Self-Play Preference Optimization for Language Model Alignment

Yue Wu[1]*, **Zhiqing Sun**[2]*, **Huizhuo Yuan**[1]*, **Kaixuan Ji**[1], **Yiming Yang**[2], **Quanquan Gu**[1]

[1]University of California, Los Angeles
[2]Carnegie Mellon University
{ywu, hzyuan, kauxuanji, qgu}@cs.ucla.edu,
{zhiqings, yiming}@cs.cmu.edu

Abstract

Standard 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 *Algorithm* (SPPO), utilizes iterative policy updates to provably approximate the Nash equilibrium. Additionally, we propose a new SPPO objective which is both strongly motivated by theory and is simple and effective in practice. 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, Arena-Hard, and the Open LLM Leaderboard. Starting from a stronger base model Llama-3-8B-Instruct, we are able to achieve a length-controlled win rate of 38.77%. 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 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 and Terry (1952), which assume consistent and tran-

Table 1: AlpacaEval 2.0 leaderboard results.

Model	AlpacaEval 2.0	
	LC. Win Rate	Win Rate
GPT-4 Turbo	50.0	50.0
Claude 3 Opus	40.5	29.1
Llama-3-8B-SPPO Iter3	38.8	39.9
GPT-4 0314	35.3	22.1
Llama 3 70B Instruct	34.4	33.2
Mistral-7B-SPPO Iter3 (best-of-16)	32.1	34.9
GPT-4 0613	30.2	15.8
Snorkel (Mistral-PairRM-DPO best-of-16)	30.0	34.9
Mistral Medium	28.6	21.9
Mistral-7B-SPPO Iter3	28.5	31.0
Claude 2	28.2	17.2
Snorkel (Mistral-PairRM-DPO)	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

---

*Equal contribution

sitive 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 algorithmic framework, optimizing a theoretically motivated and practically effective loss objective. Empirical evidence shows that SPPO enhances the Mistral-7B-Instruct-v0.2 and Llama-3-8B-Instruct model, achieving significant improvements without external supervision from stronger models like GPT-4.

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 $\mathrm{KL}(p\|q) = \mathbb{E}_{\mathbf{y} \sim p(\mathbf{y})}\left[\log \frac{p(\mathbf{y})}{q(\mathbf{y})}\right]$.

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:

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

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 and 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:

$$\pi_{t+1}(\mathbf{y}|\mathbf{x}) \propto \pi_t(\mathbf{y}|\mathbf{x}) \exp(\eta \mathbb{P}(\mathbf{y} \succ \pi_t|\mathbf{x})). \tag{3.1}$$

(3.1) 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, (3.1) can be written as

$$\pi_{t+1}(\mathbf{y}|\mathbf{x}) = \frac{\pi_t(\mathbf{y}|\mathbf{x}) \exp\left(\eta \mathbb{P}(\mathbf{y} \succ \pi_t|\mathbf{x})\right)}{Z_{\pi_t}(\mathbf{x})}, \tag{3.2}$$

where $Z_{\pi_t}(\mathbf{x}) = \sum_{\mathbf{y}} \pi_t(\mathbf{y}|\mathbf{x}) \exp\left(\eta \mathbb{P}(\mathbf{y} \succ \pi_t|\mathbf{x})\right)$ 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:

$$\log\left(\frac{\pi_{t+1}(\mathbf{y}|\mathbf{x})}{\pi_t(\mathbf{y}|\mathbf{x})}\right) = \eta \cdot \mathbb{P}(\mathbf{y} \succ \pi_t|\mathbf{x}) - \log Z_{\pi_t}(\mathbf{x}). \tag{3.3}$$

Unlike the pair-wise design in DPO or IPO that cancels the log normalizing factor $\log Z_{\pi_t}(\mathbf{x})$ by differentiating (3.3) between $\mathbf{y}$ and $\mathbf{y}'$, we choose to approximate (3.3) directly in terms of L_2 distance:

$$\pi_{t+1} = \underset{\pi}{\arg\min} \, \mathbb{E}_{\mathbf{x} \sim \mathcal{X}, \mathbf{y} \sim \pi_t(\cdot|\mathbf{x})} \left(\log\left(\frac{\pi(\mathbf{y}|\mathbf{x})}{\pi_t(\mathbf{y}|\mathbf{x})}\right) - \left(\eta \mathbb{P}(\mathbf{y} \succ \pi_t|\mathbf{x}) - \log Z_{\pi_t}(\mathbf{x})\right)\right)^2. \tag{3.4}$$

