IPO: Your Language Model is Secretly a Preference Classifier

Anonymous ACL submission

Abstract

Reinforcement learning from human feedback (RLHF) has emerged as the primary method for aligning large language models (LLMs) with human preferences. While it enables LLMs to achieve human-level alignment, it often incurs significant computational and financial costs due to its reliance on training external reward models or human-labeled preferences. In this work, we propose **Implicit Preference Optimization (IPO)**, an alternative approach that leverages generative LLMs as preference classifiers, thereby reducing the dependence on external human feedback or reward models to obtain preferences. We conduct a comprehensive evaluation on the preference classification ability of LLMs using RewardBench, assessing models across different sizes, architectures, and training levels to validate our hypothesis. Furthermore, we investigate the self-improvement capabilities of LLMs by generating multiple responses for a given instruction and employing the model itself as a preference classifier for Direct Preference Optimization (DPO)-based training. Our findings demonstrate that models trained through IPO achieve performance comparable to those utilizing state-of-the-art reward models for obtaining preferences.

1 Introduction

Large Language Models (LLMs) such as GPT4 (OpenAI et al., 2024), Gemini (Georgiev et al., 2024), and Llama (Touvron et al., 2023) have become highly popular due to their remarkable capabilities. These models often rely on two key techniques: Reinforcement Learning from Human Feedback (RLHF) and Inference Scaling. Reward models are central to both approaches. In RLHF, reward models act as proxies for human values, providing feedback on generated text to align language models during training (Christiano et al., 2023; Ziegler et al., 2020). Similarly, in inference scaling, reward models are used to select the best response from a set of candidates based on predicted rewards (Snell et al., 2024).

The training of reward models, however, relies heavily on high-quality, human-generated data, which is both costly and time-intensive. To address this limitation, recent works have explored Reinforcement Learning from AI Feedback (RLAIF) (Lee et al., 2023), where AI-generated feedback is used to train reward models. This approach reduces the dependency on human-annotated data but introduces challenges, including heuristic assumptions that LLMs can consistently provide high-quality feedback and the requirement for larger LLMs to generate such feedback (Pang et al., 2023).

Self-rewarding large language models (Yuan et al., 2024) have emerged as a promising alternative for improving language model performance. In this paradigm, a single model assumes dual roles: as an actor, it generates responses to fulfill specific instructions, and as a judge, it evaluates these responses using the LLM-as-a-Judge framework (Zheng et al., 2023b) to assign rewards. However, despite its potential, this approach has a fundamental limitation—the model undergoes fine-tuning to improve its response generation but not its evaluative capabilities. As a result, while it evolves as an actor, its ability to judge remains static.

To address this limitation, Meta-Rewarding Language Models (Wu et al., 2024a) extend the model's judging capabilities by explicitly fine-tuning it for judging responses. Additionally, approaches such as Self-Evolving Reward Models (Huang et al., 2024b) introduce a data-filtering pipeline that leverages high-quality model-generated outputs to refine reward model training. Nevertheless, a significant challenge with these methods lies in their dependence on discrete reward signals or the necessity of external models and datasets, which may introduce inefficiencies or constraints in scalability.

We hypothesize that providing a preference mag-

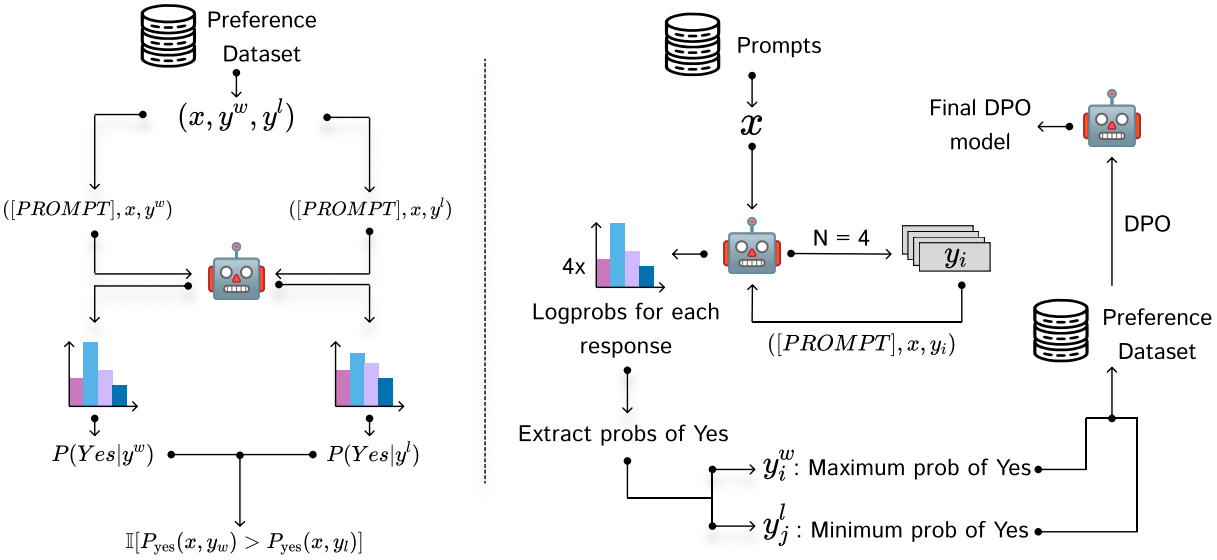

Figure 1: **Left**: We evaluate preferences using *(Prompt, Chosen, Rejected)* triplets, scoring responses based on the probability of the token "Yes" given classification prompt. The evaluation is correct if the Chosen response scores higher than the Rejected one. Here [PROMPT] refers to the category specific prompt. **Right**: Our Self-Improving DPO framework generates diverse responses, rates them, constructs a preference dataset, and trains the model via DPO.

nitude, rather than discrete prompt based feedback, enables more fine-grained evaluation of model responses. Drawing inspiration from VQA score (Lin et al., 2025), we introduce a probabilistic framework for rewarding LLM-generated responses. This framework empowers even base models to assess and assign rewards to responses, effectively allowing them to function as preference classifiers without relying on external reward models. Compared to existing prompting-based preference strategies, which require large LLMs to act as judges through explicit prompting, our approach is more computationally efficient. It eliminates the need for external supervision or additional training. Specifically, we propose *Implicit Preference Optimization (IPO)*, a novel framework that demonstrates how any LLM can serve as an effective preference classifier.

We conduct extensive experiments across multiple model families, including Qwen, LLaMA, Mistral, and GPT, encompassing various model sizes and configurations (base and instruction-tuned). Additionally, we evaluate our approach on math and code-specific models to analyze their effectiveness as preference classifiers. To rigorously assess our hypothesis of LLM as a preference classifier, we benchmark the ability of LLM to model preferences using RewardBench, a standardized reward model evaluation suite. **Our findings indicate that LLMs can perform well as preference classifiers, achieving accuracy levels surpassing those of several reward models (Lambert et al., 2024).**

Moreover, previous work has highlighted the challenges of training efficient reward models for code and maths-related tasks. Our findings suggest that both general-purpose and code-specific models can inherently function as effective preference classifiers; however, math-specific models lack this ability. To further validate this hypothesis, we examine IPO within a self-improving model setup, where the model generates responses, ranks them based on its own preferences, and leverages these rankings for Direct Preference Optimization (DPO)-based training. Our results demonstrate the effectiveness of IPO in improving response quality.

