Extended Abstract

Motivation Significant advancements have been made in training large language models (LLMs) for question-answer tasks. However, using a single model results in several limitations. Often times these models struggle to produce diverse, high-quality responses. Constraints such as limited training data and training resources hinder the ability to improve a single model's performance. Techniques such as direct preference optimization (DPO) have been found to be effective in aligning model responses to preferred responses. However, even when using this approach, training can degrade overtime due to reductions in the likelihood of preferred responses compared to rejected responses.

Method Our project addresses these two issues by both implementing a modified DPO algorithm and a multi-agent framework. We explore the usage of the DPO Positive algorithm (DPOP) to improve model performance on mathematical reasoning tasks Pal et al. (2024). This algorithm modifies the loss function of the DPO algorithm by introducing a penalty term to ensure the likelihood of preferred responses increases during training. We add the penalty term $\lambda \cdot \max(0, \log \frac{\pi_{\text{ref}}(y_w|x)}{\pi_\theta(y_w|x)}$ to the DPO loss function. This ensures that likelihood of the preferred responses remains high in the newly trained model relative to the reference model.

To improve the quality and diversity of responses for instruction reasoning, we implement a 3-agent approach. We pre-train a response agent using supervised finetuning to generate responses to prompts. We additionally pre-train a feedback agent on prompt-response-feedback samples. The prompt-response pairs from the response agent are then fed to a feedback agent. Our feedback agent generates feedback which is concatenated to the initial prompt and returned to the response agent to generate a revised response. We finally train a judge agent using DPO to select the best quality response. The judge agent receives the initial and the revised response and selects the best one which is used as the final response.

Implementation For the math reasoning task, we perform supervised finetuning (SFT) on the Qwen 2.5-0.5B Base model using the Warmup Dataset. This is our initial baseline model. We sample responses from this finetuned model to create a preference dataset and train a DPO-optimized baseline model. We additionally utilize RLOO to train a final baseline model on the Countdown dataset. We implement the DPOP extension using our newly created preference dataset. For the instruction tuning task, we perform SFT on the Smoltalk dataset and DPO on the Ultrafeedback dataset. We additionally, train the base model using RLOO on the Ultrafeedback dataset using a Bradley-Terry Reward model. For the multi-agent extension, we pre-train the agents using the Smoltalk dataset and the unprocessed Ultrafeedback dataset.

Results For the math reasoning task, we do not see significant differences in the performance for the DPOP-trained model compared to the DPO-trained model. The best performing model is the RLOO baseline model that obtains a score of 0.72 on the validation set, a score of 0.78 on the first leaderboard test set, and a score of 0.36 on the second leaderboard test set. For instruction reasoning, we find that the multi-agent framework improves model performance and the quality of responses compared to all baseline models, resulting in a win-rate of 0.83 against the reference model on the first leaderboard and win-rate of 0.15 on the second leaderboard.

Discussion Introducing feedback significantly improves the quality of responses without needing to increase model size or obtain more data. However, not all feedback is equally informative and additional work such as further pretraining or introducing additional agents can improve feedback quality and diversity. DPOP does not significantly improve model performance on math reasoning. Performance may be dependent on the amount of training time which should be explored in future work.

Conclusion The multi-agent framework is effective for producing higher quality responses for instruction reasoning tasks. The benefits for DPOP are not seen within our limited training times. Its efficacy should be examined in future studies.

Many Minds, One Goal: Enhancing DPO in a Multi-Agent Framework

Yasmine Mabene
Department of Computer Science
Stanford University
ymabene@stanford.edu

Abstract

We implement a multi-agent framework to produce high-quality responses for instruction tuning tasks. We find that this approach improves model performance without the need to increase training data or to implement significant changes in model size or training time. The multi-agent model achieves leaderboard win rates of 0.83 and 0.15, 5% increases from the baseline SFT model. For math reasoning, we assess the impact of introducing a penalty term to the DPO loss function to mitigate a failure mode in the likelihood of preferred responses over time. This modified model achieves a score of 0.71 and 0.36 on the first and second leaderboards. These scores are commesurate with the DPO and RLOO baselines. Although we did not find significant changes in DPO performance with the additional term, future work is needed to assess its benefit with longer training durations.

