Self-Play Preference Optimization for Language Model Alignment

Anonymous Authors[1]

Abstract

Traditional reinforcement learning from human feedback (RLHF) approaches relying on parametric models like the Bradley-Terry model fall short in capturing the intransitivity and irrationality in human preferences. Recent advancements suggest that directly working with preference probabilities can yield a more accurate reflection of human preferences, enabling more flexible and accurate language model alignment. In this paper, we propose a self-play-based method for language model alignment, which treats the problem as a constant-sum two-player game aimed at identifying the Nash equilibrium policy. Our approach, dubbed *Self-Play Preference Optimization* (SPPO), approximates the Nash equilibrium through iterative policy updates and enjoys a theoretical convergence guarantee. Our method can effectively increase the log-likelihood of the chosen response and decrease that of the rejected response, which cannot be trivially achieved by symmetric pairwise loss such as Direct Preference Optimization (DPO) and Identity Preference Optimization (IPO). In our experiments, using only 60k prompts (without responses) from the UltraFeedback dataset and without any prompt augmentation, by leveraging a pre-trained preference model PairRM with only 0.4B parameters, SPPO can obtain a model from fine-tuning Mistral-7B-Instruct-v0.2 that achieves the state-of-the-art length-controlled win-rate of 28.53% against GPT-4-Turbo on AlpacaEval 2.0. It also outperforms the (iterative) DPO and IPO on MT-Bench and the Open LLM Leaderboard. Notably, the strong performance of SPPO is achieved without additional external supervision (e.g., responses, preferences, etc.) from GPT-4 or other stronger language models.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

1 Introduction

Large Language Models (LLMs) have demonstrated impressive capabilities, yet they face challenges in ensuring reliability, safety, and ethical alignment. Reinforcement Learning from Human Feedback (RLHF) offers a solution by fine-tuning models to align with human preferences. Traditional RLHF methods (Christiano et al., 2017; Ouyang et al., 2022) rely on reward models to guide this process, but they often fall short of capturing the complexities of human behavior.

Recent research highlights the limitations of parametric preference models like Bradley & Terry (1952), which assume consistent and transitive human preferences. Instead, studies suggest that human preferences can be inconsistent and influenced by various factors, challenging the effectiveness of these models(Tversky, 1969).

To address these issues, researchers have begun exploring more flexible algorithms that directly handle preference probabilities. Emerging approaches, such as Self-play Preference Optimization (SPO, Swamy et al., 2024), aim to identify optimal policies through self-play mechanisms. These methods offer potential improvements but require significant adaptation for large-scale LLM fine-tuning.

In this paper, we introduce Self-Play Preference Optimization (SPPO), a new self-play algorithm designed to solve the two-player constant-sum game for LLM alignment. SPPO utilizes an exponential weight update algorithm within a self-play framework, where policies are fine-tuned on synthetic data generated by the model itself. Our contributions include a provably convergent SPPO algorithm for LLM alignment, optimizing a simple loss function. Comparisons with state-of-the-art methods like DPO, IPO, and KTO, demonstrate SPPO's superior performance on various benchmarks. Empirical evidence shows that SPPO enhances the Mistral-7B-Instruct-v0.2 model, achieving significant improvements without external supervision from stronger models like GPT-4. Our findings suggest that SPPO provides a robust and scalable solution for aligning large language models with human preferences.

2 Preliminaries

We consider the preference learning scenario as follows. Given a text sequence (commonly referred to as prompt) $\mathbf{x} = [x_1, x_2, \dots]$, two text sequences $\mathbf{y} = [y_1, y_2, \dots]$ and

$\mathbf{y}'$ are generated as responses to the prompt $\mathbf{x}$. An autoregressive language model π given the prompt $\mathbf{x}$ can generate responses $\mathbf{y}$ following the probability decomposition

$$\pi(\mathbf{y}|\mathbf{x}) = \prod_{i=1}^{N} \pi(y_i|\mathbf{x}, \mathbf{y}_{<i}).$$

Given the prompt $\mathbf{x}$ and two responses $\mathbf{y}$ and $\mathbf{y}'$, a preference oracle (either a human annotator or a language model) will provide preference feedback $o(\mathbf{y} \succ \mathbf{y}'|\mathbf{x}) \in \{0, 1\}$ indicating whether $\mathbf{y}$ is preferred over $\mathbf{y}'$. We denote $\mathbb{P}(\mathbf{y} \succ \mathbf{y}'|\mathbf{x}) = \mathbb{E}[o(\mathbf{y} \succ \mathbf{y}'|\mathbf{x})]$ as the probability of $\mathbf{y}$ "winning the duel" over $\mathbf{y}'$. The KL divergence of two probability distributions of density p and q is defined as $\mathrm{KL}(p\|q) = \mathbb{E}_{\mathbf{y}\sim p(\mathbf{y})}\Big[\log \frac{p(\mathbf{y})}{q(\mathbf{y})}\Big]$.

2.1 RLHF with General Preference

Following Wang et al. (2024); Munos et al. (2023), we aim to establish RLHF methods without a reward model, as the human preference can be non-transitive (Tversky, 1969). Under a general preference oracle $\mathbb{P}(\mathbf{y} \succ \mathbf{y}'|\mathbf{x})$, we follow Dudík et al. (2015) and aim to identify the *von Neumann winner*. More specifically, the von Neumann winner π^* is the (symmetric) Nash equilibrium of the following two-player constant-sum game:

$$(\pi^*, \pi^*) = \arg\max_{\pi}\min_{\pi'} \mathbb{E}_{\mathbf{x}\sim\mathcal{X}}\Big[\mathbb{E}_{\mathbf{y}\sim\pi(\cdot|\mathbf{x}),\mathbf{y}'\sim\pi'(\cdot|\mathbf{x})}\big[\mathbb{P}(\mathbf{y} \succ \mathbf{y}'|\mathbf{x})\big]\Big]. \quad (2.1)$$

In addition, we define the winning probability of one response $\mathbf{y}$ against a distribution of responses π as

$$\mathbb{P}(\mathbf{y} \succ \pi|\mathbf{x}) = \mathbb{E}_{\mathbf{y}'\sim\pi(\cdot|\mathbf{x})}[\mathbb{P}(\mathbf{y} \succ \mathbf{y}'|\mathbf{x})],$$

and the winning probability of one policy π against another policy π' as

$$\mathbb{P}(\pi \succ \pi'|\mathbf{x}) = \mathbb{E}_{\mathbf{y}\sim\pi(\cdot|\mathbf{x})}\mathbb{E}_{\mathbf{y}'\sim\pi'(\cdot|\mathbf{x})}[\mathbb{P}(\mathbf{y} \succ \mathbf{y}'|\mathbf{x})].$$

Furthermore, we define $\mathbb{P}(\pi \succ \pi') = \mathbb{E}_{\mathbf{x}\sim\mathcal{X}}[\mathbb{P}(\pi \succ \pi'|\mathbf{x})]$, where $\mathbf{x}$ is a prompt drawn from the prompt distribution $\mathcal{X}$. The two-player constant-sum game (2.1) can be simplified as

$$(\pi^*, \pi^*) = \arg\max_{\pi}\min_{\pi'} \mathbb{P}(\pi \succ \pi').$$

3 Self-Play Preference Optimization (SPPO)

In this section, we introduce the Self-Play Preference Optimization (SPPO) algorithm, derived from the following theoretical framework.

3.1 Theoretical Framework

There are well-known algorithms to approximately solve the Nash equilibrium in a constant-sum two-player game. In this work, we follow Freund & Schapire (1999) to establish an iterative framework that can asymptotically converge to the optimal policy on average. We start with a theoretical framework that conceptually solves the two-player game for $t = 1, 2, \ldots$ as follows:

$$\pi_{t+1}(\mathbf{y}|\mathbf{x}) \propto \pi_t(\mathbf{y}|\mathbf{x}) \exp(\eta\mathbb{P}(\mathbf{y} \succ \pi_t|\mathbf{x})). \quad (3.2)$$

(3.2) is an iterative framework that relies on the multiplicative weight update in each round t and enjoys a clear structure. Initially, we have a base policy π_1 usually from some supervised fine-tuned model. In each round, the updated policy π_{t+1} is obtained from the reference policy π_t following the multiplicative weight update. More specifically, a response $\mathbf{y}$ should have a higher probability weight if it has a higher average advantage over the current policy π_t.