Estimation of the Probability The optimization objective (3.4) 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

$$\pi_{t+1} = \underset{\pi}{\arg\min} \, \mathbb{E}_{\mathbf{x} \sim \mathcal{X}, \mathbf{y} \sim \pi_t(\cdot|\mathbf{x})} \left(\log\left(\frac{\pi(\mathbf{y}|\mathbf{x})}{\pi_t(\mathbf{y}|\mathbf{x})}\right) - \left(\eta \mathbb{P}(\mathbf{y} \succ \widehat{\pi}_t^K|\mathbf{x}) - \log Z_{\widehat{\pi}_t^K}(\mathbf{x})\right)\right)^2. \tag{3.5}$$

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 new samples with in total $O(KB)$ queries of the preference oracle $\mathbb{P}$. (3.5) is an efficiently tractable optimization problem. Informally speaking, when $K \to \infty$, (3.5) will recover (3.4). We have the following guarantee on the convergence of (3.4):

Theorem 3.1. Assume the optimization problem (3.4) is realizable. Denote π_t as the policy obtained via (3.4) 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

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

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 and Schapire (1999) with slight modification. For completeness, we include the proof in Appendix C. Alternatively, we can avoid estimating $\log Z_{\widehat{\pi}_t^K}(\mathbf{x})$ by replacing it simply with $\eta/2$[2] in (3.5) to obtain a more clear objective:

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

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 (3.6).

---

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

Algorithm 1 `Self-play Probabilistic 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 (3.6):

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

7: **end for**

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 (3.6) on the dataset $\mathcal{D}_t$ (Line 6).

3.3 Connection to Policy Gradient

While SPPO is derived from the iterative framework (Freund and Schapire, 1999) for two-player games, the square loss in the SPPO objective (3.4) provides an alternative interpretation for SPPO as a *semi-online* variant of policy gradient method due to its special loss form. The difference from standard policy gradient is that it collects samples from $\pi_{\boldsymbol{\theta}_t}$ at the start of iteration t, rather than perform on-policy sampling at each gradient step.

Consider a general reward function $r(\mathbf{y}; \mathbf{x})$, the RLHF problem can be written as:

$$\max_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) := \mathbb{E}_{\mathbf{x} \sim \mathcal{X}, \mathbf{y} \sim \pi_{\boldsymbol{\theta}}(\cdot|\mathbf{x})} \left[r(\mathbf{y}; \mathbf{x}) - \eta^{-1} \log \frac{\pi_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x})}{\pi_{\text{ref}}(\mathbf{y}|\mathbf{x})} \right]. \tag{3.8}$$

The policy gradient of the objective $J(\boldsymbol{\theta})$ is:

$$\nabla J(\boldsymbol{\theta}) = \mathbb{E}_{\mathbf{x} \sim \mathcal{X}, \mathbf{y} \sim \pi_{\boldsymbol{\theta}}(\cdot|\mathbf{x})} \left[\left(r(\mathbf{y}; \mathbf{x}) - \eta^{-1} \log \frac{\pi_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x})}{\pi_{\text{ref}}(\mathbf{y}|\mathbf{x})} - b(\mathbf{x}) \right) \nabla \log \pi_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x}) \right] \tag{3.9}$$

$$= \eta \mathbb{E}_{\mathbf{x} \sim \mathcal{X}, \mathbf{y} \sim \pi_{\boldsymbol{\theta}}(\cdot|\mathbf{x})} \left[-\nabla \left(r(\mathbf{y}; \mathbf{x}) - \eta^{-1} \log \frac{\pi_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x})}{\pi_{\text{ref}}(\mathbf{y}|\mathbf{x})} - b(\mathbf{x}) \right)^2 \right], \tag{3.10}$$

where the first line follows the policy gradient theorem (Sutton et al., 1999) and the baseline $b(\mathbf{x})$ is an arbitrary constant relying only on $\mathbf{x}$ used for variance reduction. Comparing the square loss (3.10) with the SPPO objective (3.4) (rewritten below):