1 Introduction

Training large language models (LLMs) using direct preference optimization (DPO) has been a powerful tool for improving responses to natural language prompts when preference data is available Rafailov et al. (2023). However, the performance of a single model trained for a task is often constrained by factors such as limited data, sparse rewards, and lack of diversity compared to the variability in typical human language Subramaniam et al. (2025). To address these limitations, we draw on the work developed by Subramaniam et al. (2025) and adopt a multi-agent framework in which multiple agents are trained to generate responses to natural language prompts, provide feedback, and select the best response. We believe that this framework, which is both iterative and modular, can improve the quality and diversity of responses without modifying the amount of training data. This is valuable given the challenges and costs associated with data collection. Additionally, promoting diversity is essential for mitigating bias and ensuring more representative outputs from LLMs. Furthermore, we aim to address existing limitations within the DPO method by integrating modifications into the algorithm including the DPO-Positive algorithm (DPO), which ensures the model does not reduce the likelihood for preferred examples over time. We believe that these modifications in conjunction with the multi-agent framework can improve model performance and make training more efficient.

2 Related Work

Many researchers have turned to multi-agent frameworks as an approach for addressing existing constraints from training LLMs. Although training a single LLM can be powerful, performance often plateaus with limited data and ability to self-improve Subramaniam et al. (2025). Researchers such as Subramaniam et al. (2025), Du et al. (2023), and Chan et al. (2023) have developed multi-agent approaches that allow for LLMs to improve eachother's responses. In Subramaniam et al. (2025), Subramaniam and co-authors introduce a framework where actor agents respond to a prompt and debate with one another, summarizing the responses of other LLMs. A critic LLM then votes on the improved responses to select the best one. This approach, which utilizes specialized and diverse agents, outperformed baseline frameworks using a single LLM. Additionally, the incorporation of the critic improved model performance. Although this approach performs well, it is computationally expensive. It is also unclear whether or not there is benefit in the actor agents performing the summarization as opposed to delegating this task to an independent agent. Furthermore, all of the actor agents undergo identical training. In contrast, in Chan et al. (2023) all of the agents have their own unique prompts and roles. This level of diversification increases model performance. However, this approach does not incorporate a critic agent, and the final output is generated through consensus from the agents, a method that does not guarantee convergence. We aim to build on both of these frameworks by exploring the usage of separate specialized agents for response generation, feedback, and critique. Additionally, we would like to incorporate diversity into our agents by employing modified training objectives.

Within our instruction tuning and math reasoning tasks, we aim to use DPO, a successful method for aligning model behavior with human preferences. Despite the benefits of DPO, researchers have uncovered a limitation of training technique: models trained with the objective have reductions in the likelihood of preferred responses over time Pal et al. (2024). This reduction is more pronounced in datasets where there are small edit distances between the preferred and rejected responses. This occurs because the DPO loss function is constructed using a ratio of log likelihoods. Thus, it is possible to decrease the objective even if the likelihood for preferred responses decreases so long as the likelihood for rejected responses decreases as well. This means that the DPO-trained model may have lower likelihoods for preferred responses than the initial reference model. To overcome this issue, Pal and co-authors devised DPOP, an alternative to DPO that does not share this failure mode and achieves state of the art performance on academic and chat benchmarks. Due to limited computation, the authors were unable to evaluate if this improvement holds across models of varying sizes. In our work, we will apply the DPOP objective to a smaller-sized model.

3 Method

Our multi-agent framework consists of multiple specialized agents that are trained to generate responses, provide feedback, and evaluate natural language prompts. The response agent receives and prompt and is trained to output a response. Both the prompt and response are concatenated and fed to the feedback agent which outputs feedback. This feedback is combined with the initial prompt and a system prompt that instructs the model to adjust its response based off of the feedback and is given as input to the response agent. The response agent outputs a newly revised response. Both the initial prompt, the initial response, and the revised response are given as input to the judge agent to select the final response. This inference framework is depicted in Figure 1 and 2 below.