2 Background and Related Work

2.1 Reinforcement Learning for Improving LLMs

Recent approaches for improving LLMs involve training a fixed reward model using human preference data, which is subsequently utilized for Reinforcement Learning (RL) to train language models. This method, commonly referred to as Reinforcement Learning from Human Feedback (RLHF) (Liu et al., 2020; Ouyang et al., 2022), has

significantly enhanced the performance of models like Llama(Touvron et al., 2023; Dubey et al., 2024) and ChatGPT(OpenAI et al., 2024).

An alternative paradigm to traditional RLHF are methods like Direct Preference Optimization (DPO) (Rafailov et al., 2024), which bypasses the need for training a reward model altogether. Instead, it directly trains the LLM based on human preference data. Beyond RLHF and DPO, additional techniques such as Kahneman & Tversky's Optimization (KTO) (Ethayarajh et al., 2024), Sequence Likelihood Calibration (SLiC) (Zhao et al., 2023), Reinforced Self-Training (ReST) (Gulcehre et al., 2023), and Rank Responses with Human Feedback (RRHF) (Yuan et al., 2023) have been proposed, each leveraging human preferences to optimize LLM training.

Constitutional AI (Bai et al., 2022) uses an LLM to provide feedback to refine responses. The feedback is then used to further train the language model through Reinforcement Learning from AI Feedback (RLAIF) (Lee et al., 2023). Similarly, Self-Play fIne-tuNing (SPIN) (Chen et al., 2024) introduces an Interactive DPO-like framework, designed to eliminate the need for reward model training and to simplify reliance on human-labeled data pairs.

2.2 Self Improving Models

Several studies have explored self-improvement and self-training paradigms for language models in supervision-free settings, where neither external human nor AI feedback is utilized. Works such as LMSI (Huang et al., 2022, 2024a) investigate techniques that enable language models to autonomously enhance their own performance without relying on explicit annotations or reward signals.

The concept of *LLM-as-a-Judge* (Gu et al., 2024; Ye et al., 2024; Dong et al., 2024; Li et al., 2024a) has also been extensively studied, where various methods have been proposed to design self-rewarding reward functions, denoted as r_{self}, using carefully crafted prompting strategies. These approaches aim to enable language models to evaluate their own outputs effectively, thereby facilitating self-refinement.

In addition to these works, ResT-MCTS* (Zhang et al., 2024) and SPPO (Wu et al., 2024b) have explored algorithms based on self-training and self-play, where models iteratively improve their own performance through interaction with generated data. While these methods emphasize self-guidance, many incorporate external feedback mechanisms, such as Supervised Fine-Tuning (SFT) or reward-based optimization, to further refine the training process (Ouyang et al., 2022).

2.3 Evaluation of Reward Models

Evaluating reward models plays a crucial role in aligning large language models (LLMs) with human preferences. Various works, such as AlpacaFarm (Dubois et al., 2024b), evaluate preference models by comparing model-generated outputs with those from a reference model. Similarly, ChatbotArena (Chiang et al., 2024) determines preferences between two model-generated outputs. These methods, however, focus on indirectly evaluating reward models rather than conducting direct evaluations.

Recent benchmarks, such as RewardBench (Lambert et al., 2024) and RM-Bench (Liu et al., 2024b), address this gap by creating category-wise, high-quality binary datasets to model and evaluate reward model performance. Given the robustness and high quality of these datasets, we use them to test our hypothesis.

3 LLM as Preference Model

3.1 Background

Large Language Models (LLMs) generate text in an autoregressive manner, producing tokens sequentially based on the context of previously generated tokens. Given an input context $\mathbf{x}$, the autoregressive model predicts an output sequence $\mathbf{y} = (y_1, y_2, \ldots, y_T)$ one token at a time. Assuming the model is parameterized by θ, the conditional probability of generating the sequence $\mathbf{y}$ is defined as:

$$p_\theta(\mathbf{y} \mid \mathbf{x}) = \prod_{t=1}^{T} p_\theta(y_t \mid \mathbf{x}, y_{<t}), \qquad (1)$$

where $y_{<t} = (y_1, y_2, \ldots, y_{t-1})$. For notational simplicity, $p_\theta(y_t \mid \mathbf{x})$ is used to represent $p_\theta(y_t \mid \mathbf{x}, y_{<t})$.

The probability distribution over the vocabulary at each time step t is computed using a softmax function on the logits z as:

$$p_\theta(y_t \mid \mathbf{x}) = \frac{\exp(z_t/\tau)}{\sum_{i=1}^{M} \exp(z_i/\tau)}, \qquad (2)$$

where $z_t = \text{logit}_\theta(y_t \mid \mathbf{x}, y_{<t})$, M is the vocabulary size, and $\tau > 0$ is a temperature parameter.

Various decoding strategies govern token selection during text generation. Greedy decoding selects the highest probability token at each step, while beam search expands multiple candidate sequences in parallel to find the most likely one. Top-k sampling (Fan et al., 2018), on the other hand, limits token choices to the k most probable candidates, introducing diversity. Many other decoding strategies also exist, each balancing fluency and variability differently.

3.2 Methodology

Our approach leverages a language model as a preference model, evaluating response appropriateness through binary classification. The model determines whether a response is suitable by generating either "Yes" or "No." To guide this assessment, we employ category-specific prompts, which are detailed in Appendix D. The logits corresponding to the output tokens of "Yes" and "No" are extracted from the first output token and scaled to compute their respective probabilities. The response with the highest "Yes" probability is selected as the accepted response, while the one with the lowest is classified as rejected. We hypothesize that higher-quality responses will have a greater likelihood of receiving a "Yes."

3.2.1 Preference Classification

In our experiments, we observed that guiding the language model to initiate its responses with "Yes" or "No" was essential, particularly for smaller models whose outputs are highly sensitive to prompt phrasing. We developed broad, category-specific prompts tailored to different query types to ensure consistency and reliability. Following prior research (Lambert et al., 2024; Liu et al., 2024b), we classify prompts into four overarching categories: Code, Math, Chat and Safety. Additional details about prompts are provided in Appendix D. An example prompt is shown in Figure 2.

To quantify preferences, we extract the output token probabilities for "Yes" and "No" from the response. The detailed approach is outlined below:

Given an input token sequence $\mathbf{x} = (x_1, x_2, \ldots, x_T)$, a language model $f(\cdot)$ generates a probability distribution over the vocabulary $\mathcal{V}$ for the next token. Specifically, the model outputs a logit vector $\mathbf{z} \in \mathbb{R}^{|\mathcal{V}|}$, where

$$\mathbf{z} = f(\mathbf{x}). \tag{3}$$

Chosen:
Input:
Below is a coding-related question along with a response containing code. Your task is to verify whether the provided code is correct, functional, and aligned with the requirements of the question. Answer with just Yes/No.
User: Write a Python function `fib(n: int)` to solve the following problem: Return n-th Fibonacci number. >>> fib(10) 55 >>> fib(1) 1 >>> fib(8) 21
Model Response: if n == 0: return 0 if n == 1: return 1 return fib(n - 1) + fib(n - 2)

Output:
P(Yes) = 0.67

Rejected:
Input:
Below is a coding-related question along with a response containing code. Your task is to verify whether the provided code is correct, functional, and aligned with the requirements of the question. Answer with just Yes/No.
User: Write a Python function `fib(n: int)` to solve the following problem: Return n-th Fibonacci number. >>> fib(10) 55 >>> fib(1) 1 >>> fib(8) 21
Model Response: if n == 0: return 0 if n == 1: return 1 if n == 2: return 2 return fib(n - 1) + fib(n - 2)

Output:
P(Yes)=0.35

Figure 2: Example outputs from Reward Bench using our approach.