Equivalently, (3.2) can be written as

$$\pi_{t+1}(\mathbf{y}|\mathbf{x}) = \frac{\pi_t(\mathbf{y}|\mathbf{x}) \exp\big(\eta\mathbb{P}(\mathbf{y} \succ \pi_t|\mathbf{x})\big)}{Z_{\pi_t}(\mathbf{x})}, \quad (3.3)$$

where $Z_{\pi_t}(\mathbf{x}) = \sum_{\mathbf{y}} \pi_t(\mathbf{y}|\mathbf{x}) \exp\big(\eta\mathbb{P}(\mathbf{y} \succ \pi_t|\mathbf{x})\big)$ is the normalizing factor (a.k.a., the partition function). For any fixed $\mathbf{x}$ and $\mathbf{y}$, the ideal update policy π_{t+1} should satisfy the following equation:

$$\log\left(\frac{\pi_{t+1}(\mathbf{y}|\mathbf{x})}{\pi_t(\mathbf{y}|\mathbf{x})}\right) = \eta \cdot \mathbb{P}(\mathbf{y} \succ \pi_t|\mathbf{x}) - \log Z_{\pi_t}(\mathbf{x}). \quad (3.4)$$

Unlike the pair-wise design in DPO or IPO that cancels the log normalizing factor $\log Z_{\pi_t}(\mathbf{x})$ by differentiating (3.4) between $\mathbf{y}$ and $\mathbf{y}'$, we choose to approximate (3.4) directly in terms of L_2 distance:

$$\pi_{t+1} = \operatorname*{argmin}_{\pi} \mathbb{E}_{\mathbf{x}\sim\mathcal{X},\mathbf{y}\sim\pi_t(\cdot|\mathbf{x})} \left(\log\left(\frac{\pi(\mathbf{y}|\mathbf{x})}{\pi_t(\mathbf{y}|\mathbf{x})}\right) - \Big(\eta\mathbb{P}(\mathbf{y} \succ \pi_t|\mathbf{x}) - \log Z_{\pi_t}(\mathbf{x})\Big)\right)^2. \quad (3.5)$$

Estimation of the Probability The optimization objective (3.5) can be approximated with finite samples. We choose to sample K responses $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_K \sim \pi_t(\cdot|\mathbf{x})$ for each prompt $\mathbf{x}$, and denote the empirical distribution by $\widehat{\pi}_t^K$. The finite-sample optimization problem can be approximated as

$$\pi_{t+1} = \operatorname*{argmin}_{\pi} \mathbb{E}_{\mathbf{x}\sim\mathcal{X},\mathbf{y}\sim\pi_t(\cdot|\mathbf{x})} \left(\log\left(\frac{\pi(\mathbf{y}|\mathbf{x})}{\pi_t(\mathbf{y}|\mathbf{x})}\right) - \Big(\eta\mathbb{P}(\mathbf{y} \succ \widehat{\pi}_t^K|\mathbf{x}) - \log Z_{\widehat{\pi}_t^K}(\mathbf{x})\Big)\right)^2. \quad (3.6)$$

Specifically, $\mathbb{P}(\mathbf{y} \succ \widehat{\pi}_t^K|\mathbf{x}) = \sum_{k=1}^K \mathbb{P}(\mathbf{y} \succ \mathbf{y}_k|\mathbf{x})/K$ and $Z_{\widehat{\pi}_t^K}(\mathbf{x}) = \mathbb{E}_{\mathbf{y}\sim\pi_t(\cdot|\mathbf{x})}[\exp(\eta\mathbb{P}(\mathbf{y} \succ \widehat{\pi}_t^K|\mathbf{x}))]$. $Z_{\widehat{\pi}_t^K}(\mathbf{x})$, treated as an expectation, can be further estimated by B

Algorithm 1 `Self-Play Preference Optimization (SPPO)`

1: **input**: base policy $\pi_{\boldsymbol{\theta}_1}$, preference oracle $\mathbb{P}$, learning rate η, number of generated samples K.
2: **for** $t = 1, 2, \ldots$ **do**
3: Generate synthetic responses by sampling $\mathbf{x} \sim \mathcal{X}$ and $\mathbf{y}_{1:K} \sim \pi_t(\cdot|\mathbf{x})$.
4: Annotate the win-rate $\mathbb{P}(\mathbf{y}_k \succ \mathbf{y}_{k'}|\mathbf{x}), \forall k, k' \in [K]$.
5: Select responses from $\mathbf{y}_{1:K}$ to form dataset $\mathcal{D}_t = \{(\mathbf{x}_i, \mathbf{y}_i, \widehat{P}(\mathbf{y}_i \succ \pi_t|\mathbf{x}_i))\}_{i\in[N]}$.
6: Optimize $\pi_{\boldsymbol{\theta}_{t+1}}$ according to (3.7):

$$\boldsymbol{\theta}_{t+1} \leftarrow \operatorname*{argmin}_{\boldsymbol{\theta}} \mathbb{E}_{(\mathbf{x},\mathbf{y},\widehat{P}(\mathbf{y}\succ\pi_t|\mathbf{x}))\sim\mathcal{D}_t}\left(\log\left(\frac{\pi_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x})}{\pi_t(\mathbf{y}|\mathbf{x})}\right) - \eta\left(\widehat{P}(\mathbf{y} \succ \pi_t|\mathbf{x}) - \frac{1}{2}\right)\right)^2. \tag{3.1}$$

7: **end for**

new samples with in total $O(KB)$ queries of the preference oracle $\mathbb{P}$. (3.6) is an efficiently tractable optimization problem. Informally speaking, when $K \to \infty$, (3.6) will recover (3.5). We have the following guarantee on the convergence of (3.5):

Theorem 3.1. *Assume the optimization problem (3.5) is realizable. Denote π_t as the policy obtained via (3.5) and the mixture policy $\bar{\pi}_T = \frac{1}{T}\sum_{t=1}^{T}\pi_t$. By setting $\eta = \Theta(1/\sqrt{T})$, we have that*

$$\max_{\pi}\big[\mathbb{P}(\pi \succ \bar{\pi}_T)\big] - \min_{\pi}\big[\mathbb{P}(\pi \prec \bar{\pi}_T)\big] = O(1/\sqrt{T}).$$

Theorem 3.1 characterizes the convergence rate of the average policy across the time horizon T towards the Nash equilibrium, in terms of the duality gap. The proof is based on Theorem 1 in Freund & Schapire (1999) with slight modification. For completeness, we include the proof in Appendix E.

Alternatively, we can avoid estimating $\log Z_{\widehat{\pi}_t^K}(\mathbf{x})$ by replacing it simply with $\eta/2$[1] in (3.6) to obtain a more clear objective:

$$\pi_{t+1} = \operatorname*{argmin}_{\pi} \mathbb{E}_{\mathbf{x}\sim\mathcal{X},\mathbf{y}\sim\pi_t(\cdot|\mathbf{x})} \left(\log\left(\frac{\pi(\mathbf{y}|\mathbf{x})}{\pi_t(\mathbf{y}|\mathbf{x})}\right) - \eta\left(\mathbb{P}(\mathbf{y} \succ \widehat{\pi}_t^K|\mathbf{x}) - \frac{1}{2}\right)\right)^2. \tag{3.7}$$

Intuitively, if a tie occurs (i.e., $\mathbb{P}(\mathbf{y} \succ \widehat{\pi}_t^K|\mathbf{x}) = 1/2$), we prefer the model does not update weight at $\mathbf{y}$. If $\mathbf{y}$ wins over $\widehat{\pi}_t^K$ on average (i.e., $\mathbb{P}(\mathbf{y} \succ \widehat{\pi}_t^K|\mathbf{x}) > 1/2$), then we increase the probability density at $\mathbf{y}$ to employ the advantage of $\mathbf{y}$ over $\widehat{\pi}_t^K$. In our experiments, we choose to minimize the objective (3.7).

3.2 The SPPO Algorithm

Based on the aforementioned theoretical framework, we propose the *Self-Play Preference Optimization* algorithm in Algorithm 1.

In each round t, Algorithm 1 will first generate K responses $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_K$ according to $\pi_t(\cdot|\mathbf{x})$ for each prompt $\mathbf{x}$ (Line 3). Then, the preference oracle $\mathbb{P}$ will be queried to calculate the win rate among the K responses (Line 4). At Line 5, certain criteria can be applied to determine which response should be kept in the constructed dataset $\mathcal{D}_t$ and construct the prompt-response-probability triplet $(\mathbf{x}, \mathbf{y}, \widehat{P}(\mathbf{y} \succ \pi_t|\mathbf{x}))$. We will discuss the design choices later in Section 4. One straightforward design choice is to include all K responses into $\mathcal{D}_t$ and each $\widehat{P}(\mathbf{y}_i \succ \pi_t|\mathbf{x})$ is estimated by comparing $\mathbf{y}_i$ to all K responses. In total, $O(K^2)$ queries will be made. Then the algorithm will optimize (3.7) on the dataset $\mathcal{D}_t$ (Line 6).

4 Experiments

We conduct extensive experiments to show the performance of our method and compare it with other baselines.

4.1 Experiment Setup

We briefly summarize our experiment setup as below. For a full description of our experiment setup, see Section C.

Base Model and Datasets: We follow Snorkel's experimental setup, using Mistral-7B-Instruct-v0.2 as our base model and Ultrafeedback for prompts. We split the dataset into three portions to avoid overfitting and ensure fair comparison with Snorkel.

Preference Model: We use PairRM, a 0.4B pair-wise preference model based on DeBERTA-V3, trained on high-quality human-preference datasets. PairRM outputs a "relative reward" to balance accuracy and efficiency, following Snorkel's methodology.

Response Generation and Selection: We sample $K = 5$ responses per prompt with top $p = 1.0$ and temperature 1.0. We select the responses with the highest and lowest PairRM scores as the winning and losing responses respectively.

Baselines and Benchmarks: We evaluate Mistral-7B-Instruct-v0.2, Snorkel, iterative DPO and IPO, and Self-rewarding LM. Benchmarks include AlpacaEval 2.0, MT-

[1]Assuming the winning probability between any pair is a fair coin toss, when $K \to \infty$, we can show that indeed $Z_{\widehat{\pi}_t^K}(\mathbf{x}) \to e^{\eta/2}$.

Table 1: AlpacaEval 2.0 evaluation of various models (detailed in Baselines) in terms of both normal and length-controlled (LC) win rates in percentage (%). Mistral-7B stands for Mistral-7B-instruct-v0.2; Snorkel stands for Snorkel (Mistral-PairRM-DPO); bo16 stands for best-of-16. SPPO demonstrates steady performance gains across iterations and outperforms other baselines which show a tendency to produce longer responses. Additionally, re-ranking with the PairRM reward model (best-of-16) at test time consistently enhances the performance across all models.