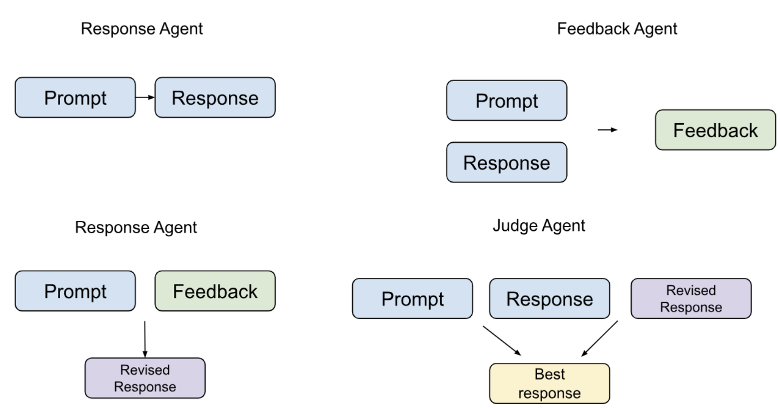


Figure 1: Multi-Agent Inference Architecture

Initial Prompt:
Write a function in Python to add two numbers together.
Initial Response:

```
def add_numbers(a, b):
    return a - b
```

Prompt and Feedback:
Write a function in Python to add two numbers together. You previously responded to this prompt and received the following feedback: "The function is well formatted, but is performing subtraction instead of addition." Please generate an improved response to the original prompt, taking the feedback into account.
Revised Response:

```
def add_numbers(a, b):
    return a + b
```

Figure 2: Sample Prompts and Responses (Response Agent)

We pretrain the response agent using supervised finetuning (SFT) on prompt-response pairs from the Smoltalk and Ultrafeedback datasets. We mask the loss for the prompts and only consider the loss over response tokens. To train the feedback agent, we utilize the unprocessed Ultrafeedback dataset which consists of feedback responses generated from GPT-4 on the helpfulness of the output. We train the agent using SFT on prompt-response-feedback samples, only using the loss over the feedback tokens. The judge agent is trained using DPO on the Ultrafeedback dataset. During inference, the judge agent determines the best response based on which response has the highest average log-probability. What differentiates our approach from the multi-agent frameworks previously described is the separation of response generation, feedback, and critiques into separate agents as well as variable training objectives used for the agents.

In addition to the multi-agent framework, we experiment with modifications to the DPO algorithm for the math reasoning task.

We implement the DPOP algorithm where we modify the DPO loss function to the equation below:

$$\mathcal{L}_{\text{DPOP}}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}}\Bigg[\log\sigma\Bigg(\beta\left(\log\frac{\pi_\theta(y_w \mid x)}{\pi_{\text{ref}}(y_w \mid x)} - \log\frac{\pi_\theta(y_l \mid x)}{\pi_{\text{ref}}(y_l \mid x)}\right) \\ - \lambda \cdot \max\left(0, \log\frac{\pi_{\text{ref}}(y_w \mid x)}{\pi_\theta(y_w \mid x)}\right)\Bigg)\Bigg]$$

This new formula adds a penalty term to the DPO loss function that ensures that the model needs to improve to likelihood of preferred responses to decrease the loss. We will assess the impact of this penalty term using a smaller-scaled model compared to the original analysis.

In order to evaluate the instruction tuning task, we use the win rate comparing the ratio of preferred responses belonging to the trained model and the reference model. To compute the win rate, we use the Nemotron 70-B reward model. For the validation dataset, we use a baseline SFT model as the reference. For the test dataset, we use the Qwen2.5-0.5B Instruct model. We will also report the RougeL-Sum score between the generated responses of different models and the preferred responses to assess how aligned the outputs are Lin (2004). We additionally report the proportion of revised responses that are ranked higher than the initial responses, a measure of the feedback quality. For the mathematical reasoning task, we use the leaderboard score which incorporates the accuracy and formatting performance of the model. We also examine performance stratified by different prompt types.

