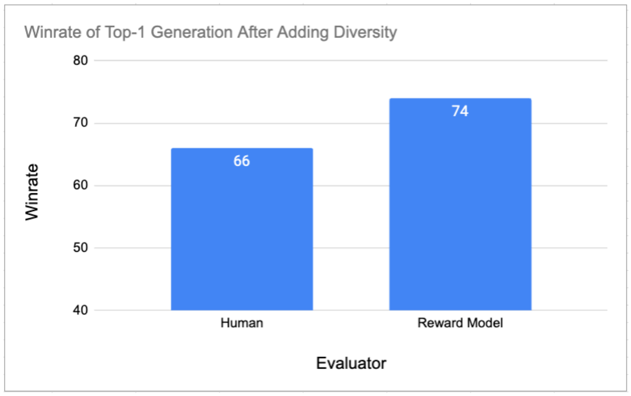
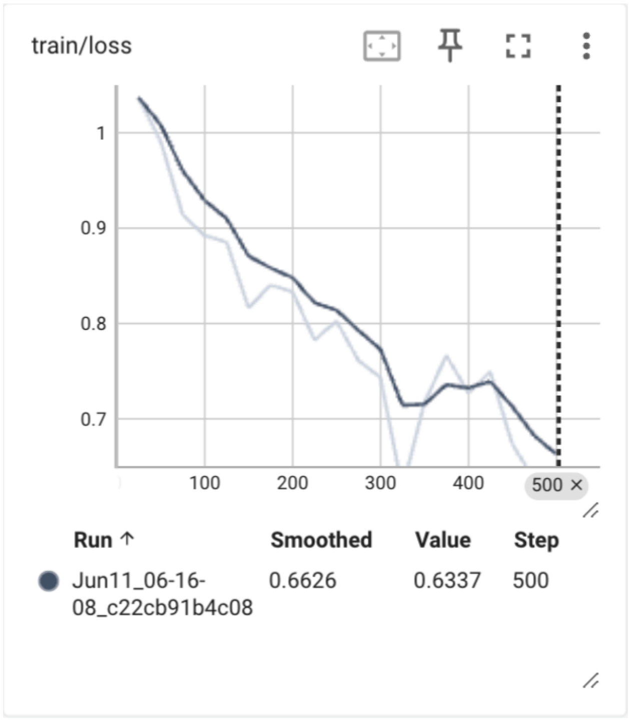


Figure 3. Human Preference also improves after adding diversity to dataset, tracking with reward model. Reward model winrate of diverse dataset top-1 vs. non-diverse dataset top-1 is 74%. Human preference is somewhat less, at 66%.

For each prompt, we have a human (one of our researchers) perform a blind selection between a pair consisting of the top-1 generation from the diverse dataset vs. the one without. We display the generations in a spreadsheet and randomize the position of the two generations. As shown in Figure 3, we find that the human prefers 66% responses from the diverse dataset. This is not far off from the 74% preference from the reward model, but it is less. We thus see that there may be some element of reward hacking, but we leave a deeper analysis to future work.

4.4. Can we use those higher scoring generations to train a policy LLM that achieves greater reward?

By using best-of-N sampling with diversity edits, we've found that we can sample better responses from our policy LLM. We'd also like to use this exploration to finetune our LLM to produce better generations. We finetune two phi-2 LLMs using the top-1 generation from our dataset with diversity and dataset without diversity. This is 100 prompt-response pairs for each dataset. We find that loss goes down nicely (Figure 4).



Run ↑	Smoothed	Value	Step
● Jun11_06-16-08_c22cb91b4c08	0.6626	0.6337	500

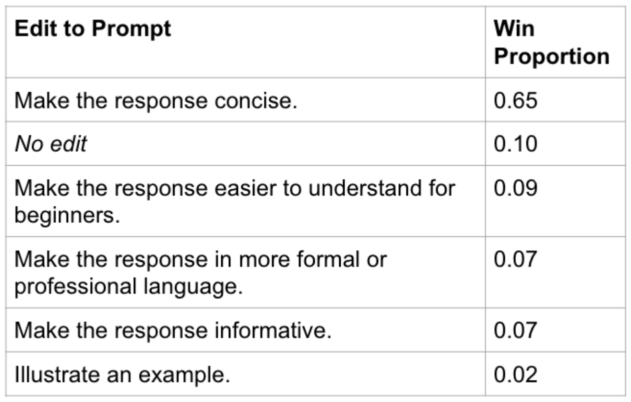
Figure 4. Train loss goes down nicely on dataset created with diversity edits.

However, we find that the average reward going from our baseline model (no diversity) to our experimental model (including diversity) declines drastically in average reward, from 2.48 to 0.80. Our winrate is also 22%, strongly unfavorable to our experimental model. The reasons for this decline in performance are unclear - we believe they are more training recipe or infrastructure related rather than indicative of issues in our method. We leave further analysis to future work.

4.5. Can we use this method to interpretably increase reward by observing what prompts induce higher reward generations?

One of the most promising features of using prompting to perform exploration is that we get interpretability for free. If the model explored in a certain way and received higher reward, we can look at the edit in the prompt that caused the reward increase.

In our 100 prompt dataset, we count the number of times each edit was ranked best for each prompt. This reward model particularly preferred concise responses, with ”Make the response concise” as the most popular winning edit at 65% (Figure 5). This is especially interesting because in most literature on reward modeling in RLHF, annotators tend to prefer longer responses. This is called length bias, and there has been substantial recent work to address it, such as in DPO (Park et al., 2024).