Model	AlpacaEval 2.0		
	LC Win Rate	Win Rate	Avg. Len
Mistral-7B	17.11	14.72	1676
Mistral-7B (bo16)	22.45	17.94	1529
Snorkel	26.39	30.22	2736
Snorkel (bo16)	29.97	34.86	2616
Self-Rewarding Iter1	-	9.94	1092
Self-Rewarding Iter2	-	15.38	1552
Self-Rewarding Iter3	-	20.44	2552
DPO Iter1	23.81	20.44	1723
DPO Iter2	24.23	24.46	2028
DPO Iter3	22.30	23.39	2189
IPO Iter1	23.78	20.77	1693
IPO Iter2	21.08	23.38	2660
IPO Iter3	20.06	22.47	2760
SPPO Iter1	24.79$_{(+7.69)}$	23.51$_{(+8.79)}$	1855
SPPO Iter2	26.89$_{(+2.10)}$	27.62$_{(+4.11)}$	2019
SPPO Iter3	**28.53**$_{(+1.64)}$	**31.02**$_{(+3.40)}$	2163
SPPO Iter1 (bo16)	28.71$_{(+6.26)}$	27.77$_{(+9.83)}$	1901
SPPO Iter2 (bo16)	31.23$_{(+2.52)}$	32.12$_{(+4.35)}$	2035
SPPO Iter3 (bo16)	**32.13**$_{(+0.9)}$	**34.94**$_{(+2.82)}$	2174

Bench, and the Open LLM Leaderboard, covering various aspects of language model evaluation.

4.2 Experimental Results

In the assessment of AI chatbots, human evaluation remains the benchmark for quality and accuracy (Askell et al., 2021; Ouyang et al., 2022). However, due to its limitations in scalability and reproducibility, we explore the alternative approach of using the advanced capabilities of GPT-4 (OpenAI et al., 2023) as an automatic evaluation tool. We conduct GPT-4-based automatic evaluation on AlpacaEval 2.0 (Li et al., 2023b) and MT-Bench (Zheng et al., 2023) to measure the chatbot capability of our model. Due to the space limit, we only report the results on AlpacaEval 2.0 in the following and postpone other results including ablation studies to the appendix.

Table 1 (AlpacaEval 2.0) shows the win rate over the GPT-4-Turbo baseline of different models on 805 prompts. We also include one column indicating the length-controlled win rate, and one column on the average length of each model, to account for the tendency of the LLM-based judge to favor longer sequence outputs — an issue colloquially termed the "reward hacking" phenomenon. According to the table, SPPO Iter3 has the highest win rate, 28.52% for the length-controlled version, and 31.02% for the overall win rate. The performance gains over previous iterations are 7.69% (Mistral-7B-Instruct → Iter1), 2.10% (Iter1 → Iter2), and 1.64% (Iter2 → Iter3), respectively, indicating steady improvements across iterations. Additionally, the data indicates that SPPO achieves superior performance compared to the iterative variants of DPO and IPO. The length-controlled win rate for SPPO reaches 28.53%, outperforming the DPO's best rate of 26.39% (by Snorkel) and IPO's rate of 25.45% . Notably, while DPO and IPO training tend to significantly increase the average output length—2736 and 2654, respectively—SPPO shows a more moderate length increase, moving from 1676 in the base model to 2163 at the third iteration. We find that re-ranking with the preference model at test time can consistently improve the performance of base models (Mistral-7B-Instruct-v0.2), DPO (Snorkel), and SPPO (Iter3) by 5.34%, 3.57%, and 3.6%, respectively. Notably, this shows that while SPPO significantly enhances model alignment using PairRM-0.4B as the sole external supervision, it has not resulted in over-optimization against the preference model (Gao et al., 2023). Future work will explore further improvements in model alignment, potentially through additional iterations beyond the current three (following Snorkel's methodology).

Table 2: AlpacaEval 2.0 leaderboard results of both normal and length-controlled (LC) win rates in percentage (%). Snorkel stands for Snorkel (Mistral-PairRM-DPO). Our SPPO model outperforms many competing models trained on proprietary alignment data (e.g., Claude 2, Gemini Pro, & Llama 3 8B Instruct). With test-time reranking, SPPO Iter3 (best-of-16) is even competitive to GPT-4 0613 and Llama 3 70B Instruct.

Model	AlpacaEval 2.0	
	LC. Win Rate	Win Rate
GPT-4 Turbo	50.0	50.0
Claude 3 Opus	40.5	29.1
GPT-4 0314	35.3	22.1
Llama 3 70B Instruct	34.4	33.2
SPPO Iter3 (best-of-16)	32.1	34.9
GPT-4 0613	30.2	15.8
Snorkel (best-of-16)	30.0	34.9
Mistral Medium	28.6	21.9
SPPO Iter3	28.5	31.0
Claude 2	28.2	17.2
Snorkel	26.4	30.2
Gemini Pro	24.4	18.2
Mistral 8×7B v0.1	23.7	18.1
Llama 3 8B Instruct	22.9	22.6
GPT-3.5 Turbo 0613	22.7	14.1
Vicuna 33B v1.3	17.6	12.7

References

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

Azar, M. G., Rowland, M., Piot, B., Guo, D., Calandriello, D., Valko, M., and Munos, R. A general theoretical paradigm to understand learning from human preferences. *arXiv preprint arXiv:2310.12036*, 2023.

Beeching, E., Fourrier, C., Habib, N., Han, S., Lambert, N., Rajani, N., Sanseviero, O., Tunstall, L., and Wolf, T. Open llm leaderboard. *Hugging Face*, 2023a.

Beeching, E., Fourrier, C., Habib, N., Han, S., Lambert, N., Rajani, N., Sanseviero, O., Tunstall, L., and Wolf, T. Open llm leaderboard. `https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard`, 2023b.

Bradley, R. A. and Terry, M. E. Rank Analysis of Incomplete Block Designs: I. The Method of Paired Comparisons. *Biometrika*, 39(3/4):324–345, 1952. ISSN 0006-3444. doi: 10.2307/2334029.

Calandriello, D., Guo, D., Munos, R., Rowland, M., Tang, Y., Pires, B. A., Richemond, P. H., Lan, C. L., Valko, M., Liu, T., et al. Human alignment of large language models through online preference optimisation. *arXiv preprint arXiv:2403.08635*, 2024.

Chen, Z., Deng, Y., Yuan, H., Ji, K., and Gu, Q. Self-play fine-tuning converts weak language models to strong language models. *arXiv preprint arXiv:2401.01335*, 2024.

Christiano, P. F., Leike, J., Brown, T., Martic, M., Legg, S., and Amodei, D. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

Clark, P., Cowhey, I., Etzioni, O., Khot, T., Sabharwal, A., Schoenick, C., and Tafjord, O. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Cui, G., Yuan, L., Ding, N., Yao, G., Zhu, W., Ni, Y., Xie, G., Liu, Z., and Sun, M. Ultrafeedback: Boosting language models with high-quality feedback. *arXiv preprint arXiv:2310.01377*, 2023.

Dubois, Y., Galambosi, B., Liang, P., and Hashimoto, T. B. Length-controlled alpacaeval: A simple way to debias automatic evaluators. *arXiv preprint arXiv:2404.04475*, 2024a.

Dubois, Y., Li, C. X., Taori, R., Zhang, T., Gulrajani, I., Ba, J., Guestrin, C., Liang, P. S., and Hashimoto, T. B. Alpacafarm: A simulation framework for methods that learn from human feedback. *Advances in Neural Information Processing Systems*, 36, 2024b.

Dudík, M., Hofmann, K., Schapire, R. E., Slivkins, A., and Zoghi, M. Contextual dueling bandits. In *Conference on Learning Theory*, pp. 563–587. PMLR, 2015.

Ethayarajh, K., Xu, W., Muennighoff, N., Jurafsky, D., and Kiela, D. Kto: Model alignment as prospect theoretic optimization. *arXiv preprint arXiv:2402.01306*, 2024.

Freund, Y. and Schapire, R. E. Adaptive game playing using multiplicative weights. *Games and Economic Behavior*, 29(1-2):79–103, 1999.

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

Gao, Z., Chang, J. D., Zhan, W., Oertell, O., Swamy, G., Brantley, K., Joachims, T., Bagnell, J. A., Lee, J. D., and Sun, W. Rebel: Reinforcement learning via regressing relative rewards. *arXiv preprint arXiv:2404.16767*, 2024.

He, P., Gao, J., and Chen, W. Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing, 2021.

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

Hong, J., Lee, N., and Thorne, J. Reference-free monolithic preference optimization with odds ratio. *arXiv preprint arXiv:2403.07691*, 2024.

Ji, K., He, J., and Gu, Q. Reinforcement learning from human feedback with active queries. *arXiv preprint arXiv:2402.09401*, 2024.

Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., Casas, D. d. l., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023a.

Jiang, D., Ren, X., and Lin, B. Y. Llm-blender: Ensembling large language models with pairwise ranking and generative fusion. *arXiv preprint arXiv:2306.02561*, 2023b.

Li, J., Sun, S., Yuan, W., Fan, R.-Z., Zhao, H., and Liu, P. Generative judge for evaluating alignment. *arXiv preprint arXiv:2310.05470*, 2023a.

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

Lin, S., Hilton, J., and Evans, O. Truthfulqa: Measuring how models mimic human falsehoods. *arXiv preprint arXiv:2109.07958*, 2021.

Liu, T., Zhao, Y., Joshi, R., Khalman, M., Saleh, M., Liu, P. J., and Liu, J. Statistical rejection sampling improves preference optimization. *arXiv preprint arXiv:2309.06657*, 2023.