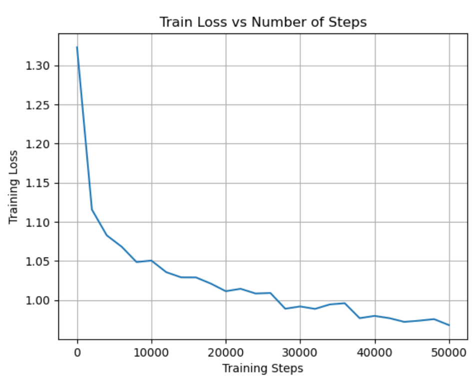
We implement several baselines to compare our model performance. For instruction tuning, we perform supervised finetuning on the Qwen2.5-0.5B Base model. We additionally train this finetuned model using DPO. We also train the SFT model with Online Policy Gradient (RLOO) using the Bradley Terry Reward Model for preference datasets. To do so, we add an MLP head to the SFT model and train it using the Bradley-Terry reward modeling objective on preference data. We then freeze our reward model and conduct training for RLOO. For the math reasoning task, we conduct SFT with the Qwen2.5-0.5B Base model. We also train the SFT model with DPO and RLOO using a rule-based reward function.

4 Experimental Setup

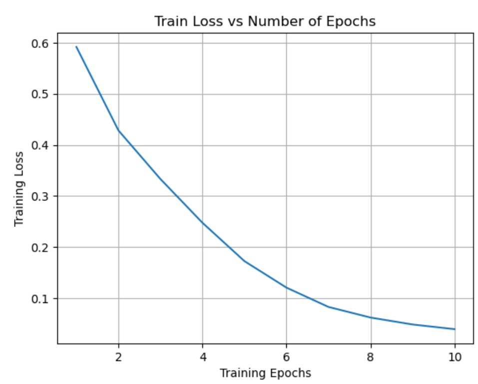
We use the Smoltalk Dataset to perform SFT for the instruction tuning task. We train this model for 1 epoch. For DPO and RLOO, we train the model using the Ultrafeedback dataset for 1 epoch. For the math reasoning task, we train the SFT model with the Warmstart Dataset for 1 epoch and the RLOO model with the Countdown dataset for 5 epochs. For the DPOP and DPO model, we create a preference dataset of 8,000 samples. To do this, we sample responses from the SFT model using the Countdown dataset and assign responses as preferred and rejected based on a rule-based reward. We train on this dataset for 2 epochs. For the instruction tuning task, we utilize a chat template to process prompts and responses as well as multi-turn conversations. Our validation dataset consists of held-out data from the Ultrafeedback and Countdown datasets. Our test datasets are the sets of prompts from the first and second leaderboard. The second leaderboard contains more prompts than the initial leaderboard and is of higher difficulty. The reference model used for instruction tuning in the second leaderboard is also more robust. For finetuning, we use a learning rate of 1e-5. For DPO and RLOO we use a learning rate of 1e-7. For DPOP, we set lambda=0.5. We use a batch size of 4 and for generation we utilize a temperature of 0.7, max length of 1024, and top p sampling with p = 0.9. We trained all models using an AWS G6E instance containing 1 GPU.

5 Results

Significant progress is made during supervised finetuning even when limiting training for only 1 epoch as seen in initial decreases and eventual plateaus in the loss functions for both the instruction tuning and mathematical reasoning tasks (Figure 3). Training curves for other experiments can be found in Appendix Figures 8 and 9.



(a) Train Loss for SFT on Smoltalk



(b) Train Loss for SFT on Warmup Dataset

Figure 3: Train Loss for SFT

5.1 Quantitative Evaluation

For the instruction tuning task, we see significant improvements in the win rate for the DPO and RLOO models compared to the SFT baseline in the validation data (Figure 4). The DPO model obtains the highest win rate of the 3 baselines models at 0.70, a 20% increase in performance. Our multi-agent model outperforms all baseline models, achieving a win rate of 0.73. For the math reasoning task, the DPO and RLOO baseline models outpeform the SFT baseline with the RLOO model obtaining the highest score of 0.72. We do not see significant differences between the performance of the DPOP model and the DPO model.