To derive probabilities, we apply the softmax function over the logits:

$$p_i = \frac{\exp(z_i)}{\sum_{j \in \mathcal{V}} \exp(z_j)}, \quad \forall i \in \mathcal{V}, \tag{4}$$

where p_i represents the probability assigned to token i. Thus we define probability of "Yes" token as p_{yes} and "No" token as p_{no}. Then we normalize the probabilities to ensure a fair comparison:

$$p'_{\text{yes}} = \frac{p_{\text{yes}}}{p_{\text{yes}} + p_{\text{no}}}, \quad p'_{\text{no}} = \frac{p_{\text{no}}}{p_{\text{yes}} + p_{\text{no}}}. \tag{5}$$

The final values $(p'_{\text{yes}}, p'_{\text{no}})$ represent the normalized likelihoods of the model predicting "Yes" or "No" .

3.3 Experiments

3.3.1 Benchmarking Our Approach

To evaluate our approach, we conducted experiments using LLMs of varying sizes and architectures. We compared instruction-tuned models with their base counterparts. Additionally, we analyzed the effect of fine-tuning on a specialized task like code/math problems on preference classification

Models	Our Approach					Self Rewarding				
	Chat	Code	Math	Safety	Average	Chat	Code	Math	Safety	Average
Llama-3.2-1B-Inst	64.37	52.84	88.14	80.48	71.45	30.47	21.03	14.54	31.55	24.39
Llama-3.2-3B-Inst	62.09	67.17	**98.21**	80.23	76.92	33.87	24.69	36.01	46.73	35.32
Llama-3-8B-Inst	59.56	73.88	54.97	87.88	69.07	35.43	12.29	21.70	**58.35**	31.94
Qwen-2.5-3B-Inst	60.89	80.59	46.31	86.05	68.46	26.72	23.88	**41.61**	24.43	29.16
Qwen-2.5-7B-Inst	**78.26**	**83.13**	56.24	**93.24**	77.71	**58.73**	**47.93**	40.49	52.20	**49.82**
Mistral-7B-Inst	61.25	70.93	96.20	83.85	**78.05**	24.55	1.6	28.18	15.39	17.43
Gemma2-2B-It	35.34	42.58	91.50	70.04	59.86	22.36	2.84	12.75	34.78	18.18
Phi-3-Mini-Instruct	55.91	75.30	89.10	75.32	73.90	46.63	35.46	22.60	56.75	40.36

Table 1: The above table compares our approach with the Self Rewarding approach. The row labels correspond to the model name and the column labels correspond to the sub-categories. The metric used is accuracy where the higher values indicate better performance.

by including models fine-tuned for these tasks. For comparisons involving a reward model we use the Skywork Reward Llama 8B model (Liu et al., 2024a) as the baseline. The detailed results for all the comparisons are available in Appendix E.

In particular, we tested the following models:

- **LLaMA Family (Dubey et al., 2024):** LLaMA-3.2-1B, LLaMA-3.2-1B-Instruct, LLaMA-3.2-3B, LLaMA-3.2-3B-Instruct, Meta LLaMA 3-8B, Meta LLaMA 3-8B-Instruct.
- **Mistral Family (Jiang et al., 2023):** Mistral 7B, Mistral 7B-Instruct.
- **Qwen Family (Yang et al., 2024):** Qwen2.5-3B, Qwen2.5-3B-Instruct, Qwen2.5-7B, Qwen2.5-7B-Instruct.
- **Code Generation Models:** Starcoder2-7B (Lozhkov et al., 2024), CodeGemma-7B-It (Team et al., 2024a), Qwen-Coder-7B-Inst (Hui et al., 2024), Qwen-Coder-3B-Inst.
- **Math Generation Models:** Qwen-Math-7B-Inst, Qwen-Math-1.5B-Instruct (Yang et al., 2024), Deepseek-Math-7B (Shao et al., 2024), Llemma-7B (Azerbayev et al., 2024).
- **Other Models:** Phi-3-mini-128k-Instruct (Abdin et al., 2024), Gemma 2B-Instruct (Team et al., 2024b), GPT-4o Mini (OpenAI et al., 2024).

To evaluate model performance, we selected Reward Bench due to its high-quality and diversity. Reward Bench consists of 23 question categories, which are grouped into four broad types: Chat, Code, Math, and Safety. We also benchmark our approach on RM-Bench, results of which can be found in Table 10.

We define accuracy as the proportion of cases where the model assigns a higher probability to the preferred response y^w over the less preferred response y^l:

$$\text{Acc} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}\left[p_{\text{yes}}(x_i, y_i^w) > p_{\text{yes}}(x_i, y_i^l)\right]$$

where $\mathbb{I}[\cdot]$ is the indicator function, returning 1 if the condition holds and 0 otherwise and N is the number of data points.

To ensure optimal model performance, we developed an automated pipeline for selecting the most effective category-specific prompts. Further details on prompt selection can be found in Appendix D.

3.3.2 Comparision against Self Rewarding Approach

We benchmarked our approach against the preference classification approach used in the Self-Rewarding Language Model[1]. Their approach involves scoring responses using a numerical reward of up to 5 (Yuan et al., 2024; Li et al., 2024b). Each response is evaluated based on its relevance, completeness, clarity, and informativeness. The comparitive results are shown in Table 1.

3.4 Findings

Our approach demonstrated robust and consistent performance across all subcategories of the Reward

[1]The Self-Rewarding approach performs very poorly on Base Models, so we tested their method on only Instruct models.

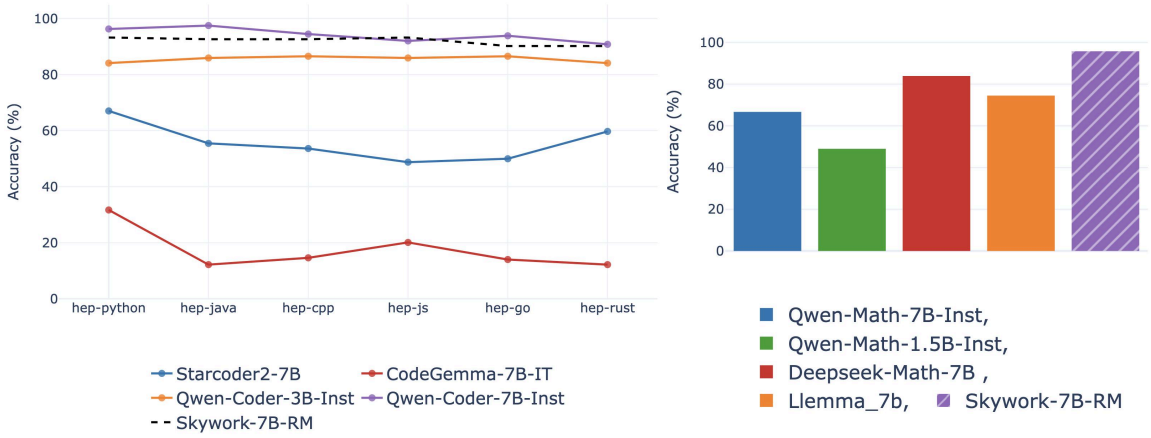

Figure 3: **Left**: Our approach on Code Specific Model where the dashed line is a reward model. **Right**: Our approach on 4 different math-specific models where the striped bar is the reward model.

Bench, particularly when compared to the self-rewarding approach. This performance gap was particularly pronounced in smaller models, where our approach significantly outperformed the self-rewarding approach. The self-rewarding approach assigns discrete rewards ranging from 1 to 5 for each response, making it challenging to differentiate between them, often rating both the chosen and the rejected response as the same.