Munos, R., Valko, M., Calandriello, D., Azar, M. G., Rowland, M., Guo, Z. D., Tang, Y., Geist, M., Mesnard, T., Michi, A., et al. Nash learning from human feedback. *arXiv preprint arXiv:2312.00886*, 2023.

OpenAI, Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

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

Pal, A., Karkhanis, D., Dooley, S., Roberts, M., Naidu, S., and White, C. Smaug: Fixing failure modes of preference optimisation with dpo-positive. *arXiv preprint arXiv:2402.13228*, 2024.

Rafailov, R., Sharma, A., Mitchell, E., Manning, C. D., Ermon, S., and Finn, C. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36, 2024.

Rosset, C., Cheng, C.-A., Mitra, A., Santacroce, M., Awadallah, A., and Xie, T. Direct nash optimization: Teaching language models to self-improve with general preferences. *arXiv preprint arXiv:2404.03715*, 2024.

Sakaguchi, K., Bras, R. L., Bhagavatula, C., and Choi, Y. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.

Singh, A., Co-Reyes, J. D., Agarwal, R., Anand, A., Patil, P., Liu, P. J., Harrison, J., Lee, J., Xu, K., Parisi, A., et al. Beyond human data: Scaling self-training for problem-solving with language models. *arXiv preprint arXiv:2312.06585*, 2023.

Swamy, G., Dann, C., Kidambi, R., Wu, Z. S., and Agarwal, A. A minimaximalist approach to reinforcement learning from human feedback. *arXiv preprint arXiv:2401.04056*, 2024.

Tversky, A. Intransitivity of preferences. *Psychological review*, 76(1):31, 1969.

Wang, Y., Liu, Q., and Jin, C. Is rlhf more difficult than standard rl? a theoretical perspective. *Advances in Neural Information Processing Systems*, 36, 2024.

Xiong, W., Dong, H., Ye, C., Zhong, H., Jiang, N., and Zhang, T. Gibbs sampling from human feedback: A provable kl-constrained framework for rlhf. *arXiv preprint arXiv:2312.11456*, 2023.

Xu, J., Lee, A., Sukhbaatar, S., and Weston, J. Some things are more cringe than others: Preference optimization with the pairwise cringe loss. *arXiv preprint arXiv:2312.16682*, 2023.

Ye, C., Xiong, W., Zhang, Y., Jiang, N., and Zhang, T. A theoretical analysis of nash learning from human feedback under general kl-regularized preference. *arXiv preprint arXiv:2402.07314*, 2024.

Yuan, W., Pang, R. Y., Cho, K., Sukhbaatar, S., Xu, J., and Weston, J. Self-rewarding language models. *arXiv preprint arXiv:2401.10020*, 2024.

Zellers, R., Holtzman, A., Bisk, Y., Farhadi, A., and Choi, Y. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*, 2019.

Zhao, Y., Joshi, R., Liu, T., Khalman, M., Saleh, M., and Liu, P. J. Slic-hf: Sequence likelihood calibration with human feedback. *arXiv preprint arXiv:2305.10425*, 2023.

Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E., et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36, 2023.

Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E., et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36, 2024.

Zhu, B., Jiao, J., and Jordan, M. I. Principled reinforcement learning with human feedback from pairwise or k-wise comparisons. *arXiv preprint arXiv:2301.11270*, 2023.

A Related Work

RLHF with Explicit/Implicit Reward Model Originally, reinforcement learning from human feedback (RLHF) was proposed by Christiano et al. (2017) as a methodology that first learns a reward model reflecting human preferences and then uses reinforcement learning algorithms to maximize the reward. This methodology is applied by Ouyang et al. (2022) to fine-tune instruction-following large language models and leads to the popular ChatGPT.

The reward model in the works mentioned above assumes a parametric model such as the Bradley-Terry model (Bradley & Terry, 1952), which assigns a "score" representing how preferred a given response is. More recently, Rafailov et al. (2024) proposed to instead directly solve the closed-form solution of such a score implied by the Bradley-Terry model. The Direct Policy Optimization (DPO) method is claimed to be more efficient and stable, yet, still implicitly assumes such a reward model that specifies the "score". In a similar spirit, Zhao et al. (2023) proposed to calibrate the score so that the score of the winner in comparison has a margin over the score of the loser, and induces a different SLic loss. Similarly, Ethayarajh et al. (2024) derived a different loss function (called KTO) from the Kahneman-Tversky human utility function, which implicitly denotes a score of the given response. Liu et al. (2023) proposed Rejection Sampling Optimization (RSO) which utilizes a preference model to generate preference pairs with candidates sampled from the optimal policy; then preference optimization is applied on the sampled preference pairs. Hong et al. (2024) proposed Odds Ratio Preference Optimization (ORPO) algorithm that can perform supervised fine-tuning and preference alignment in one training session without maintaining an intermediate reference policy.

RLHF with General Preference Model Often, the human preference is not strictly transitive, and cannot be sufficiently represented by a single numerical score. Azar et al. (2023) proposed a general preference optimization objective based on the preference probability between a pair of responses instead of a score of a single response. They further propose a learning objective based on identity mapping of the preference probability called IPO (Preference Optimization with Identity mapping), which aims to maximize the current policy's expected winning probability over a given reference policy. Munos et al. (2023) formulated the RLHF problem with general preference as a two-player, constant-sum game, where each player is one policy that aims to maximize the probability of its response being preferred against its opponent. They aim to identify the Nash equilibrium policy of this game and propose a mirror-descent algorithm that guarantees the last-iterate convergence of a policy with tabular representations[2]. Wang et al. (2024) proposed to identify the Nash equilibrium policy for multi-step MDPs when a general preference model is present and shows that the problem can be reduced to a two-player zero-sum Markov game.

Theory of RLHF There is also a line of research to analyze RLHF and provide its theoretical guarantees. Zhu et al. (2023) studied the standard RLHF with separate reward-learning and model-tuning and proposed a pessimistic reward-learning process that provably learns a linear reward model. Wang et al. (2024) proposed a framework to reduce any RLHF problem with a reward model to a reward-based standard RL problem. Additionally, they proposed to identify the Nash equilibrium policy when a general preference model is present and show that the problem can be reduced to a two-player zero-sum Markov game. Xiong et al. (2023) studied the reverse-KL regularized contextual bandit for RLHF in different settings and proposed efficient algorithms with finite-sample theoretical guarantees. Ye et al. (2024) studied the theoretical learnability of the KL-regularized Nash-Learning from Human Feedback (NLHF) by considering both offline and online settings and proposed provably efficient algorithms. Ji et al. (2024) proposed an active-query-based proximal policy optimization algorithm with regret bounds and query complexity based on the problem dimension and the sub-optimality gap.

Self-Play Fine-Tuning Most works mentioned above (Rafailov et al., 2024; Zhao et al., 2023; Azar et al., 2023; Ethayarajh et al., 2024) consider one single optimization procedure starting from some reference policy. The same procedure may be applied repeatedly for multiple rounds in a self-play manner. In each round, new data are generated by the policy obtained in the last round; these new data are then used for training a new policy that can outperform the old policy.

The self-play fine-tuning can be applied to both scenarios with or without human preference data. For example, Singh et al. (2023) proposed an Expectation-Maximization (EM) framework where in each round, new data are generated and annotated with a reward score; the new policy is obtained by fine-tuning the policy on the data with a high reward. Chen et al. (2024) proposed a self-play framework to fine-tune the model in a supervised way. In each round, new preference pairs are synthesized by labeling the policy-generated responses as losers and the human-generated responses as winners. Then DPO is applied in each round to fine-tune another policy based on these synthesized preference data. Yuan et al. (2024) proposed Self-Rewarding Language Models, where the language model itself is used to annotate preference on its

[2]Due to the tabular representation, computing the normalizing factor is prohibitive and the algorithm is approximately executed by sampling one token instead of a full response.

own responses. Iterative DPO is applied to fine-tune language models on these annotated data. These works show iterative fine-tuning can significantly improve the performance.

Swamy et al. (2024) considered a more general multi-step Markov Decision Process (MDP) setting and proposed Self-play Preference Optimization (SPO), an RLHF framework that can utilize any no-regret online learning algorithm for preference-based policy optimization. They then instantiated their framework with the Soft Policy Iteration as an idealized variant of their algorithm, which reduces to the exponential weight update rule (3.2) when constrained to the bandit setting. The main difference is that they focus on the multi-round Markov decision process (MDP) in robotic or game tasks rather than on fine-tuning large language models and approximating the update using policy optimization methods such as PPO.

Concurrent to our work, Rosset et al. (2024) proposed the Direct Nash Optimization (DNO) algorithm based on the cross-entropy between true and predicted win rate gaps, and provided theoretical guarantees on the error of finite-sample approximations. However, their practical version still utilizes the iterative-DPO framework as in Xu et al. (2023) with the DPO loss instead of their own DNO loss. Notably, in their experiments, they added the GPT-4 generated responses as their "gold sample" into their fine-tuning data, and used GPT-4 as a judge to assign a numerical score to each response for preference pair construction. In sharp contrast, our work does not require use any strong external supervision besides a small-sized reward model. Another concurrent work (Gao et al., 2024) proposed REBEL, an iterative self-play framework via regressing the relative reward. When applied to the preference setting, it results a similar algorithm to our algorithm SPPO, except that SPPO approximate the log-partition factor $\log Z_{\pi_t}(\mathbf{x})$ with $\eta/2$ while REBEL regresses on the win rate difference (so that $\log Z_{\pi_t}(\mathbf{x})$ is cancelled). Additionally, Calandriello et al. (2024) pointed out that optimising the IPO loss (Azar et al., 2023) iteratively with self-play generated data is equivalent to finding the Nash equilibrium of the two-player game, and they proposed the IPO-MD algorithm based on this observation which generates data with a mixture policy similarly as the Nash-MD algorithm.