$$\boldsymbol{\theta}_{t+1} = \operatorname*{argmin}_{\boldsymbol{\theta}} \mathbb{E}_{\mathbf{x} \sim \mathcal{X}, \mathbf{y} \sim \pi_{\boldsymbol{\theta}_t}(\cdot|\mathbf{x})} \left[\left(\mathbb{P}(\mathbf{y} \succ \pi_{\boldsymbol{\theta}_t}|\mathbf{x}) - \eta^{-1} \log \left(\frac{\pi_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x})}{\pi_{\boldsymbol{\theta}_t}(\mathbf{y}|\mathbf{x})} \right) - \eta^{-1} \log Z_{\pi_{\boldsymbol{\theta}_t}}(\mathbf{x}) \right)^2 \right],$$

one can see that the win rate $\mathbb{P}(\mathbf{y} \succ \pi_{\boldsymbol{\theta}_t}|\mathbf{x})$ is exactly the reward SPPO aims to maximize, and $\eta^{-1} \log Z_{\pi_{\boldsymbol{\theta}_t}}(\mathbf{x})$ is in fact the best possible baseline–the (soft) value function. When the value function is not available in practice, it can be replaced by any constant baseline to reduce the variance of the policy gradient. We choose $1/2$ as a good approximation to $\eta^{-1} \log Z_{\pi_{\boldsymbol{\theta}_t}}(\mathbf{x})$ but the constant can vary depending on the human preference model (see Appendix B). This connection can also be extended to token-level preference optimization under the Max-Entropy RL formuation (see Section A).

4

Comparing with the general framework proposed by Swamy et al. (2024), SPPO can be seen as a new, straightforward variant of policy gradient method without the need of extra modifications such as gradient clipping in PPO, Hessian calculation in TRPO, or maintaining multiple components (Q-critic, V-critic, actor, etc.) in many policy optimization algorithms.

4 Experiments

4.1 Experiment Setup

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

Base Model and Datasets: We follow Snorkel's experimental setup, using Mistral-7B-Instruct-v0.2 and Llama-3-8B-Instruct 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 Snorkel, iterative DPO and IPO, and Self-rewarding LM as baselines. Benchmarks include AlpacaEval 2.0, MT-Bench, Arena-Hard, and the Open LLM Leaderboard, covering various aspects of language model evaluation.

4.2 Experimental Results

Due to the space limit, we only report the results on AlpacaEval 2.0 in the following and postpone the rest results including ablation studies to the appendix.

Table 2 (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, Mistral-7B-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 $\rightarrow$ Iter1), 2.10% (Iter1 $\rightarrow$ Iter2), and 1.64% (Iter2 $\rightarrow$ Iter3), respectively, indicating steady improvements across iterations, as illus-

Table 2: AlpacaEval 2.0 evaluation of models in Baselines).

Model	AlpacaEval 2.0		
	LC Win Rate	Win Rate	Avg. Len
Mistral-7B-Instruct-v0.2	17.11	14.72	1676
Mistral-7B-Instruct-v0.2 (best-of-16)	22.45	17.94	1529
Snorkel (Mistral-PairRM-DPO)	26.39	30.22	2736
Snorkel (Mistral-PairRM-DPO best-of-16)	29.97	34.86	2616
Self-Rewarding 70B Iter1	-	9.94	1092
Self-Rewarding 70B Iter2	-	15.38	1552
Self-Rewarding 70B Iter3	-	20.44	2552
Mistral-7B-DPO Iter1	23.81	20.44	1723
Mistral-7B-DPO Iter2	24.23	24.46	2028
Mistral-7B-DPO Iter3	22.30	23.39	2189
Mistral-7B-IPO Iter1	23.78	20.77	1693
Mistral-7B-IPO Iter2	21.08	23.38	2660
Mistral-7B-IPO Iter3	20.06	22.47	2760
Mistral-7B-SPPO Iter1	24.79$_{(+7.69)}$	23.51$_{(+8.79)}$	1855
Mistral-7B-SPPO Iter2	26.89$_{(+2.10)}$	27.62$_{(+4.11)}$	2019
Mistral-7B-SPPO Iter3	28.53$_{(+1.64)}$	31.02$_{(+3.40)}$	2163
Mistral-7B-SPPO Iter1 (best-of-16)	28.71$_{(+6.26)}$	27.77$_{(+9.83)}$	1901
Mistral-7B-SPPO Iter2 (best-of-16)	31.23$_{(+2.52)}$	32.12$_{(+4.35)}$	2035
Mistral-7B-SPPO Iter3 (best-of-16)	32.13$_{(+0.9)}$	34.94$_{(+2.82)}$	2174
Llama-3-8B-Instruct	22.92	22.57	1899
Llama-3-8B-SPPO Iter1	31.73$_{(+8.81)}$	31.74$_{(+9.17)}$	1962
Llama-3-8B-SPPO Iter2	35.15$_{(+3.42)}$	35.98$_{(+4.24)}$	2021
Llama-3-8B-SPPO Iter3	38.77$_{(+3.62)}$	39.85$_{(+3.87)}$	2066