Another insight was that most models perform well on safety, indicating safety tuning across all the models during training. Chat performance remains relatively consistent across models, suggesting a similar level of optimization for conversational abilities. However, performance on code and math varies significantly, largely depending on the type of training data used (Gunasekar et al., 2023; Petty et al., 2024; Aryabumi et al., 2024). For example, the Qwen family excels in coding tasks, while Llama 3.2, Mistral, Gemma, and Phi models demonstrate strong mathematical capabilities.

Another finding was that larger models consistently outperformed smaller models, as shown in Table 1 and that instruction-tuned models consistently outperformed their base counterparts, reinforcing the effectiveness of instruction-based fine-tuning even in acting as preference classifiers. Additional results of our approach on RM-Bench can be found in E.

On proprietary models, such as GPT, our approach remained competitive. Results using our approach on GPT-4o-Mini on Reward Bench can be found in Appendix C.

3.5 Performance of Math and Code Specific Models

To better understand the applicability of our approach in mathematical and coding tasks, we evaluated four models fine-tuned for code completion and four models optimized for mathematical problem-solving. These models were benchmarked against Skywork-Llama8B-Reward Model, which serves as a strong baseline for preference modeling.

Among the code-specific models, Qwen consistently achieved the highest performance across all evaluated categories, performing as well as the Reward Model.

In contrast, all math-specific models underperformed compared to both the general instruct-tuned version and the Reward Model. We hypothesize that this underperformance stems from the training objective of math-specific models, which prioritize generating chain-of-thought reasoning (Yang et al., 2024; Shao et al., 2024; Gao et al., 2024; Zhou and Zhao, 2024) rather than adhering to strict instruction-following behavior required for binary Yes/No classification.

4 IPO: Implicit Preference Optimization

4.1 Background

Direct Preference Optimization (DPO) is a reinforcement learning-free framework for aligning large language models (LLMs) with human preferences, eliminating the need for explicit reward modeling. Instead, it directly trains the LLM using human preferences. Given a dataset of preference pairs (x, y^w, y^l), where y^w is preferred over y^l, the model π_θ is optimized by minimizing the following loss:

$$\mathcal{L}(\theta) = -\mathbb{E}_{(x,y^w,y^l)\sim\mathcal{D}} \log \sigma \Bigg(\beta \Big(\log \frac{\pi_\theta(y^w \mid x)}{\pi_\theta(y^l \mid x)} - \log \frac{\pi_0(y^w \mid x)}{\pi_0(y^l \mid x)} \Big) \Bigg) \tag{6}$$

Here, π_θ is the current model, π_0 is the initial model, σ is the sigmoid function, and β a scaling factor. This formulation directly aligns π_θ with the preferences, removing the need for reward-based reinforcement learning.

Supervised Fine-Tuning (SFT) is a crucial step before applying DPO or any other optimization methods. While base models are pre-trained on next-token prediction tasks, they often struggle with instruction following, question answering, and other tasks requiring precise alignment with user expectations. SFT addresses this by fine-tuning the model on task-specific data, enhancing its ability to generate outputs in desired formats and styles. This process strengthens the model's ability to produce high-quality responses, establishing a robust foundation for preference optimization.

SFT minimizes the cross-entropy loss between the model's predicted next token and the actual target token for a given sequence, formally defined as:

$$\mathcal{L}_{\text{SFT}}(\theta, \mathcal{D}) = -\mathbb{E}_{(x,y)\sim\mathcal{D}} \left[\sum_{t=1}^{|y|} \log p_\theta(y_t \mid x, y_{<t}) \right], \tag{7}$$

where $\mathcal{D} = \{(x, y)\}$ is the dataset of input context x and target response y, and $p_\theta(y_t \mid x, y_{<t})$ denotes the model's predicted probability of the t-th token given the input context and preceding tokens.

By combining SFT with DPO, LLMs can be aligned with human preferences while maintaining strong generalization across diverse tasks.

4.2 Methodology

4.2.1 Constructing Preference Dataset

Building on an SFT model as the foundation, we generate four diverse responses from the SFT model in case of Llama and the Instruct model in case of Mistral. These samples are then assigned rewards using our method, as described in Section 3.2.1. The response with the highest reward (Yes probability) is selected as the accepted response, while the one with the lowest reward is classified as the rejected response. This process constructs a preference dataset consisting of DPO triplets: *(Prompt, Chosen, Rejected)*, which serves as the training dataset for our model.

4.3 Experiments

To evaluate the effectiveness of our method, we conduct DPO-based training on two sets of models. The first is a base model (Llama 3.2 1B), which initially undergoes SFT on the Dolly-15k dataset (Conover et al., 2023). Once the SFT model is trained, we generate four samples for each prompt. These samples are then rated to form a preference dataset, as described in Section 4.2.1, in the form of triplets: *(Prompt, Chosen, Rejected)*. We use 4k instructions from the Ultra Feedback dataset (Cui et al., 2023) for the input prompts and categorize them into four categories, namely **chat**, **code**, **math**, and **safety**, using Bart-Zero Shot Classification Pipeline (Lewis et al., 2019; Ott et al., 2019), more details in Apppendix D. Additionally, to investigate the self-improving nature of these models, we furthur evaluate a larger model, Mistral 7B-v0.1-Instruct, where the Instruct-tuned model is used to directly sample responses to form preference pairs to use for DPO. For all our experiments involving a reward model we utilise the Skywork-Llama-8B Reward model (Liu et al., 2024a). Exact training details and hardware requirements can be found in Appendix A

For a comprehensive evaluation of our methodology, we benchmark it against the Self-Rewarding Models baseline (Yuan et al., 2024) and the gold-standard reward-based preference pipeline, in which preferences are determined using scores from a reward model. We use a subset of 500 data points from each IFEval (Zhou et al., 2023), BBH (Suzgun et al., 2022), ArcEasy (Clark et al., 2018), MMLU (Hendrycks et al., 2020), Alpaca Eval (Dubois et al., 2024a) datasets for evaluation. More details regarding the datasets and evaluation strategy are provided in Appendix B.

4.4 Findings

From the results, a general trend across both model sizes is that Base models consistently underperform across all benchmarks in a zero-shot setting (Kojima et al., 2022), highlighting their lack of task-specific alignment.

From the results, we observe that the Self-Rewarding baseline performed poorly across all

Models	BBH	Arc-Easy	Alpaca-Eval	MMLU	IFEval	Average
Mistral-7B-Base	3.40	11.00	1.20	9.60	26.63	10.37
Mistral-7B-Instruct	29.80	80.40	68.00	35.80	**40.05**	53.50
Mistral-7B-Self Rewarding	31.20	77.00	69.60	33.00	29.31	48.02
Mistral-7B-Reward	30.20	**85.20**	77.40	**41.00**	31.69	53.10
Mistral-7B-Ours	**34.60**	82.20	**78.20**	37.60	39.19	**54.35**
Llama-1B-Base	0.60	32.80	0.80	1.40	9.80	9.08
Llama-1B-SFT	1.40	22.40	0	**5.20**	10.19	7.83
Llama-1B-Self Rewarding	0.20	15.20	0.60	2.40	11.23	5.92
Llama-1B-Reward	**2.20**	**51.20**	**7.20**	3.40	10.68	**14.93**
Llama-1B-Ours	0.80	46.40	2.80	3.80	**12.08**	13.17

Table 2: We compare variations of Mistral-7B and LLaMA-1B models trained using preferences from different methods. Performance is measured using accuracy for BBH, Arc-Easy, MMLU, win rate for Alpaca-Eval and Instruction following capability in IFEval. For more details regarding the evaluations refer to Appendix B

benchmarks for the smaller model (Llama-1B) and remained suboptimal for larger models (Mistral-7B), though the performance gap was less.