B Comparison with DPO, IPO, and KTO

In practice, we utilize mini-batches of more than 2 responses to estimate the win rate of a given response, while the DPO and IPO loss focus on a single pair of responses. When only a pair of responses $\mathbf{y}_w$ and $\mathbf{y}_l$ is available, we have the pair-wise symmetric loss based on the preference triplet $(\mathbf{x}, \mathbf{y}_w, \mathbf{y}_l)$ defined as:

$$\begin{aligned}\ell_{\text{SPPO}}(\mathbf{x}, \mathbf{y}_w, \mathbf{y}_l; \boldsymbol{\theta}; \pi_{\text{ref}}) := &\left(\log\left(\frac{\pi_{\boldsymbol{\theta}}(\mathbf{y}_w|\mathbf{x})}{\pi_{\text{ref}}(\mathbf{y}_w|\mathbf{x})}\right) - \eta\Big(\mathbb{P}(\mathbf{y}_w \succ \mathbf{y}_l|\mathbf{x}) - \frac{1}{2}\Big)\right)^2 \\ &+ \left(\log\left(\frac{\pi_{\boldsymbol{\theta}}(\mathbf{y}_l|\mathbf{x})}{\pi_{\text{ref}}(\mathbf{y}_l|\mathbf{x})}\right) - \eta\Big(\mathbb{P}(\mathbf{y}_w \prec \mathbf{y}_l|\mathbf{x}) - \frac{1}{2}\Big)\right)^2,\end{aligned} \tag{B.1}$$

where $\mathbb{P}(\mathbf{y}_w \succ \mathbf{y}_l|\mathbf{x})$ can be either a soft probability within $[0, 1]$ or a hard label 1 indicating $\mathbf{y}_w \succ \mathbf{y}_l$.

We now compare the SPPO loss to other baselines assuming a hard label $\mathbf{y}_w \succ \mathbf{y}_l$ is given. For the ease of comparison, let

$$a = \beta \log\left(\frac{\pi_{\boldsymbol{\theta}}(\mathbf{y}_w|\mathbf{x})}{\pi_{\text{ref}}(\mathbf{y}_w|\mathbf{x})}\right), b = \beta \log\left(\frac{\pi_{\boldsymbol{\theta}}(\mathbf{y}_l|\mathbf{x})}{\pi_{\text{ref}}(\mathbf{y}_l|\mathbf{x})}\right), c = \beta \text{KL}(\pi_{\boldsymbol{\theta}} \| \pi_{\text{ref}}),$$

then we have

$$\ell_{\text{DPO}}(\mathbf{y}_w, \mathbf{y}_l, \mathbf{x}) = -\log \sigma(a - b), \tag{B.2}$$

$$\ell_{\text{IPO}}(\mathbf{y}_w, \mathbf{y}_l, \mathbf{x}) = [(a - b) - 1]^2, \tag{B.3}$$

$$\ell_{\text{KTO}}(\mathbf{y}_w, \mathbf{y}_l, \mathbf{x}) = \sigma(-a + c) + \sigma(b - c) \text{ (simplified)}, \tag{B.4}$$

where $\sigma(x) = e^x/(e^x + 1)$ and the SPPO loss can be written as

$$\ell_{\text{SPPO}}(\mathbf{y}_w, \mathbf{y}_l, \mathbf{x}) = (a - 1/2)^2 + (b + 1/2)^2.$$

It can be seen that SPPO not only pushes the gap between a and b to be 1, but also attempts to push value of a to be close to $1/2$ and the value of b to be close to $-1/2$ such that $\pi_{\boldsymbol{\theta}}(\mathbf{y}_w|\mathbf{x}) > \pi_{\text{ref}}(\mathbf{y}_w|\mathbf{x})$ and $\pi_{\boldsymbol{\theta}}(\mathbf{y}_l|\mathbf{x}) < \pi_{\text{ref}}(\mathbf{y}_l|\mathbf{x})$. We believe this is particularly important: when there are plenty of preference pairs, DPO and IPO can ensure the policy will converge to the target policy, but when the preference pairs are scarce (e.g., one pair for each prompt), there is no guarantee that the estimated reward of the winner a will increase and the estimated reward of the loser b will decrease. Instead, only the reward gap between the winner and the loser (i.e., $a - b$) will increase. This phenomenon is observed by Pal et al. (2024) that DPO only drives the loser's likelihood to be small, but the winner's likelihood barely changes. We believe that fitting $\beta \log\left(\frac{\pi_{t+1}(\mathbf{y}|\mathbf{x})}{\pi_t(\mathbf{y}|\mathbf{x})}\right)$ directly to $\mathbb{P}(\mathbf{y} \succ \pi_t|\mathbf{x}) - 1/2$ is more effective than IPO which attempts to fit $\beta \log\left(\frac{\pi_{t+1}(\mathbf{y}_w|\mathbf{x})}{\pi_t(\mathbf{y}_w|\mathbf{x})}\right) - \beta \log\left(\frac{\pi_{t+1}(\mathbf{y}_l|\mathbf{x})}{\pi_t(\mathbf{y}_l|\mathbf{x})}\right)$ to $\mathbb{P}(\mathbf{y}_w \succ \pi_t|\mathbf{x}) - \mathbb{P}(\mathbf{y}_l \succ \pi_t|\mathbf{x})$. In addition, SPPO shares a similar spirit as KTO. The KTO loss pushes a to be large by minimizing $\sigma(-a + c)$ and pushes b to be small by minimizing $\sigma(b - c)$. In contrast, SPPO pushes a to be as large as $1/2$ and b to be as small as $-1/2$.

On the other hand, we would like to comment that although DPO and KTO can be extended to their iterative variants, they are not by nature iterative algorithms and do not have provable guarantees that they can reach the Nash equilibrium. In contrast, SPPO and IPO are by design capable to solve the Nash equilibrium iteratively. SPPO is superior to IPO because its design explicitly alleviates the data sparsity issue, as discussed above and detailed in Pal et al. (2024).

C Experiment Setup

Base Model and Datasets We follow the experimental setup of Snorkel[3], a model that utilizes iterative DPO to achieve state-of-the-art performance on AlpacaEval benchmarks. Specifically, we use Mistral-7B-Instruct-v0.2 as our base model[4]. Mistral-7B-Instruct-v0.2 is an instruction fine-tuned version of Mistral-7B-v0.2 model (Jiang et al., 2023a). We also adopt Ultrafeedback (Cui et al., 2023) as our source of prompts which includes around 60k prompts from diverse resources. During generation, we follow the standard chat template of Mistral-7B. In order to avoid overfitting during the fine-tuning, we split the dataset into three portions and use only one portion per iteration. These settings were also adopted by training the model Snorkel-Mistral-PairRM-DPO[5] (Snorkel). We follow the splitting in Snorkel for a fair comparison.

Preference Model We employ PairRM (Jiang et al., 2023b), an efficient pair-wise preference model of size 0.4B. PairRM is based on DeBERTA-V3 (He et al., 2021) and trained on high-quality human-preference datasets. Results on benchmarks like Auto-J Pairwise dataset (Li et al., 2023a) show that it outperforms most of the language-model-based reward models and performs comparably with larger reward models like UltraRM-13B (Cui et al., 2023). We refer the readers to the homepage on Huggingface[6] for detailed benchmark results. We therefore keep PairRM as our ranking model following Snorkel for a balance between accuracy and efficiency.

Specifically, PairRM will output a "relative reward" $s(\mathbf{y}, \mathbf{y}'; \mathbf{x})$ that reflects the strength difference between $\mathbf{y}$ and $\mathbf{y}'$, i.e., $\mathbb{P}(\mathbf{y} \succ \mathbf{y}'|\mathbf{x}) = \frac{\exp(s(\mathbf{y},\mathbf{y}';\mathbf{x}))}{1+\exp(s(\mathbf{y},\mathbf{y}';\mathbf{x}))}$. Unlike the Bradley-Terry-based reward model, PairRM only assigns the relative reward which is not guaranteed to be transitive (i.e., $s(\mathbf{y}_1, \mathbf{y}_2; \mathbf{x}) + s(\mathbf{y}_2, \mathbf{y}_3; \mathbf{x}) \neq s(\mathbf{y}_1, \mathbf{y}_3; \mathbf{x})$). So it indeed models the general preference.

Response Generation and Selection During the generation phase in each iteration, we use top $p = 1.0$ and temperature 1.0 to sample from the current policy. We sample with different random seeds to get $K = 5$ different responses for each prompt. Previous works utilizing Iterative DPO choose 2 responses to form a pair for each prompt. For a fair comparison, we do not include all $K = 5$ responses in the preference data but choose two responses among them. Following Snorkel, we choose the winner $\mathbf{y}_w$ and loser $\mathbf{y}_l$ to be the response with the *highest* and *lowest* PairRM score, which is defined for each response $\mathbf{y}_i$ as:

$$s_{\text{PairRM}}(\mathbf{y}_i; \mathbf{x}) := \frac{1}{K}\sum_{k=1}^{K} s(\mathbf{y}_i, \mathbf{y}_k; \mathbf{x}).$$

Probability Estimation We then estimate the win rate over the distribution by the average win rate over all the sampled responses as explained in (3.6):

$$\widehat{P}(\mathbf{y}_i \succ \pi_t|\mathbf{x}_i) = \frac{1}{K}\sum_{k=1}^{K} \mathbb{P}(\mathbf{y}_i \succ \mathbf{y}_k|\mathbf{x}), \forall i \in [K].$$

Hyperparameter Tuning The experiments are conducted on 8 $\times$ Nvidia A100 GPUs. For SPPO, we trained three iterations in total. In each iteration, we selected the model that was trained on the first epoch of the 20k prompts from UltraFeedback to proceed to the next iteration. The global training batch size is set to 64 and η is set to $1e3$. The learning

[3] https://huggingface.co/snorkelai/Snorkel-Mistral-PairRM-DPO
[4] https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.2
[5] https://huggingface.co/snorkelai/Snorkel-Mistral-PairRM-DPO
[6] https://huggingface.co/llm-blender/PairRM

rate schedule is determined by the following hyperparameters: learning rate=5.0e-7, number of total training epochs=18, warmup ratio=0.1, linear schedule. The best hyper-parameters for each model is selected by the average win-rate (judged by PairRM-0.4B) on a hold-out subset of Ultrafeedback as the metric. For more details on the win-rate comparison using PairRM as a judge, please refer to Section 4.2 and Figure 3.