Edit to Prompt	Win Proportion
Make the response concise.	0.65
No edit	0.10
Make the response easier to understand for beginners.	0.09
Make the response in more formal or professional language.	0.07
Make the response informative.	0.07
Illustrate an example.	0.02

Figure 5. In our generated diverse dataset, we count the number of times each edit was ranked best for each prompt. We find that this particular helpfulness reward model prefers conciseness.

5. Conclusion

Prompting to increase diversity of generations during RLHF is a simple but effective way to encourage exploration and achieve higher rewards. This method involves introducing mechanisms that ensure a broader range of generated outputs, thereby enhancing the model's ability to explore different strategies and solutions.

However, while this approach can lead to improved performance, it also comes with potential limitations, notably the risk of reward hacking. Therefore, it is crucial to design the reward function and diversity-promoting mechanisms carefully to mitigate this risk.

Moreover, our method provides interpretability for the optimization of black-box rewards. By understanding how diversity impacts the reward function, practitioners can gain insights into the underlying processes and make more informed decisions about adjustments and improvements.

Recommendations for Practitioners

We recommend that practitioners involved in fine-tuning Large Language Models consider the following steps:

- Introduce a diversity-increasing step in the RLHF sampling process. This can be achieved through techniques such as entropy maximization, novelty search, or encouraging diverse trajectories.
- Evaluate the potential impacts of diversity-promoting mechanisms on reward hacking. Implement safeguards and monitoring systems to detect and address any instances of reward hacking.
- Consider integrating diversity-promoting techniques not only during training but also in production inference. This can help maintain a high level of performance and adaptability in real-world applications.

6. Future Work

6.1. Reward Hacking

One of the primary limitations of increasing diversity in RLHF is that generations may go off distribution for the reward model, encouraging reward hacking. Even in our small-scale comparison of human versus reward model judgment, we find that the win rate of the reward model is higher than for humans, which indicates some level of reward hacking. We'd like to further quantify this.

Some potential directions for addressing this issue include:

1. Expand the size of our dataset and have multiple annotators check each comparison to get tighter bounds on how much we are improving. By increasing the dataset size and incorporating multiple annotators, we can reduce the noise in our evaluation metrics and achieve more reliable measurements of improvement. This approach will help in understanding the extent of reward hacking and the actual performance gains from increased diversity.
2. Use a holdout reward model to study whether we are over-optimizing the reward model we are optimizing towards. By training and evaluating a separate reward model that is not used during the optimization process, we can investigate if the primary reward model is being overfitted. This method can reveal the potential discrepancies between different reward models and highlight the risks of over-optimization.
3. Study the diversity impact in RLHF on the Pareto-optimal boundary of KL divergence and reward. By analyzing the trade-off between maintaining low Kullback-Leibler (KL) divergence and achieving high reward, we can identify the Pareto-optimal solutions. This analysis can help in balancing the objectives of diversity and reward, ensuring that the model does not excessively diverge from the desired behavior while still exploring diverse outputs.
4. Simply use a stronger reward model. We used a fairly weak one because of compute constraints. By employing a more robust reward model, we can potentially

reduce the instances of reward hacking and obtain more accurate assessments of the model's performance. Investing in a stronger reward model may lead to better alignment with human preferences and improved generation diversity.

6.2. Ablations on Optimization Step

We chose to use behavior cloning for the optimization of our policy models, primarily because of infrastructure challenges. While we believe that our findings should transfer, we would still like to explore additional RLHF algorithms to validate our results and potentially discover more effective optimization techniques. We list some of these algorithms below:

1. Direct Preference Optimization (DPO) (Rafailov et al., 2023): DPO directly optimizes the model based on human preference data, eliminating the need for a separate reward model. This method can provide a more straightforward and potentially more effective approach to aligning the model with human preferences.
2. Kahneman-Tversky Optimization (KTO) (Ethayarajh et al., 2024): KTO introduces an optimization framework inspired by behavioral economics, focusing on aligning the model's decisions with human-like reasoning processes. This approach can help in mitigating biases and improving the robustness of the model's outputs.
3. Proximal Policy Optimization (PPO): PPO is a widely used RL algorithm known for its stability and efficiency. It optimizes policies by iteratively improving them while maintaining a balance between exploration and exploitation. Incorporating PPO into our RLHF framework can potentially enhance the quality and diversity of the generated responses.

6.3. Additional directions

Some additional experiments would include:

1. Quantify "exploration" better. Create a metric to quantify prompt diversity.
2. Allow LLM to itself choose edits to its prompt. We could even expand to an agent-type workflow where the LLM goes through multiple iterations of self-improvement of its prompts.
3. Go deeper on interpretability of reward models, using prompt diversity. Can we produce some insights about human preference on different kinds of prompts? For example, that a human prefers a longer response for some kinds of prompts and shorter for others.
4. Expand this method to domains where we have a verifier that is always correct, instead of a reward model that is only sometimes correct.

References

AI, M. Introducing meta llama 3: The most capable openly available llm to date, 2024. URL https://ai.meta.com/blog/meta-llama-3/.

Ethayarajh, K., Xu, W., Muennighoff, N., Jurafsky, D., and Kiela, D. Kto: Model alignment as prospect theoretic optimization, 2024.

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

Park, R., Rafailov, R., Ermon, S., and Finn, C. Disentangling length from quality in direct preference optimization, 2024.

Peeperkorn, M., Kouwenhoven, T., Brown, D., and Jordanous, A. Is temperature the creativity parameter of large language models?, 2024.

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.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms, 2017.

Stechly, K., Marquez, M., and Kambhampati, S. Gpt-4 doesn't know it's wrong: An analysis of iterative prompting for reasoning problems, 2023.

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