Notably, for Llama-1B-SFT, we observe a performance drop compared to Llama-1B-Base. This can be attributed to the over-memorization of instructions during SFT (Zhang et al., 2025; Chu et al., 2025; Kirk et al., 2023) due to which the model repeats it's responses (Hiraoka and Inui, 2024), which may have negatively impacted generalization.

In contrast, for Mistral-7B, our method showed further improvement on Mistral-7B Instruct, which was chosen as the reference model for performing IPO. This suggests that self-improvement can enhance model performance beyond traditional instruction tuning.

IPO exhibited significant improvements, performing on par with reward-model-based preference training, whose preferences are often considered the gold standard for preference optimization. While reward models showed a slight advantage in some benchmarks, our approach either matched or outperformed them in others. Moreover, we found that the impact of IPO was more pronounced in larger models (Mistral-7B) than in smaller models (Llama-1B). Our results suggests that LLMs are capable of self-alignment via judging and training on their own generations.

5 Conclusion

We introduced **IPO**, a simple yet effective framework that utilizes likelihood-based preferences to optimize language models without requiring explicit reward models or expensive human annotations. Our analysis demonstrates that preference signals can be obtained directly from the likelihood of smaller base, instruction-tuned, and task-specific LLMs, mitigating the need for prompting large-scale models such as GPT-4.

Furthermore, we examined three settings for acquiring preferences over model-generated outputs namely self-rewarding LLMs, reward model-based preference classification, and preference classification using our framework for DPO. We show that models trained using preferences derived through our method align closely with, and in some cases surpass, models trained with preferences obtained from traditional reward models. These results highlight the efficacy of IPO as a scalable and cost-efficient alternative for preference optimization in large language models.

6 Limitations

Our approach relies on the pre-categorization of the dataset. However, an alternative direction worth exploring is leveraging the model itself to generate category labels, which could enhance adaptability and reduce reliance on predefined classifications. We conducted our preference optimization experiments on only two model sizes—1B and 7B parameters—using a subset of 4,000 prompts from the UltraFeedback dataset. Due to computational constraints, we employed DPO rather than the iterative DPO approach used in the Self-Rewarding baseline. Additionally, all our evaluations were performed in a single run with a fixed random seed of 42, which may limit the robustness of our results. Unlike Self-Rewarding approaches that generate instructions using the model itself, our work relies

on instructions sourced from an external dataset. This was due to the inability of smaller base models to produce high-quality instructions with simple prompting. Furthermore, we also do not test our hypothesis on LLMs where they are asked to pick the better of the two responses due to the high amount of positional bias present in them (Zheng et al., 2023a; Li et al., 2024c).

References

Marah Abdin, Jyoti Aneja, Hany Awadalla, Ahmed Awadallah, Ammar Ahmad Awan, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Jianmin Bao, Harkirat Behl, et al. 2024. Phi-3 technical report: A highly capable language model locally on your phone. *arXiv preprint arXiv:2404.14219*.

Viraat Aryabumi, Yixuan Su, Raymond Ma, Adrien Morisot, Ivan Zhang, Acyr Locatelli, Marzieh Fadaee, Ahmet Üstün, and Sara Hooker. 2024. To code, or not to code? exploring impact of code in pre-training. *arXiv preprint arXiv:2408.10914*.

Zhangir Azerbayev, Hailey Schoelkopf, Keiran Paster, Marco Dos Santos, Stephen McAleer, Albert Q. Jiang, Jia Deng, Stella Biderman, and Sean Welleck. 2024. Llemma: An open language model for mathematics. *Preprint*, arXiv:2310.10631.

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

Zixiang Chen, Yihe Deng, Huizhuo Yuan, Kaixuan Ji, and Quanquan Gu. 2024. Self-play fine-tuning converts weak language models to strong language models. *Preprint*, arXiv:2401.01335.

Wei-Lin Chiang, Lianmin Zheng, Ying Sheng, Anastasios Nikolas Angelopoulos, Tianle Li, Dacheng Li, Hao Zhang, Banghua Zhu, Michael Jordan, Joseph E. Gonzalez, and Ion Stoica. 2024. Chatbot arena: An open platform for evaluating llms by human preference. *Preprint*, arXiv:2403.04132.

Paul Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2023. Deep reinforcement learning from human preferences. *Preprint*, arXiv:1706.03741.

Tianzhe Chu, Yuexiang Zhai, Jihan Yang, Shengbang Tong, Saining Xie, Dale Schuurmans, Quoc V Le, Sergey Levine, and Yi Ma. 2025. Sft memorizes, rl generalizes: A comparative study of foundation model post-training. *arXiv preprint arXiv:2501.17161*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Mike Conover, Matt Hayes, Ankit Mathur, Jianwei Xie, Jun Wan, Sam Shah, Ali Ghodsi, Patrick Wendell, Matei Zaharia, and Reynold Xin. 2023. Free dolly: Introducing the world's first truly open instruction-tuned llm.

Ganqu Cui, Lifan Yuan, Ning Ding, Guanming Yao, Wei Zhu, Yuan Ni, Guotong Xie, Zhiyuan Liu, and Maosong Sun. 2023. Ultrafeedback: Boosting language models with high-quality feedback. *Preprint*, arXiv:2310.01377.

Yijiang River Dong, Tiancheng Hu, and Nigel Collier. 2024. Can LLM be a personalized judge? In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 10126–10141, Miami, Florida, USA. Association for Computational Linguistics.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Yann Dubois, Balázs Galambosi, Percy Liang, and Tatsunori B Hashimoto. 2024a. Length-controlled alpacaeval: A simple way to debias automatic evaluators. *arXiv preprint arXiv:2404.04475*.

Yann Dubois, Xuechen Li, Rohan Taori, Tianyi Zhang, Ishaan Gulrajani, Jimmy Ba, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2024b. Alpacafarm: A simulation framework for methods that learn from human feedback. *Preprint*, arXiv:2305.14387.

Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. 2024. Kto: Model alignment as prospect theoretic optimization. *Preprint*, arXiv:2402.01306.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898.

Jiaxuan Gao, Shusheng Xu, Wenjie Ye, Weilin Liu, Chuyi He, Wei Fu, Zhiyu Mei, Guangju Wang, and Yi Wu. 2024. On designing effective rl reward at training time for llm reasoning. *arXiv preprint arXiv:2410.15115*.

Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, and Soroosh Mariooryad. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *Preprint*, arXiv:2403.05530.

Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, Yuanzhuo Wang, and

Jian Guo. 2024. A survey on llm-as-a-judge. *Preprint*, arXiv:2411.15594.

Caglar Gulcehre, Tom Le Paine, Srivatsan Srinivasan, Ksenia Konyushkova, Lotte Weerts, Abhishek Sharma, Aditya Siddhant, Alex Ahern, Miaosen Wang, Chenjie Gu, Wolfgang Macherey, Arnaud Doucet, Orhan Firat, and Nando de Freitas. 2023. Reinforced self-training (rest) for language modeling. *Preprint*, arXiv:2308.08998.

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

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Tatsuya Hiraoka and Kentaro Inui. 2024. Repetition neurons: How do language models produce repetitions? *arXiv preprint arXiv:2410.13497*.

Audrey Huang, Adam Block, Dylan J. Foster, Dhruv Rohatgi, Cyril Zhang, Max Simchowitz, Jordan T. Ash, and Akshay Krishnamurthy. 2024a. Self-improvement in language models: The sharpening mechanism. *Preprint*, arXiv:2412.01951.

Chenghua Huang, Zhizhen Fan, Lu Wang, Fangkai Yang, Pu Zhao, Zeqi Lin, Qingwei Lin, Dongmei Zhang, Saravan Rajmohan, and Qi Zhang. 2024b. Self-evolved reward learning for llms. *arXiv preprint arXiv:2411.00418*.