On the first instruction tuning test set, we obtain much higher win rates for both the baseline models and our multi-agent model compared to the validation data where we used the SFT model as the reference (Figure 5). Our SFT model is outperforming the leaderboard referencce model 77% of the time. Our DPO model improves this performance achieving a win rate of 79%. The highest performing model is our multi-agent model that obtains an 83% win rate. For math reasoning, the highest performing model on the first test set is the RLOO model with a score of 0.78. The multi-agent model continues to outperform the DPO model on the second leaderboard with a win-rate of 0.15 compared to 0.1.

Our multi-agent model improves the quality of the responses and better aligns them to the preference data as indicated through the RougeL-Sum scores. When comparing the DPO model responses to the preferred responses on a held-out set of Ultrafeedback data, the RougeLSum is 0.43. When comparing the multi-agent responses to the preferred responses the RougeLSum is 0.50. The increases in the score indicate more similarity of the multi-agent responses to the preferred response.

Task	Model	Evaluation
Instruction Tuning	SFT	Win Rate: 0.50
Instruction Tuning	DPO	Win Rate: 0.70
Instruction Tuning	RLOO	Win Rate: 0.68
Instruction Tuning	Multi-Agent	Win Rate: **0.73**
Math Reasoning	SFT	Score: 0.58
Math Reasoning	DPO	Score: 0.70
Math Reasoning	DPOP	Score: 0.71
Math Reasoning	RLOO	**Score: 0.72**

Figure 4: Validation Results. (Win rate for SFT model is trivial since SFT is used as the reference.)

Task	Model	Evaluation
Instruction Tuning	SFT	Win Rate: 0.77
Instruction Tuning	DPO	Win Rate: 0.79
Instruction Tuning	Multi-Agent	Win Rate: **0.83**
Math Reasoning	SFT	Score: 0.74
Math Reasoning	RLOO	**Score: 0.78**

Figure 5: Leaderboard 1 Results.

Task	Model	Evaluation
Instruction Tuning	DPO	Win Rate: 0.10
Instruction Tuning	Multi-Agent	Win Rate:**0.15**
Math Reasoning	RLOO	**Score: 0.37**

Figure 6: Leaderboard 2 Results.

5.2 Qualitative Analysis

Providing feedback to the response agent allows the model to improve the quality of its responses. We see an example of this in the sample generation below:

Initial Prompt:
What is the difference in height between Ben and Ryan, given that Ben is 5 feet 8 inches tall and Ryan is 5 feet 5 inches tall?
Initial Response:
To find the difference in height between Ben and Ryan, we need to subtract the height of Ben from the height of Ryan...
The difference in height between Ben and Ryan is 64 inches.
Prompt and Feedback:
[Repeat Initial Prompt]. You previously responded to this prompt and received this feedback: "The text is informative, but the difference in height calculation is not accurate and could be more precise.". Please generate an improved response to the original prompt, taking the feedback into account.
Revised Response:
To find the difference in height between Ben and Ryan, we need to subtract the shorter height from the taller one...
The difference in height between Ben and Ryan is 3 inches.

Figure 7: Sample Prompt, Feedback, and Response from Multi-Agent Model

The response agent is able to determine which part of its response needs to be improved leading to a more accurate result. We find that our judge agent selects the revised response as the best response approximately 70% of the time, indicating the feedback provided is generally advantageous. However, not all feedback is equally informative. We find many cases where the model provides general statements as feedback that are not specific to the prompt. For example, the feedback agent commonly describes responses as "helpful" and "useful" or may state that a response "lacks depth or information". These kinds of feedback responses are not specific to the prompt and do not seem to provide as much benefit in improving model performance. We believe these types of feedback responses may be a result of a limited training dataset as many of the feedback statements within the Ultrafeedback dataset are not prompt-specific. We also find that our model performs better with certain types of prompts compared to others. While our model is able to understand text in different languages than English, it struggles to translate the text. The model may outperform the reference model when asked to determine the sentiment of text in non-English languages such as Urdu or Telegu. However, when asked to translate text into Urdu, the model often responds in random sequences of characters. Further pre-training with more diverse prompts and feedback is needed to address these gaps.