trated in Figure 1. We also apply SPPO to a stronger baseline model, i.e., Llama-3-8B-Instruct, and the fine-tuned model Llama-3-8B-SPPO has a higher length-controlled win rate 38.77% and overall win rate 39.85%. The performance gains are more significant: 8.81% (Llama-3-8B-Instruct $\rightarrow$ Iter1), 3.42% (Iter1 $\rightarrow$ Iter2), and 3.62% (Iter2 $\rightarrow$ Iter3), summing up to a total gain of 15.85%.

Additionally, the result 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. Finally, we present the best-of-16 results for each model, selected using the PairRM reward model. We find that re-ranking with the preference model at test time can consistently improve the performance of base model (Mistral-7B-Instruct-v0.2), DPO (Snorkel), and SPPO (Iter3) by 5.34%, 3.57%, and 3.6%, respectively.

References

ASKELL, A., BAI, Y., CHEN, A., DRAIN, D., GANGULI, D., HENIGHAN, T., JONES, A., JOSEPH, N., MANN, B., DASSARMA, N. ET AL. (2021). A general language assistant as a laboratory for alignment. *arXiv preprint arXiv:2112.00861* .

AZAR, M. G., ROWLAND, M., PIOT, B., GUO, D., CALANDRIELLO, D., VALKO, M. and MUNOS, R. (2023). A general theoretical paradigm to understand learning from human preferences. *arXiv preprint arXiv:2310.12036* .

BEECHING, E., FOURRIER, C., HABIB, N., HAN, S., LAMBERT, N., RAJANI, N., SANSEVIERO, O., TUNSTALL, L. and WOLF, T. (2023a). Open llm leaderboard. `https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard`.

BEECHING, E., FOURRIER, C., HABIB, N., HAN, S., LAMBERT, N., RAJANI, N., SANSEVIERO, O., TUNSTALL, L. and WOLF, T. (2023b). Open llm leaderboard. *Hugging Face* .

BRADLEY, R. A. and TERRY, M. E. (1952). Rank Analysis of Incomplete Block Designs: I. The Method of Paired Comparisons. *Biometrika* **39** 324–345.

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. (2024). Human alignment of large language models through online preference optimisation. *arXiv preprint arXiv:2403.08635* .

CHEN, Z., DENG, Y., YUAN, H., JI, K. and GU, Q. (2024). Self-play fine-tuning converts weak language models to strong language models. *arXiv preprint arXiv:2401.01335* .

CHRISTIANO, P. F., LEIKE, J., BROWN, T., MARTIC, M., LEGG, S. and AMODEI, D. (2017). Deep reinforcement learning from human preferences. *Advances in neural information processing systems* **30**.

CLARK, P., COWHEY, I., ETZIONI, O., KHOT, T., SABHARWAL, A., SCHOENICK, C. and TAFJORD, O. (2018). Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457* .

COBBE, K., KOSARAJU, V., BAVARIAN, M., CHEN, M., JUN, H., KAISER, L., PLAPPERT, M., TWOREK, J., HILTON, J., NAKANO, R. ET AL. (2021). Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168* .

CUI, G., YUAN, L., DING, N., YAO, G., ZHU, W., NI, Y., XIE, G., LIU, Z. and SUN, M. (2023). Ultrafeedback: Boosting language models with high-quality feedback. *arXiv preprint arXiv:2310.01377* .

DUBOIS, Y., GALAMBOSI, B., LIANG, P. and HASHIMOTO, T. B. (2024a). Length-controlled alpacaeval: A simple way to debias automatic evaluators. *arXiv preprint arXiv:2404.04475* .

DUBOIS, Y., LI, C. X., TAORI, R., ZHANG, T., GULRAJANI, I., BA, J., GUESTRIN, C., LIANG, P. S. and HASHIMOTO, T. B. (2024b). Alpacafarm: A simulation framework for methods that learn from human feedback. *Advances in Neural Information Processing Systems* **36**.

DUDÍK, M., HOFMANN, K., SCHAPIRE, R. E., SLIVKINS, A. and ZOGHI, M. (2015). Contextual dueling bandits. In *Conference on Learning Theory*. PMLR.