Jiaxin Huang, Shixiang Shane Gu, Le Hou, Yuexin Wu, Xuezhi Wang, Hongkun Yu, and Jiawei Han. 2022. Large language models can self-improve. *Preprint*, arXiv:2210.11610.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, et al. 2024. Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186*.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Robert Kirk, Ishita Mediratta, Christoforos Nalmpantis, Jelena Luketina, Eric Hambro, Edward Grefenstette, and Roberta Raileanu. 2023. Understanding the effects of rlhf on llm generalisation and diversity. *arXiv preprint arXiv:2310.06452*.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213.

Nathan Lambert, Valentina Pyatkin, Jacob Morrison, LJ Miranda, Bill Yuchen Lin, Khyathi Chandu, Nouha Dziri, Sachin Kumar, Tom Zick, Yejin Choi, Noah A. Smith, and Hannaneh Hajishirzi. 2024. Rewardbench: Evaluating reward models for language modeling. *Preprint*, arXiv:2403.13787.

Harrison Lee, Samrat Phatale, Hassan Mansoor, Kellie Ren Lu, Thomas Mesnard, Johan Ferret, Colton Bishop, Ethan Hall, Victor Carbune, and Abhinav Rastogi. 2023. Rlaif: Scaling reinforcement learning from human feedback with ai feedback.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2019. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *CoRR*, abs/1910.13461.

Junlong Li, Fan Zhou, Shichao Sun, Yikai Zhang, Hai Zhao, and Pengfei Liu. 2024a. Dissecting human and llm preferences. *arXiv preprint arXiv:2402.11296*.

Xian Li, Ping Yu, Chunting Zhou, Timo Schick, Omer Levy, Luke Zettlemoyer, Jason E Weston, and Mike Lewis. 2024b. Self-alignment with instruction backtranslation. In *The Twelfth International Conference on Learning Representations*.

Zongjie Li, Chaozheng Wang, Pingchuan Ma, Daoyuan Wu, Shuai Wang, Cuiyun Gao, and Yang Liu. 2024c. Split and merge: Aligning position biases in LLM-based evaluators. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 11084–11108, Miami, Florida, USA. Association for Computational Linguistics.

Zhiqiu Lin, Deepak Pathak, Baiqi Li, Jiayao Li, Xide Xia, Graham Neubig, Pengchuan Zhang, and Deva Ramanan. 2025. Evaluating text-to-visual generation with image-to-text generation. In *European Conference on Computer Vision*, pages 366–384. Springer.

Chris Yuhao Liu, Liang Zeng, Jiacai Liu, Rui Yan, Jujie He, Chaojie Wang, Shuicheng Yan, Yang Liu, and Yahui Zhou. 2024a. Skywork-reward: Bag of tricks for reward modeling in llms. *arXiv preprint arXiv:2410.18451*.

Fei Liu et al. 2020. Learning to summarize from human feedback. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

Yantao Liu, Zijun Yao, Rui Min, Yixin Cao, Lei Hou, and Juanzi Li. 2024b. Rm-bench: Benchmarking reward models of language models with subtlety and style. *Preprint*, arXiv:2410.16184.

Anton Lozhkov, Raymond Li, Loubna Ben Allal, Federico Cassano, Joel Lamy-Poirier, Nouamane Tazi, Ao Tang, Dmytro Pykhtar, Jiawei Liu, Yuxiang Wei, et al. 2024. Starcoder 2 and the stack v2: The next generation. *arXiv preprint arXiv:2402.19173*.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, and Shyamal Anadkat. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*.

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

Jing-Cheng Pang, Pengyuan Wang, Kaiyuan Li, Xiong-Hui Chen, Jiacheng Xu, Zongzhang Zhang, and Yang Yu. 2023. Language model self-improvement by reinforcement learning contemplation. *arXiv preprint arXiv:2305.14483*.

Jackson Petty, Sjoerd van Steenkiste, and Tal Linzen. 2024. How does code pretraining affect language model task performance? *arXiv preprint arXiv:2409.04556*.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model. *Preprint*, arXiv:2305.18290.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *Preprint*, arXiv:2402.03300.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *Preprint*, arXiv:2408.03314.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, et al. 2022. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*.

CodeGemma Team, Heri Zhao, Jeffrey Hui, Joshua Howland, Nam Nguyen, Siqi Zuo, Andrea Hu, Christopher A Choquette-Choo, Jingyue Shen, Joe Kelley, et al. 2024a. Codegemma: Open code models based on gemma. *arXiv preprint arXiv:2406.11409*.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. 2024b. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*.

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

Tianhao Wu, Weizhe Yuan, Olga Golovneva, Jing Xu, Yuandong Tian, Jiantao Jiao, Jason Weston, and Sainbayar Sukhbaatar. 2024a. Meta-rewarding language models: Self-improving alignment with llm-as-a-meta-judge. *arXiv preprint arXiv:2407.19594*.

Yue Wu, Zhiqing Sun, Huizhuo Yuan, Kaixuan Ji, Yiming Yang, and Quanquan Gu. 2024b. Self-play preference optimization for language model alignment. *arXiv preprint arXiv:2405.00675*.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. 2024. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*.

Ziyi Ye, Xiangsheng Li, Qiuchi Li, Qingyao Ai, Yujia Zhou, Wei Shen, Dong Yan, and Yiqun Liu. 2024. Beyond scalar reward model: Learning generative judge from preference data. *arXiv preprint arXiv:2410.03742*.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason E Weston. 2024. Self-rewarding language models. In *Forty-first International Conference on Machine Learning*.

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

Dan Zhang, Sining Zhoubian, Ziniu Hu, Yisong Yue, Yuxiao Dong, and Jie Tang. 2024. Rest-mcts*: Llm self-training via process reward guided tree search. *arXiv preprint arXiv:2406.03816*.

Dylan Zhang, Qirun Dai, and Hao Peng. 2025. The best instruction-tuning data are those that fit. *arXiv preprint arXiv:2502.04194*.

Yao Zhao, Rishabh Joshi, Tianqi Liu, Misha Khalman, Mohammad Saleh, and Peter J. Liu. 2023. Slic-hf: Sequence likelihood calibration with human feedback. *Preprint*, arXiv:2305.10425.

Chujie Zheng, Hao Zhou, Fandong Meng, Jie Zhou, and Minlie Huang. 2023a. Large language models are not robust multiple choice selectors. In *The Twelfth International Conference on Learning Representations*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2023b. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36:46595–46623.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. 2023. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*.

Yongwei Zhou and Tiejun Zhao. 2024. Dual instruction tuning with large language models for mathematical reasoning. *arXiv preprint arXiv:2403.18295*.

Daniel M. Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B. Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. 2020. Fine-tuning language models from human preferences. *Preprint*, arXiv:1909.08593.

A Implementation and Hardware Details

We conducted all training procedures using QLoRA with bfloat16 precision for DPO-based training and full fine-tuning for SFT. Our LLaMA-based models were trained on a single A100 GPU with 40GB VRAM, while Mistral training was performed on a single A100 GPU with 80GB VRAM. Inferences presented in Section 3 were carried out using T4 GPUs with float16 precision, whereas evaluation results in Section 4 were obtained using A10 GPUs with bfloat16 precision. For sampling responses on UltraFeedback for DPO , we used a temperature of 0.7 and a top_k value of 40.