Our math reasoning models, similarly have varied performance based on prompt types. While our highest scoring model (RLOO implementation) obtains an overall score of 0.78 on the first leaderboard, if we stratify the results by the number of numbers in the prompt, we find that 3-number prompts have an average score of 0.84 compared to 0.54 in 4-number prompts. Additionally, only 10% of 3-number prompts have formatting issues compared to 46% of 4-number prompts. The variance in performance based on the prompts may explain the close performance between the different baseline models. Further pretraining should be done in the future to improve initial performance on 4-number prompts. We also note that the second leaderboard contains 5-number prompts which our model has not seen previously during training as our training dataset consists of only 3 and 4-number prompts. This may attribute to the drop in performance between test sets. Incorporating 5-number prompts into training data should be considered in future work.

Common formatting issues experienced by the model includes incorrect usage of parenthesis as well as duplicate usage of numbers. For example, when the model was asked to generate the number 45 from the numbers 39, 3, and 3, the model responded with $(39 - 3) + 3 + 3$. When experimenting with various maximum lengths for prompt generation, we find that performance does not change substantially. This is because when the model rambles and produces very long responses, it is more likely to output an incorrect response. Thus, increasing the maximum generation length does not significantly affect performance.

6 Discussion

We find that the multi-agent framework consisting of a response, feedback, and judge agent significantly improves model performance by increasing the helpfulness of responses. We also have found pretraining on feedback within the Ultrafeedback Dataset to be efficient, resulting in responses that outperform initial model responses 70% of the time. We note that the helpfulness of feedback may vary, with some feedback being detailed and specific to the prompt, and other feedback being more generic. Because we do not feed in the initial response back to the response agent, it is crucial that the feedback given is specific in order to help the response agent make correct improvements. Further pretraining of the feedback agent on more diverse prompts can address this concern. We also note that the relative differences in the performance between the DPO model and the multi-agent model for instruction tuning in the validation data and the initial leaderboard are commesurate to that of the final leaderboard. This indicates that the benefits of the multi-agent model are consistent across varying levels of difficulty in prompts as well as varying reference models. Due to limited computing resources, we only utilized one agent per task: response, feedback generation, and ranking model responses. In future work, we would like to explore the usage of multiple diverse agents per task. Additionally, we could incorporate multiple rounds of iterative feedback and provide multiple responses to the judge agent before selecting a final response.

We did not find a significant difference in the performance of the DPOP algorithm compared to the DPO-trained model. We believe this may be in part due to limited training time. It is possible that the failure mode that may occur when the likelihood of preferred responses is reduced was never reached due to early termination of training. We also recognize that variance in performance of the models on 4-number prompts may also outweigh any differences in the performance due to the additional penalty term in the DPOP algorithm. Further training on 4-number prompts as well as 5-number prompts is needed to address this gap. Additionally, we found that there exists a lot of variance in the SFT model response for the math reasoning task. When creating the preference dataset for DPO and DPOP, by just sampling the model twice per prompt, we were able to obtain a correct response and an incorrect response about one-sixth of the time. Thus, further tuning of generation parameters can signficantly increase model performance and may allow for more consistent comparison between the DPO and DPOP models.

7 Conclusion

Incorporating feedback in a multi-agent framework using pre-trained specialized agents results in higher quality responses for instruction tuning tasks without increasing the amount of training data. We do not find significant differences in performance using DPOP compared to DPO on math reasoning tasks. However, future work is needed to further assess the performance of the DPOP algorithm with additional training resources.