ETHAYARAJH, K., XU, W., MUENNIGHOFF, N., JURAFSKY, D. and KIELA, D. (2024). Kto: Model alignment as prospect theoretic optimization. *arXiv preprint arXiv:2402.01306* .

FREUND, Y. and SCHAPIRE, R. E. (1999). Adaptive game playing using multiplicative weights. *Games and Economic Behavior* **29** 79–103.

GAO, L., SCHULMAN, J. and HILTON, J. (2023). Scaling laws for reward model overoptimization. In *International Conference on Machine Learning*. PMLR.

GAO, Z., CHANG, J. D., ZHAN, W., OERTELL, O., SWAMY, G., BRANTLEY, K., JOACHIMS, T., BAGNELL, J. A., LEE, J. D. and SUN, W. (2024). Rebel: Reinforcement learning via regressing relative rewards. *arXiv preprint arXiv:2404.16767* .

HE, P., GAO, J. and CHEN, W. (2021). Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing.

HENDRYCKS, D., BURNS, C., BASART, S., ZOU, A., MAZEIKA, M., SONG, D. and STEINHARDT, J. (2020). Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300* .

HONG, J., LEE, N. and THORNE, J. (2024). Reference-free monolithic preference optimization with odds ratio. *arXiv preprint arXiv:2403.07691* .

JI, K., HE, J. and GU, Q. (2024). Reinforcement learning from human feedback with active queries. *arXiv preprint arXiv:2402.09401* .

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. (2023a). Mistral 7b. *arXiv preprint arXiv:2310.06825* .

JIANG, D., REN, X. and LIN, B. Y. (2023b). Llm-blender: Ensembling large language models with pairwise ranking and generative fusion. *arXiv preprint arXiv:2306.02561* .

LI, J., SUN, S., YUAN, W., FAN, R.-Z., ZHAO, H. and LIU, P. (2023a). Generative judge for evaluating alignment. *arXiv preprint arXiv:2310.05470* .

LI, T., CHIANG, W.-L., FRICK, E., DUNLAP, L., WU, T., ZHU, B., GONZALEZ, J. E. and STOICA, I. (2024). From crowdsourced data to high-quality benchmarks: Arena-hard and benchbuilder pipeline. *arXiv preprint arXiv:2406.11939* .

LI, X., ZHANG, T., DUBOIS, Y., TAORI, R., GULRAJANI, I., GUESTRIN, C., LIANG, P. and HASHIMOTO, T. B. (2023b). Alpacaeval: An automatic evaluator of instruction-following models. `https://github.com/tatsu-lab/alpaca_eval`.

LIN, S., HILTON, J. and EVANS, O. (2021). Truthfulqa: Measuring how models mimic human falsehoods. *arXiv preprint arXiv:2109.07958* .

LIU, T., ZHAO, Y., JOSHI, R., KHALMAN, M., SALEH, M., LIU, P. J. and LIU, J. (2023). Statistical rejection sampling improves preference optimization. *arXiv preprint arXiv:2309.06657* .

MUNOS, R., VALKO, M., CALANDRIELLO, D., AZAR, M. G., ROWLAND, M., GUO, Z. D., TANG, Y., GEIST, M., MESNARD, T., MICHI, A. ET AL. (2023). Nash learning from human feedback. *arXiv preprint arXiv:2312.00886* .

NG, A. Y., HARADA, D. and RUSSELL, S. (1999). Policy invariance under reward transformations: Theory and application to reward shaping. In *Icml*, vol. 99.

OPENAI, J., ACHIAM, ADLER, S., AGARWAL, S., AHMAD, L., AKKAYA, I., ALEMAN, F. L., ALMEIDA, D., ALTENSCHMIDT, J., ALTMAN, S., ANADKAT, S. ET AL. (2023). Gpt-4 technical report. *arXiv preprint arXiv:2303.08774* .

OUYANG, L., WU, J., JIANG, X., ALMEIDA, D., WAINWRIGHT, C., MISHKIN, P., ZHANG, C., AGARWAL, S., SLAMA, K., RAY, A. ET AL. (2022). Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems* **35** 27730–27744.

PAL, A., KARKHANIS, D., DOOLEY, S., ROBERTS, M., NAIDU, S. and WHITE, C. (2024). Smaug: Fixing failure modes of preference optimisation with dpo-positive. *arXiv preprint arXiv:2402.13228* .