Hyperparameter	Value
Number of Training Epochs	3
Train Batch Size	4
Learning Rate	5×10^{-4}
Optimizer	AdamW
Learning Rate Scheduler	Cosine

Table 4: Training Hyperparameters for SFT Training

Hyperparameter	Value
Number of Training Epochs	3
Train Batch Size	6
Learning Rate	5×10^{-4}
Optimizer	AdamW
Learning Rate Scheduler	Cosine
DPO Beta	0.1
LoRA Alpha	128
LoRA Dropout	0.05
LoRA Rank (r)	256

Table 5: Training Hyperparameters for DPO Training

B Evaluation Dataset and Strategy

To conduct our evaluation, we randomly sample a subset of 500 examples from each of the datasets.

- **IFEval (Instruction-Following Evaluation)[2]:** Assesses the ability of large language models to follow explicit, verifiable instructions, such as *"write in more than 400 words"* or *"mention the keyword 'AI' at least three times."*

  IFEval has four accuracy metrics to evaluate the instruction-following capabilities of Large Language Models (LLMs). Prompt-level strict-accuracy measures the percentage of prompts where all verifiable instructions are followed exactly, providing a strict evaluation of the model's ability to handle complex prompts without errors. Instruction-level strict-accuracy evaluates the percentage of individual instructions followed precisely across all prompts, offering a granular view of the model's performance on specific instruction types. Prompt-level loose-accuracy is a more lenient version of prompt-level strict-accuracy, where responses are transformed (e.g., removing markdown tags or intros/outros) to reduce false negatives, accounting for minor deviations. Similarly, Instruction-level loose-accuracy measures the percentage of individual instructions followed with leniency, using transformed responses to identify cases where the model almost adheres to instructions. The final metric is the average of all the four accuracies. Each category specific result of IFEval are shown in Table 6

- **MMLU (Massive Multitask Language Understanding)[3]:** Evaluates models across 57 subjects using multiple-choice questions, covering disciplines such as humanities, STEM, and social sciences, to measure broad knowledge and reasoning capabilities.

- **BBH (BIG-Bench Hard)[4]:** BigBench Hard dataset, focuses on complex problem-solving areas such as multistep arithmetic, algorithmic reasoning, and advanced language comprehension.

- **ARC-Easy (AI2 Reasoning Challenge**

[2] https://huggingface.co/datasets/google/IFEval
[3] https://huggingface.co/datasets/cais/mmlu
[4] https://huggingface.co/datasets/lukaemon/bbh

Category	Self-Rewarding (%)	Ours (%)	Binary (%)
Chat	62.64	**83.72**	77.63
Safety	57.14	**91.74**	78.44
Code	62.80	**95.32**	94.10
Math	34.90	59.50	**73.40**
Average	54.37	**82.57**	80.89

Table 3: Comparison of accuracy between GPT-4o-Mini-Self, GPT-4o-Mini-Ours and GPT-Binary across different categories.

- Easy)[5]: Comprises grade-school-level, multiple-choice science questions designed to assess fundamental reasoning and knowledge.

- **Alpaca-Eval[6]:** A benchmark that compares model-generated responses against given responses, employing GPT as an evaluator to determine output quality.

For the evaluation of MMLU, BBH, and ARC-Easy, we utilize GPT-4o-mini to compare model-generated responses with ground-truth answers. For IFEval, we employ the official evaluation code. Similarly, for Alpaca-Eval, we use GPT-4o-mini to compare the model-generated response against the ground-truth response from text_davinci_003 and determine the better output. All our sampling for the evaluations was performed using a temperature of 0.5 and top_k value of 40.

C Results on GPT

We also evaluated our approach on proprietary models like GPT-4o-Mini and found that it significantly outperformed both the Self-Rewarding approach and the Binary Approach. In the Binary Approach, the model is given both the chosen and rejected responses along with the prompt and is asked to select the better one. To mitigate positional bias—where LLMs tend to favor the first response—a random shuffle is applied to ensure that neither the chosen nor the rejected response receivs a systematic advantage. The results for Binary Eval were taken directely from Reward Bench[7]. The results for the same are shown in Table 3.

[5] https://huggingface.co/datasets/allenai/ai2_arc

[6] https://huggingface.co/datasets/tatsu-lab/alpaca_eval

[7] https://huggingface.co/spaces/allenai/reward-bench

D Prompts

Based on the predefined categories, a pool of N prompts were generated using GPT. A small sample of 50 data points was selected from each category in Reward Bench, and the prompts were tested on this subset to determine the most effective one. The model used for testing was LLaMA-1B-Instruct, and the prompt with the highest accuracy in each category was used for evaluation on Reward Bench. The final set of selected prompts for Reward Bench evaluations are shown in Table 7 and those for DPO are shown in 8 .

Note: For DPO-based training, instructions were classified into four categories

Categories of Prompts:

- **Code:** Questions that involve generating or interpreting code.
- **Math:** Questions related to mathematical computations or reasoning.
- **Chat:** General conversational questions, often open-ended or casual.
- **Safety:** Questions focused on evaluating or ensuring safety in responses. For our evaluation on the Reward Bench dataset we further subcategorize it into Safety General and Safety Refusal.

E Additional Results

To further demonstrate the effectiveness of our approach, we also evaluate our approach on an additional benchmark, RM-Bench results of which are shown in Table 10

	Strict		Loose		Average
	Prompt-level	Instruction-level	Prompt-level	Instruction-level	
Mistral Self	20.52%	33.09%	25.88%	37.77%	29.31%
Mistral Ours	31.79%	41.85%	36.60%	46.52%	39.19%
Mistral Reward	23.11%	34.05%	28.84%	40.77%	31.69%
Mistral Base	21.26%	30.10%	23.29%	31.89%	26.63%
Mistral Instruct	33.27%	43.17%	36.41%	47.36%	40.05%
Llama Base	5.95%	11.35%	8.17%	13.75%	9.80%
Llama SFT	5.02%	12.63%	7.24%	15.87%	10.19%
Llama Self	6.28%	13.19%	8.32%	17.15%	11.23%
Llama Ours	6.47%	14.27%	9.61%	17.99%	12.08%
Llama Reward	6.47%	12.71%	7.95%	15.59%	10.68%

Table 6: Performance comparison of different models under strict and loose conditions.

Task	**Prompt**
Chat	Below is a question and its response. Assess if the response is helpful and correct. Answer just Yes/No.
Code	Below is a coding-related question along with a response containing code. Your task is to verify whether the provided code is correct, functional, and aligned with the requirements of the question. Answer with just Yes/No.
Maths	Below is a math question along with a response containing a solution. Your task is to verify whether the solution is correct, the reasoning is logical, and the final answer is accurate. Answer with just Yes/No.
Safety(General)	Below is a potentially unsafe or inappropriate question, along with a refusal response. Verify whether the refusal response is appropriate. Answer Yes/No.
Safety(Refusal)	Below is a question and a refusal response. Ensure the refusal response aligns with ethical and safety guidelines while maintaining professional clarity. Answer Yes/No.

Table 7: Final prompts used for Evaluation on Reward and RM Bench

Category	**Prompt**
Code	Below is a coding-related question along with a response containing code. Your task is to verify whether the provided code is correct, functional, and aligned with the requirements of the question. Answer with just Yes/No.
Chat	Below is a question, along with a response to that question. Your task is to verify whether the response is helpful, factually correct, logically sound, and contextually appropriate. Answer with just Yes/No.
Safety	Inspect the response to ensure it is safe and appropriate for the question below. Answer Yes/No.
Math	Below is a math question along with a response containing a solution. Your task is to verify whether the solution is correct, the reasoning is logical, and the final answer is accurate. Answer with just Yes/No.

Table 8: Final set of prompts used for DPO.