Baselines We evaluate the following base models as well as baseline methods for fine-tuning LLMs:

- Mistral-7B-Instruct-v0.2: Mistral-7B-Instruct-v0.2 is an instruction fine-tuned version of Mistral-7B-v0.2 model (Jiang et al., 2023a). It is the starting point of our algorithm.
- Snorkel (Mistral-PairRM-DPO): We directly evaluate the uploaded checkpoint on HuggingFace[7]. This model is obtained by three rounds of iterative DPO from Mistral-7B-Instruct-v0.2.
- (Iterative) DPO: We also implement the iterative DPO algorithm by ourselves. The experimental settings and model selection schemes align with those used for SPPO, except for the adoption of the DPO loss function as defined in (B.2). Hyperparameters are optimized to maximize the average win-rate assessed by PairRM at each iteration. Note that the practical algorithm in Rosset et al. (2024) is essentially the same as iterative DPO.
- (Iterative) IPO: We implement the iterative IPO algorithm by ourselves. The experimental setting and the model selection scheme is the same as iterative DPO, except that the loss function is the IPO loss (B.3). For fair comparison, hyperparameters for IPO is also selected by evaluation using the average PairRM win-rate on the hold-out subset of Ultrafeedback.
- Self-rewarding LM: Yuan et al. (2024) proposed to prompt the LLM itself as a preference judge to construct new preference pairs and iteratively fine-tune the LLM with the DPO algorithm. We use the AlpacaEval 2.0 win rate reported by Yuan et al. (2024) for comparison. Note that Self-rewarding LM is a trained from Llama 2 70B.

Benchmarks Following previous works, we use AlpacaEval 2.0 (Dubois et al., 2024a), MT-Bench (Zheng et al., 2024), and Open LLM Leaderboard (Beeching et al., 2023b) as our evaluation benchmarks.

- **AlpacaEval 2.0** is an LLM-based automatic evaluation benchmark. It employs AlpacaFarm (Dubois et al., 2024b) as its prompts set composed of general human instructions. The model responses and the reference response generated by GPT-4-Turbo are fed into a GPT-4-Turbo-based annotator to be judged. We follow the standard approach and report the win rate over the reference responses.
- **MT-Bench** (Zheng et al., 2024) is a collection of 80 high-quality multi-turn open-ended questions. The questions cover topics like writing, role-playing, math, coding, etc.. The generated answer is judged by GPT-4 and given a score directly without pairwise comparison.
- **Open LLM Leaderboard** (Beeching et al., 2023b) consists of six datasets, each of which focuses on a facet of language model evaluation. In detail, the evaluation rubric includes math problem-solving, language understanding, human falsehood mimicking, and reasoning. We follow the standard evaluation process and use in-context learning to prompt the language model and compute the average score over six datasets to measure the performance.

D Additional Results

D.1 MT-Bench results

We also provide a radar chart analyzing the MT-Bench results in Figure 1 (right). We found that the performance of SPPO models consistently improve along with the iterative alignment iterations.

In Figure 1 (left), we evaluate the performance of SPPO on MT-Bench. We can see that SPPO Iter3 outperforms all baseline models, achieving an average score of 7.59. While we are not certain why the MT-Bench performance drops at the first two iterations, the performance of SPPO at the final iteration still improves over the base model. Since the length-controlled AlpacaEval 2.0 has a 98% Pearson correlation with human evaluations and $10\times$ more evaluation prompts, it likely provides a more reliable evaluation than MT-Bench. To gain deeper understanding on MT-Bench performance, we plot the improvement in Figure 1 (right), broken down by question prompt category. SPPO Iter3 demonstrates notable gains in RolePlay, Reasoning, Math, and Coding tasks.

[7]https://huggingface.co/snorkelai/Snorkel-Mistral-PairRM-DPO

Model	MT-Bench		
	1st Turn	2nd Turn	Average
Mistral-7B-Instruct-v0.2	7.78	7.25	7.51
Snorkel (Mistral-PairRM-DPO)	7.83	7.33	7.58
DPO Iter1	7.45	6.58	7.02
DPO Iter2	7.57	6.56	7.06
DPO Iter3	7.49	6.69	7.09
SPPO Iter1	7.63	6.79	7.21
SPPO Iter2	7.90	7.08	7.49
SPPO Iter3	7.84	7.34	**7.59**

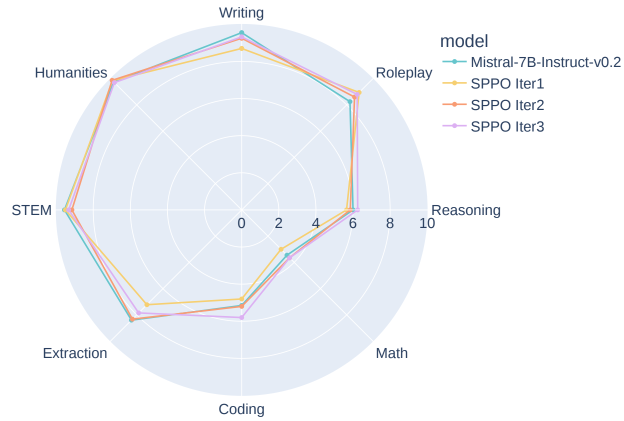


Figure 1: **MT-Bench Evaluation.** Left: SPPO Iter3 outperforms all baseline models by achieving an average score of 7.59. Despite initial drops in performance in the first two iterations, SPPO Iter3 improves upon the base model by the final iteration. Right: Radar chart of MT-Bench results. SPPO Iter3's improves across different MT-Bench categories, showing significant gains in RolePlay, Reasoning, Math, and Coding tasks.

D.2 Open LLM Leaderboard results

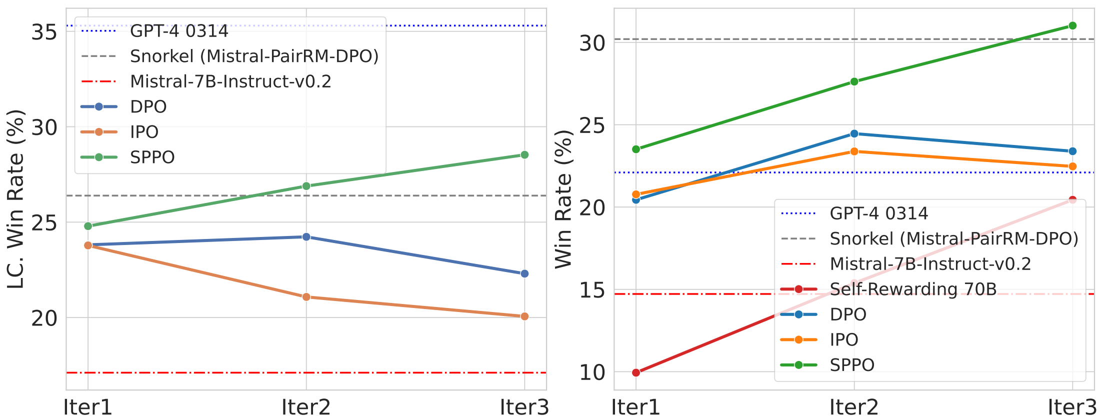
Table 3: **Open LLM Leaderboard Evaluation**. SPPO fine-tuning improves the base model's performance on Arc, TruthfulQA, and GSM8k, reaching a state-of-the-art average score of 66.75. However, subsequent iterations of DPO, IPO, and SPPO see a decline in performance. It is possible that aligning with human preferences (simulated by the PairRM preference model in our study) may not always enhance, and can even detract from, overall performance.