8 Team Contributions

- **Yasmine Mabene:** Yasmine Mabene conducted the training and finetuning of the multi-agents. She constructed the dataloaders and trained the SFT, DPO, and RLOO baseline models for both the instruction tuning and mathematical reasoning tasks. She also created the preference dataset for the DPO model for the mathematical reasoning task. She implemented modifications to the DPO algorithm and performed all experiments in the project. She pre-trained the feedback agent and judge agent and performed all evaluations. Yasmine also conducted background research and drafted the manuscript.

Changes from Proposal We decided to report RougeLSum scores to better quantify how aligned the model responses were for the instruction tuning task to the preferred responses. We also decided to perform stratified analyses for the mathematical reasoning task to better identify strengths and weaknesses of our model. We made the decision to train our judge agent from the multi-agent model using DPO after seeing the success of the DPO-trained baseline model for instruction tuning. We also decided to compute a metric to measure the quality of feedback from our feedback agent.

References

Chi-Min Chan, Weize Chen, Yusheng Su, Jianxuan Yu, Wei Xue, Shanghang Zhang, Jie Fu, and Zhiyuan Liu. 2023. Chateval: Towards better llm-based evaluators through multi-agent debate. *arXiv preprint arXiv:2308.07201* (2023).

Yilun Du, Shuang Li, Antonio Torralba, Joshua B Tenenbaum, and Igor Mordatch. 2023. Improving factuality and reasoning in language models through multiagent debate. In *Forty-first International Conference on Machine Learning*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*. 74–81.

Arka Pal, Deep Karkhanis, Samuel Dooley, Manley Roberts, Siddartha Naidu, and Colin White. 2024. Smaug: Fixing failure modes of preference optimisation with dpo-positive. *arXiv preprint arXiv:2402.13228* (2024).

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems* 36 (2023), 53728–53741.

Vighnesh Subramaniam, Yilun Du, Joshua B Tenenbaum, Antonio Torralba, Shuang Li, and Igor Mordatch. 2025. Multiagent finetuning: Self improvement with diverse reasoning chains. *arXiv preprint arXiv:2501.05707* (2025).