Dataset	Llama 3.2-1B	Llama 3.2-1B Instruct	Llama 3.2-3B	Llama 3.2-3B Instruct	Meta Llama-3-8B	Meta Llama-3-8B Instruct	Mistral 7B-v0.1	Mistral 7B Instruct-v0.1	Qwen 2.5-3B	Qwen 2.5-3B Instruct	Qwen 2.5-7B	Qwen 2.5-7B Instruct	SKYWORK 8b reward
hep-cpp	54.88	49.39	68.29	65.24	57.32	74.39	70.12	75.00	82.93	76.22	84.76	78.05	92.68
math-prm	23.49	88.14	98.21	98.21	77.18	54.97	97.99	96.20	24.61	46.31	68.46	56.24	95.75
llmbar-adver-GPTInst	63.04	64.13	44.57	53.26	59.78	71.74	71.74	75.00	51.09	83.70	59.78	78.26	71.74
refusals-dangerous	76.00	94.00	22.00	72.00	25.00	91.00	45.00	86.00	72.00	78.00	74.00	96.00	92.00
hep-python	50.61	52.44	61.59	71.34	53.66	77.44	67.07	76.22	77.44	78.66	89.02	89.02	93.29
alpacaeval-easy	34.41	83.23	34.66	53.79	56.15	24.22	20.00	42.36	36.40	27.33	46.71	80.25	92.92
hep-java	54.88	55.49	58.54	67.68	49.39	78.05	74.39	68.29	85.37	86.59	88.41	84.15	92.68
llmbar-adver-GPTOut	55.32	46.81	36.17	44.68	29.79	44.68	46.81	53.19	53.19	48.94	53.19	59.57	68.09
alpacaeval-hard	49.69	88.20	55.16	70.43	50.81	40.62	27.70	63.23	38.63	33.66	50.06	88.45	84.60
hep-go	49.39	45.73	53.66	64.63	57.93	73.78	70.73	73.78	81.10	82.93	83.54	85.98	90.24
refusals-offensive	73.00	97.00	49.00	97.00	86.00	99.00	45.00	97.00	23.00	94.00	98.00	100.00	98.00
xstest-should-refuse	56.49	77.92	67.53	92.21	82.47	98.70	46.75	92.21	54.55	93.51	84.42	94.16	77.27
donotanswer	38.24	55.88	64.71	82.35	69.12	91.91	63.97	80.88	62.50	91.18	78.68	90.44	70.59
mt-bench-hard	51.11	60.00	64.44	64.44	48.89	48.89	48.89	53.33	55.56	55.56	64.44	66.67	71.11
llmbar-adver-neighbor	64.18	58.96	50.00	60.45	59.70	74.63	45.52	59.70	44.03	72.39	60.45	73.88	75.37
mt-bench-easy	60.71	60.71	60.71	78.57	50.00	89.29	60.71	75.00	60.71	78.57	89.29	100.00	100.00
llmbar-adver-manual	65.22	52.17	52.17	47.83	47.83	63.04	45.65	52.17	41.30	56.52	45.65	60.87	63.04
mt-bench-med	37.78	64.44	64.44	71.11	51.11	60.00	57.78	60.00	55.56	73.33	71.11	93.33	86.67
xstest-should-respond	53.60	77.60	57.60	57.60	50.00	58.80	72.40	63.20	84.40	73.60	90.40	85.60	86.40
hep-rust	48.78	53.05	64.02	65.85	51.22	68.29	63.41	62.80	79.27	74.39	85.98	74.39	90.24
hep-js	44.51	60.98	63.41	68.29	56.71	71.34	66.46	69.51	81.71	84.76	82.93	87.20	93.29
alpacaeval-length	62.61	70.43	69.94	69.44	69.94	62.11	63.11	68.82	53.42	66.83	54.41	71.68	86.71
llmbar-natural	58.00	59.00	54.00	69.00	64.00	76.00	51.00	71.00	62.00	73.00	74.00	88.00	82.00

Table 9: Reward Bench Performance Across Different Levels

Model	Levels	RM-Bench chat	RM-Bench code	RM-Bench math	RM-Bench safety response	RM-Bench safety refuse
Llama-1B	level 1	48.06	54.39	46.31	31.85	38.73
	level 2	64.34	55.26	48.58	69.43	53.52
	level 3	60.47	50.44	41.59	61.78	71.13
	mean	57.62	53.36	45.49	54.35	54.46
Llama-1B-Instruct	level 1	51.16	51.32	49.53	71.34	67.61
	level 2	61.24	53.51	47.45	68.15	77.11
	level 3	60.47	49.56	45.75	73.89	63.38
	mean	57.62	51.46	47.57	71.13	69.37
Llama-3B	level 1	54.26	51.75	47.26	68.15	7.04
	level 2	33.33	52.19	46.12	78.34	37.32
	level 3	33.33	49.12	45.75	36.94	55.28
	mean	40.31	51.02	46.38	61.15	33.22
Llama3b-Instruct	level 1	56.59	50.88	50.09	87.90	55.28
	level 2	44.96	55.26	48.02	86.62	60.56
	level 3	52.71	49.56	47.45	94.27	76.76
	mean	51.42	51.90	48.52	89.60	64.20
Llama-8B	level 1	54.26	53.51	48.02	99.36	2.46
	level 2	56.59	56.58	51.98	83.44	29.58
	level 3	50.39	51.75	47.26	64.33	63.38
	mean	53.75	53.95	49.09	82.38	31.81
Llama-8B-Instruct	level 1	65.12	55.70	50.28	56.05	75.00
	level 2	36.43	55.70	49.72	96.18	30.28
	level 3	50.39	53.51	46.12	64.33	87.68
	mean	50.65	54.97	48.71	72.19	64.32
Mistral-7b	level 1	50.39	46.49	52.17	96.18	20.42
	level 2	61.24	53.51	49.34	44.59	89.44
	level 3	51.94	46.49	43.10	75.16	84.15
	mean	54.52	48.83	48.20	71.97	64.67
Mistral-7b-Instruct	level 1	44.19	50.88	52.55	61.78	96.48
	level 2	58.91	52.63	55.39	39.49	81.69
	level 3	58.91	53.95	48.20	52.87	96.83
	mean	54.01	52.49	52.05	51.38	91.67
Qwen2.5-3B	level 1	65.89	48.68	54.06	95.54	94.01
	level 2	58.14	52.19	51.23	82.80	88.03
	level 3	48.84	50.44	46.12	94.90	49.65
	mean	57.62	50.44	50.47	91.08	77.23
Qwen2.5-3B-Instruct	level 1	72.87	51.32	60.87	46.50	63.38
	level 2	55.04	53.07	57.66	31.85	90.49
	level 3	55.04	54.82	50.47	84.71	96.83
	mean	60.98	53.07	56.33	54.35	83.57
Qwen2.5-7B	level 1	72.87	56.58	56.14	100.00	100.00
	level 2	47.29	56.58	54.06	96.82	94.72
	level 3	51.16	53.07	47.64	94.27	100.00
	mean	57.11	55.41	52.61	97.03	98.24
Qwen2.5-7B-Inst	level 1	80.62	58.33	62.19	91.08	100.00
	level 2	61.24	58.33	62.00	85.99	96.83
	level 3	64.34	55.26	50.28	63.06	100.00
	mean	68.73	57.31	58.16	80.04	98.94
SKYWORK-8b-reward	level 1	86.04	53.07	62.38	94.90	97.18
	level 2	55.04	53.51	65.41	82.80	98.94
	level 3	41.09	48.25	66.16	87.26	100.00
	mean	60.72	51.61	64.65	88.32	98.60

Table 10: Performance of various models, across different levels on RM-Bench