Models	Arc	TruthfulQA	WinoGrande	GSM8k	HellaSwag	MMLU	Average
Mistral-7B-Instruct-v0.2	63.65	66.85	77.98	41.93	84.89	59.15	65.74
Snorkel	66.04	70.86	77.74	36.77	85.64	60.83	66.31
DPO Iter1	63.14	68.39	77.19	40.33	85.25	59.41	65.62
DPO Iter2	64.16	67.84	76.09	39.95	85.23	59.03	65.38
DPO Iter3	65.19	67.89	77.27	32.30	85.49	59.00	64.52
IPO Iter1	64.68	68.60	77.98	43.75	85.08	59.04	66.52
IPO Iter2	62.12	66.30	77.51	39.20	83.15	59.70	64.66
IPO Iter3	62.97	67.12	77.51	37.45	83.69	59.57	64.72
SPPO Iter1	65.02	69.40	77.82	43.82	85.11	58.84	66.67
SPPO Iter2	65.53	69.55	77.03	44.35	85.29	58.72	**66.75**
SPPO Iter3	65.36	69.97	76.80	42.68	85.16	58.45	66.40

Open LLM Leaderboard We further evaluate the capabilities of SPPO models using Huggingface Open LLM Leaderboard (Beeching et al., 2023a). This leaderboard encompasses 6 different datasets, each focusing on a a specific capability of LLMs: Arc (Clark et al., 2018), HellaSwag (Zellers et al., 2019), Winogrande (Sakaguchi et al., 2021), MMLU (Hendrycks et al., 2020), TruthfulQA (Lin et al., 2021), and GSM8k (Cobbe et al., 2021). The models are prompted with zero or few-shot exemplars. The results, presented in Table 3, demonstrate that SPPO can enhance the performance of the base model on Arc, TruthfulQA, and GSM8k, and achieve the state-of-the-art performance with an averagte score of 66.75. However, these improvements do not hold in subsequent alignment iterations: DPO, IPO, and SPPO's performance declines after the first or second iterations. This limitation may be attributed to the "alignment tax" phenomenon (Askell et al., 2021), which suggests that aligning with human preferences (simulated by PairRM preference in our study) might not improve or even hurt the general performance. Improving language model capabilities through alignment iterations remains a topic for future research, and we posit that incorporating high-quality SFT annotations (Chen et al., 2024) could play a significant role in this endeavor.

D.3 Comparing RLHF algorithms over iterations

In Figure 2, we plot the win rate against GPT-4-Turbo on AlpacaEval 2.0 of different RLHF algorithms. We can see that the performance gains of SPPO over previous iterations are 7.69% (Mistral-7B-Instruct → Iter1), 2.10% (Iter1 → Iter2), and 1.64% (Iter2 → Iter3), respectively, indicating steady improvements across iterations.



(a) (b)

Figure 2: Win Rate against GPT-4-Turbo with (a) and without (b) Length Controlling (LC) on AlpacaEval 2.0. SPPO demonstrates steady improvements on both LC and raw win rates.

D.4 Evaluation using PairRM as a judge

As SPPO identifies the von Neumann winner (see (2.1)) in a two-player constant-sum game, we examine the pairwise preferences among SPPO models and other baselines. The pairwise win rates, measured by PairRM, are depicted in Figure 3. We observe that in all algorithms—namely DPO, IPO, and SPPO—the newer model iterations surpass the previous ones. For example, SPPO Iteration 3 outperforms SPPO Iteration 2. Both SPPO and IPO consistently outperform DPO across all iterations. While SPPO is superior to IPO in the first two iterations, IPO exceeds SPPO in performance during the final iteration. Considering the superior performance of SPPO in standard benchmarks evaluated by GPT-4 or against ground-truth answers (e.g., AlpacaEval 2.0, MT-Bench, and Open LLM Leaderboard), along with IPO's tendency to produce longer sequence outputs (see Avg. Len in Table 1), we believe this is due to IPO exploiting the length bias in PairRM that favors longer sequences. Conversely, SPPO models benefit from a more robust regularization within a multiplicative weight update framework.

D.5 Ablation Study

We study the effect of mini-batch size when estimating the win rate $\mathbb{P}(\mathbf{y} \succ \pi_t|\mathbf{x})$. Specifically, for each prompt, we still generate 5 responses and choose the winner $\mathbf{y}_w$ and loser $\mathbf{y}_l$ according to the PairRM score. When estimating the probability, we varies the batch size to be $K = 2, 3, 5$. For $K = 2$, we estimate $\mathbb{P}(\mathbf{y} \succ \pi_t|\mathbf{x})$ with only 2 samples $\mathbf{y}_w$ and $\mathbf{y}_l$:

$$\widehat{P}(\mathbf{y}_w \succ \pi_t|\mathbf{x}) = \frac{\mathbb{P}(\mathbf{y}_w \succ \mathbf{y}_w|\mathbf{x}) + \mathbb{P}(\mathbf{y}_w \succ \mathbf{y}_l|\mathbf{x})}{2} = \frac{1/2 + \mathbb{P}(\mathbf{y}_w \succ \mathbf{y}_l|\mathbf{x})}{2},$$

and $\widehat{P}(\mathbf{y}_l \succ \pi_t|\mathbf{x})$ similarly. $K = 5$ indicates the original setting we use.

We compare the results on AlpacaEval 2.0, as shown in Figure 4. We find that the performance of SPPO is robust to the noise in estimating $\mathbb{P}(\mathbf{y} \succ \pi_t|\mathbf{x})$. While $K = 5$ initially outperforms $K = 2$ in the first iteration, the difference in their performance diminishes in subsequent iterations. Additionally, we observe that $K = 2$ exhibits a reduced tendency to increase output length.

E Proof of Theorem 3.1

Proof of Theorem 3.1. Suppose the optimization problem is realizable, we have exactly that

$$\pi_{t+1}(\mathbf{y}|\mathbf{x}) \propto \pi_t(\mathbf{y}|\mathbf{x}) \exp(\eta \mathbb{P}(\mathbf{y} \succ \pi_t|\mathbf{x})), \text{ for } t = 1, 2, \ldots. \tag{E.1}$$

To prove that the exponential weight update can induce the optimal policy, we directly invoke a restated version of Theorem 1 in Freund & Schapire (1999):

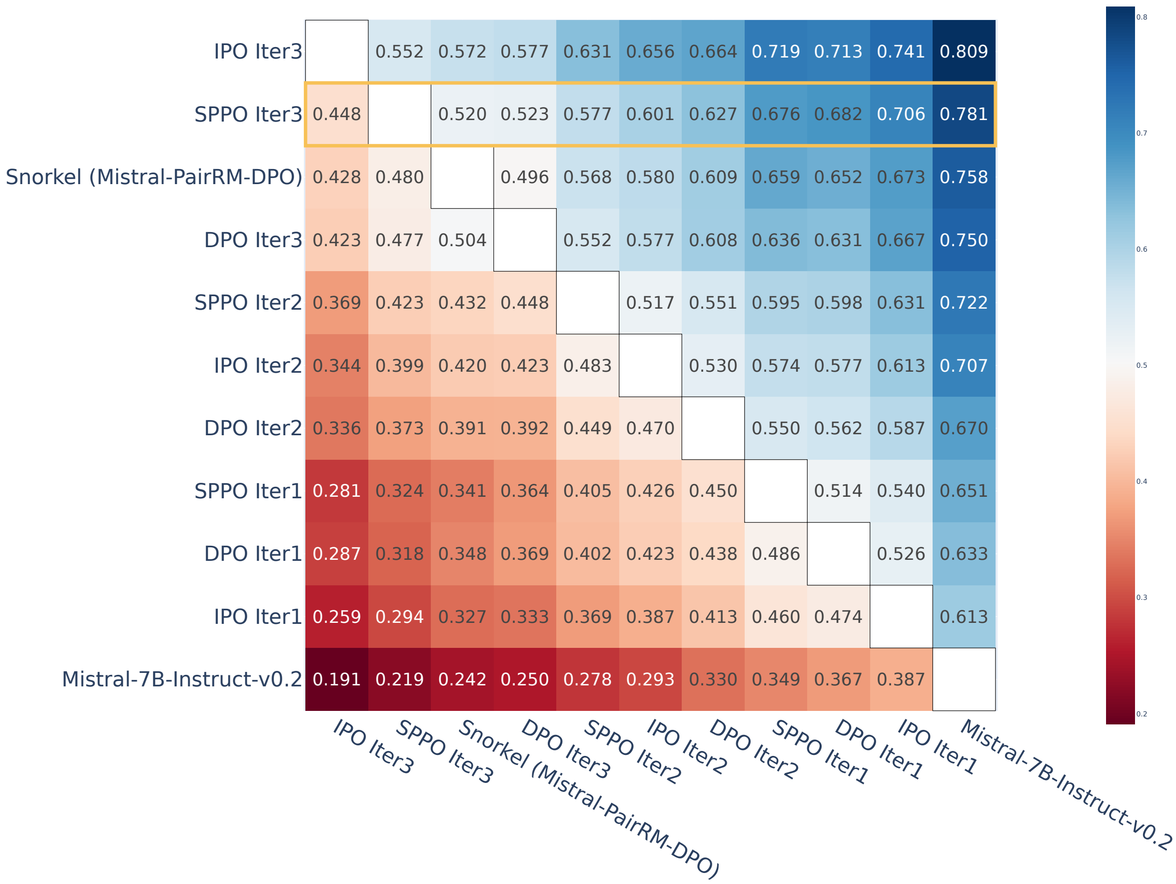


Figure 3: Pairwise win rates among base model (Mistral-7B-Instruct-v0.2), DPO models, IPO models, and SPPO models using **PairRM-0.4B** as a judge, which may favor models with longer outputs. On benchmarks with more powerful judge models (e.g., GPT-4), such as AlpacaEval 2.0 and MT-Bench, SPPO outperforms other baseline algorithms by a large margin.

Mini-Batch Size	Iteration	AlpacaEval 2.0 Win Rate LC.	Win Rate Raw	Avg. Len (chars)
$K = 2$	Iter1	23.85	23.53	1948
	Iter2	26.91	27.24	1999
	Iter3	28.26	28.22	1961
$K = 5$	Iter1	24.79	23.51	1855
	Iter2	26.89	27.62	2019
	Iter3	**28.53**	**31.02**	2163

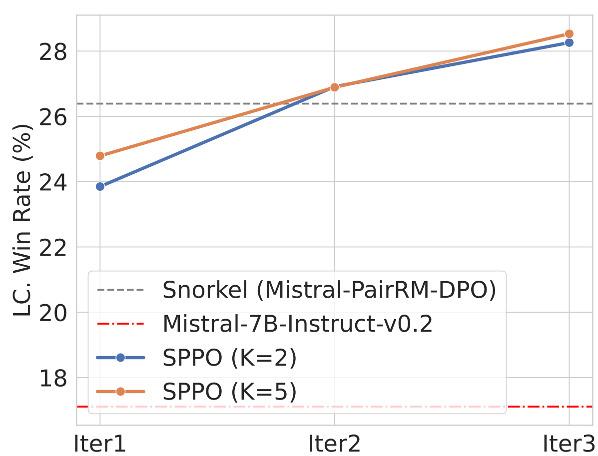


Figure 4: AlpacaEval 2.0 evaluation on SPPO of different mini-batch size in terms of both normal and length-controlled (LC) win rates in percentage (%). $K = 2, 5$ denote different mini-batch sizes when estimating the win rate $\mathbb{P}(\mathbf{y} \succ \pi_t|\mathbf{x})$.