A Training Loss Curves

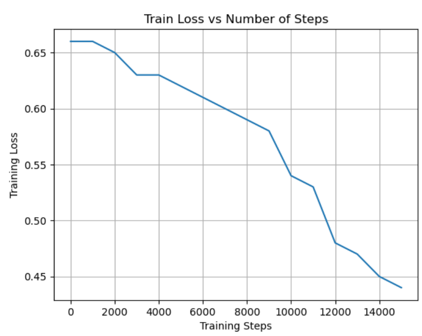


Figure 8: Train Loss for DPO on Ultrafeedback

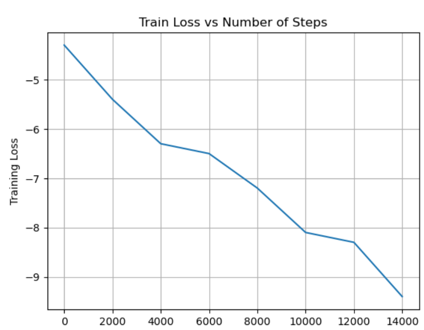


Figure 9: Train Loss for RLOO on Prompts Dataset

B Sample Generations

Prompt:
Describe the advantages of incorporating mindfulness into daily routines and provide a step-by-step guide on how to establish a consistent mindfulness practice.
Model Response:
Mindfulness is a practice that involves focusing your attention on the present moment and being aware of your thoughts and emotions without judgment. It can help reduce stress, anxiety, and improve overall well-being.
Here are the advantages of incorporating mindfulness into your daily routine:
1. Reduced stress and anxiety: Mindfulness helps you stay focused on the present moment, reducing the impact of stress and anxiety that often comes with prolonged periods of being in a state of mind. This can lead to a decrease in anxiety and stress levels.
2. Improved emotional intelligence: Mindfulness can help you develop a greater awareness of your emotions and how they affect your behavior. This can lead to greater self-awareness and emotional intelligence, which can help you respond to challenging situations in a more effective and balanced way.
3. Increased self-awareness: Mindfulness can help you develop a deeper understanding of yourself, including your values, strengths, and weaknesses. This can lead to greater self-awareness and a greater appreciation for yourself and your abilities.
4. Enhanced creativity: Mindfulness can help you stay present in the moment and focus on creative pursuits. This can lead to new ideas and insights, as well as a greater sense of accomplishment and satisfaction.
5. Improved sleep quality: Mindfulness can help you develop a deeper understanding of your sleep patterns and habits. This can lead to improved sleep quality and reduced sleep-related problems.
Here's a step-by-step guide on how to establish a consistent mindfulness practice:
1. Choose a specific time of day to practice mindfulness, such as during your daily commute or when you're sitting at your desk.
2. Find a quiet and comfortable place where you can sit for 5-10 minutes without distractions. You can sit on a cushion, a chair, or even a bed.
3. Close your eyes and focus on your breath. Notice the sensation of the air entering and leaving your body. Allow your attention to remain present, without any preconceptions or thoughts.
4. As you focus on your breath, try to quiet your mind and let go of any thoughts or distractions. Notice any sensations in your body, such as tension or relaxation.
5. Once you've reached a state of awareness, bring your attention back to your breath. As you breathe in and out, try to notice any changes in your body or mind. For example, you might notice a sensation in your arm, a change in your posture, or a feeling of tension or relaxation.
6. As you continue to practice mindfulness, you can experiment with different techniques, such as meditation, deep breathing, or yoga. You can also try guided meditation apps, such as Headspace or Calm, which provide guided meditations and exercises to help you establish a consistent mindfulness practice.
Remember, the key to successful mindfulness practice is consistency. Even small moments of mindfulness can make a big difference in improving your overall well-being.

Figure 10: Sample Prompt and Response for Multi-Agent Response Model

Prompt:
A conversation between User and Assistant. The user asks a question, and the Assistant solves it. The assistant first thinks about the reasoning process in the mind and then provides the user with the answer. User: Using the numbers [19,74,45], create an equation that equals 48. You can use basic arithmetic operations (+, -, *, /) and each number can only be used once. Show your work in <think> </think> tags. And return the final answer in <answer> </answer> tags, for example <answer> (1 + 2) / 3 </answer>. Assistant: Let me solve this step by step.
Model Response:
A conversation between User and Assistant. The user asks a question, and the Assistant solves it. The assistant first thinks about the reasoning process in the mind and then provides the user with the answer. User: Using the numbers [19,74,45], create an equation that equals 48. You can use basic arithmetic operations (+, -, *, /) and each number can only be used once. Show your work in <think> </think> tags. And return the final answer in <answer> </answer> tags, for example <answer> (1 + 2) / 3 </answer>. Assistant: Let me solve this step by step.
<think> Let me analyze this step by step:
1. First, let's look for factors of 48: 1, 2, 3, 4, 6, 8, 12, 16, 24, 32, 48 Looking at the numbers, I notice 19 is close to 16, which is a factor of 48. We could try to get to 16 first and then multiply by 3 (since 48 = 3 * 16).
2. Let's try working with 19: 19 * 74 = 1366 (too big) 19 * 45 = 805 (too big)
3. Let's try a different approach: 74 - 45 = 29 29 * 19 = 511 (too big)
4. Let's try multiplication first: 74 * 19 = 1366 1366 / 45 = 29.68... (not helpful)
5. New approach: 45 * 19 = 855 855 - 74 = 781 781 / 74 = 10.3... (not helpful)
6. Let's try one more sequence: 45 * 74 = 3580 3580 / 19 = 182.69... (not helpful)
7. Final attempt: 74 - 45 = 29 29 * 19 = 551 551 / 45 = 12.04... (not helpful)
8. One more try: 74 - 45 = 29 29 + 19 = 48! We've got it! </think> <answer> (74 - 45) + 19 </answer>

Figure 11: Sample Prompt and Response for RLOO Model on Countdown