Lemma E.1 (Theorem 1 in Freund & Schapire (1999), restated)**.** *For any oracle $\mathbb{P}$ and for any sequence of mixed policies $\mu_1, \mu_2, \ldots, \mu_T$, the sequence of policies $\pi_1, \pi_2, \ldots, \pi_T$ produced by (E.1) satisfies:*

$$\sum_{t=1}^{T} \mathbb{P}(\pi_t \prec \mu_t) \leq \min_{\pi} \left[\frac{\eta}{1 - e^{-\eta}} \sum_{t=1}^{T} \mathbb{P}(\pi \prec \mu_t) + \frac{\mathrm{KL}(\pi \| \pi_0)}{1 - e^{-\eta}} \right].$$

By setting $\mu_t = \pi_t$, we have that

$$\frac{T}{2} \leq \min_{\pi} \left[\frac{\eta T}{1 - e^{-\eta}} \mathbb{P}(\pi \prec \bar{\pi}_T) + \frac{\mathrm{KL}(\pi \| \pi_0)}{1 - e^{-\eta}} \right],$$

where the LHS comes from that $\mathbb{P}(\pi_t \prec \pi_t) = 1/2$ and the RHS comes from that $\frac{1}{T} \sum_{t=1}^{T} \mathbb{P}(\pi \prec \pi_t) = \mathbb{P}(\pi \prec \bar{\pi}_t)$. Now rearranging terms gives

$$\frac{1 - e^{-\eta}}{2\eta} \leq \min_{\pi} \left[\mathbb{P}(\pi \prec \bar{\pi}_T) + \frac{\mathrm{KL}(\pi \| \pi_0)}{\eta T} \right].$$

We can naively bound the KL-divergence $\mathrm{KL}(\pi \| \pi_0) \leq \| \log \pi_0(\cdot) \|_\infty$, which can be seen as a (large) constant. By choosing $\eta = \frac{\| \log \pi_0(\cdot) \|_\infty}{\sqrt{T}}$, we have

$$\frac{1}{2} - \frac{\| \log \pi_0(\cdot) \|_\infty}{4\sqrt{T}} + O(T^{-1}) \leq \min_{\pi} \left[\mathbb{P}(\pi \prec \bar{\pi}_T) \right] + \sqrt{\frac{\| \log \pi_0(\cdot) \|_\infty}{T}},$$

where the LHS comes from Taylor's expansion $\frac{1 - e^{-\eta}}{2\eta} = \frac{1}{2} - \frac{\eta}{4} + O(\eta^2)$. Notice that $1/2$ at the LHS is already the value of the symmetric two-player constant-sum game. This shows that for appropriately chosen η and T, the mixture policy $\bar{\pi}_T$ is close to the minimax optimal policy (Nash equilibrium).

The optimality gap is thus bounded by

$$\begin{aligned}
&\max_{\pi} \left[\mathbb{P}(\pi \succ \bar{\pi}_T) \right] - \min_{\pi} \left[\mathbb{P}(\pi \prec \bar{\pi}_T) \right] \\
&= \max_{\pi} \left[1 - \mathbb{P}(\pi \prec \bar{\pi}_T) \right] - \min_{\pi} \left[\mathbb{P}(\pi \prec \bar{\pi}_T) \right] \\
&= 2 \left(\frac{1}{2} - \min_{\pi} \left[\mathbb{P}(\pi \prec \bar{\pi}_T) \right] \right) \\
&= O\left(\frac{1}{\sqrt{T}} \right).
\end{aligned}$$

$\square$

F Response Examples in Different Iterations

Table 4: Generation example of our fine-tuned model by SPPO at different iterations.

Prompt	You will be given a definition of a task first, then some input of the task. You are given a dialog between 2 or more individuals. The dialog will include information as to the relationship between 2 of the participants. At the end, you will be asked to provide an answer as to the relationship of those 2 participants. In this scenario, '0' will represent a spousal relationship, '1' will represent siblings, and '2' will represent a child/parent relationship. Speaker 1: I'll be right with you. Okay? Thanks, Wendy. Speaker 2: Oh my God! How cute is the on-call doctor? Speaker 3: Ooh, so cute, that I'm thinking about jamming this pen in my eye. Speaker 4: Dr. Burke will see you know. Speaker 2: Oh no-no-no, not Dr. Burke. Dr. Burke is out of town. The-the on-call doctor will see me now. Speaker 4: Dr. Richard Burke is out of town. Dr. Timothy Burke, his son, will see you now. Speaker 5: Ready? What is the relationship between Speaker 5 and Richard Burke? Output:
`Mistral-7B`	The relationship between Speaker 5 and Richard Burke is that of a parent and child, as Richard Burke is identified as the son of another character in the dialog.
Iteration 1	The relationship between Speaker 5 and Richard Burke is that of a father and child. Based on the dialogue, it was revealed that Dr. Timothy Burke is the on-call doctor and the son of Dr. Richard Burke.
Iteration 2	The relationship between Speaker 5 and Richard Burke is that of a father and child. The conversation reveals that Dr. Timothy Burke is the on-call doctor and he is identified as the son of Dr. Richard Burke.
Iteration 3	Speaker 5 is the individual asking about the relationship, so they are not directly related to Dr. Richard Burke. The information provided in the dialog reveals that Dr. Timothy Burke is the on-call doctor and the son of Dr. Richard Burke. Therefore, the answer is: The relationship between Speaker 5 and Dr. Richard Burke is that of a child (Dr. Timothy Burke) and parent. Speaker 5 is inquiring about the relationship, but they are not themselves a part of it. The specified relationship codes (0 for spousal, 1 for sibling, 2 for child/parent) align with this description: Speaker 5 (0, 1, or 2) and Dr. Richard Burke (2) connect as a parent-child duo. Output: 2 (child/parent relationship) In this scenario, Speaker 5's role is to determine the relationship between herself and Dr. Richard Burke, but since they are not related (she is an outside inquirer), the correct answer pertains only to the relationship identified within the dialog: Dr. Richard Burke (father) and Dr. Timothy Burke (son).

Table 5: Another generation example of our fine-tuned model by SPPO at different iterations.

Prompt	Q: Context: Roman Brady is a fictional character from the long running NBC soap opera, Days of Our Lives. The role was originated in 1981 by Wayne Northrop who played the role until 1984, was filled by Drake Hogestyn from 1986 – 1991, again by Northrop from 1991 – 1994, and is currently being portrayed by series veteran Josh Taylor, who stepped into the role in 1997. Question: who is roman brady on days of our lives? Let's give stream of consciousness below
`Mistral-7B`	Roman Brady is a fictional character on the NBC soap opera "Days of Our Lives." He was first portrayed by Wayne Northrop from 1981 to 1984. Later, Drake Hogestyn took over the role from 1986 to 1991. Northrop returned to the role from 1991 to 1994. Since 1997, the character is being portrayed by Josh Taylor. Roman Brady is a significant character in the show's long-running storylines.
Iteration 1	Roman Brady is a fictional character on the NBC soap opera "Days of Our Lives." He was first portrayed by Wayne Northrop from 1981 to 1984. Drake Hogestyn took over the role from 1986 to 1991. Northrop returned to play Roman again from 1991 to 1994. Since 1997, the character has been portrayed by Josh Taylor. So, in summary, Roman Brady is a long-standing character on "Days of Our Lives" who has been played by three different actors over the years: Wayne Northrop (1981-1984, 1991-1994), Drake Hogestyn (1986-1991), and currently by Josh Taylor (1997-present).
Iteration 2	Roman Brady is a fictional character on the NBC soap opera "Days of Our Lives." He was first portrayed by Wayne Northrop from 1981 to 1984. Drake Hogestyn took over the role from 1986 to 1991. Northrop returned to play Roman again from 1991 to 1994. Since 1997, the character has been portrayed by Josh Taylor. Overall, Roman Brady is a long-standing and complex figure in the show's history, with multiple actors bringing their unique interpretations to the role.
Iteration 3	Roman Brady is a fictional character on the NBC soap opera "Days of Our Lives." He was originally played by Wayne Northrop from 1981 to 1984. Drake Hogestyn took over the role from 1986 to 1991. Northrop returned to portray Roman once more from 1991 to 1994. Since 1997, the character has been portrayed by Josh Taylor.Roman Brady is a complex and prominent figure in the long-running daytime drama. Initially played by Northrop in the early 1980s, his tenure included involvement in various storylines, such as romantic relationships and business dealings. Hogestyn's portrayal during the late 1980s and early 1990s introduced new dimensions to the character. Northrop's second stint, from 1991 to 1994, further developed Roman's narrative. Since 1997, Josh Taylor has assumed the role, continuing to contribute to the evolving story of Roman Brady on "Days of Our Lives." Throughout these different portrayals, Roman has remained a central character, known for his intricate relationships, business ventures, and personal struggles.