RAFAILOV, R., HEJNA, J., PARK, R. and FINN, C. (2024a). From r to q*: Your language model is secretly a q-function. *arXiv preprint arXiv:2404.12358* .

RAFAILOV, R., SHARMA, A., MITCHELL, E., MANNING, C. D., ERMON, S. and FINN, C. (2024b). Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems* **36**.

ROSSET, C., CHENG, C.-A., MITRA, A., SANTACROCE, M., AWADALLAH, A. and XIE, T. (2024). Direct nash optimization: Teaching language models to self-improve with general preferences. *arXiv preprint arXiv:2404.03715* .

SAKAGUCHI, K., BRAS, R. L., BHAGAVATULA, C. and CHOI, Y. (2021). Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM* **64** 99–106.

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. (2023). Beyond human data: Scaling self-training for problem-solving with language models. *arXiv preprint arXiv:2312.06585* .

SUTTON, R. S., MCALLESTER, D., SINGH, S. and MANSOUR, Y. (1999). Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems* **12**.

SWAMY, G., DANN, C., KIDAMBI, R., WU, Z. S. and AGARWAL, A. (2024). A minimaximalist approach to reinforcement learning from human feedback. *arXiv preprint arXiv:2401.04056* .

TVERSKY, A. (1969). Intransitivity of preferences. *Psychological review* **76** 31.

WANG, Y., LIU, Q. and JIN, C. (2024). Is rlhf more difficult than standard rl? a theoretical perspective. *Advances in Neural Information Processing Systems* **36**.

XIONG, W., DONG, H., YE, C., ZHONG, H., JIANG, N. and ZHANG, T. (2023). Gibbs sampling from human feedback: A provable kl-constrained framework for rlhf. *arXiv preprint arXiv:2312.11456* .

XU, J., LEE, A., SUKHBAATAR, S. and WESTON, J. (2023). Some things are more cringe than others: Preference optimization with the pairwise cringe loss. *arXiv preprint arXiv:2312.16682* .

YE, C., XIONG, W., ZHANG, Y., JIANG, N. and ZHANG, T. (2024). A theoretical analysis of nash learning from human feedback under general kl-regularized preference. *arXiv preprint arXiv:2402.07314* .

YUAN, W., PANG, R. Y., CHO, K., SUKHBAATAR, S., XU, J. and WESTON, J. (2024). Self-rewarding language models. *arXiv preprint arXiv:2401.10020* .

ZELLERS, R., HOLTZMAN, A., BISK, Y., FARHADI, A. and CHOI, Y. (2019). Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830* .

ZHAO, Y., JOSHI, R., LIU, T., KHALMAN, M., SALEH, M. and LIU, P. J. (2023). Slic-hf: Sequence likelihood calibration with human feedback. *arXiv preprint arXiv:2305.10425* .

ZHENG, L., CHIANG, W.-L., SHENG, Y., ZHUANG, S., WU, Z., ZHUANG, Y., LIN, Z., LI, Z., LI, D., XING, E. ET AL. (2023). Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems* **36**.

ZHENG, L., CHIANG, W.-L., SHENG, Y., ZHUANG, S., WU, Z., ZHUANG, Y., LIN, Z., LI, Z., LI, D., XING, E. ET AL. (2024). Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems* **36**.

ZHU, B., JIAO, J. and JORDAN, M. I. (2023). Principled reinforcement learning with human feedback from pairwise or k-wise comparisons. *arXiv preprint arXiv:2301.11270* .

A Token-Level Q^* Learning

Rafailov et al. (2024a) showed that under the Max-Entropy RL formulation, the token-level log-ratio $\log \frac{\pi_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x})}{\pi_{\text{ref}}(\mathbf{y}|\mathbf{x})}$ can be seen as an implicit token-level reward or advantage function (invariant under reward shaping). Below we show the square loss in SPPO can also lead to the optimal Max-Entropy policy π^*, with token-level optimal value/advantage function.

We first briefly restate the setting and results in Rafailov et al. (2024b). The token-level MDP defines the state $\mathbf{s}_h = (\mathbf{x}, y_1, y_2, \dots, y_{h-1})$ as the prefix tokens, and the action $\mathbf{a}_h = y_h$ as the next token. An auto-regressive language model $\pi(\mathbf{y}|\mathbf{x})$ can be viewed as a token-level policy $\pi(\mathbf{a}_h|\mathbf{s}_h)$ and the transition kernel is known and deterministic because it only concatenates the next token to the prefix to form a new token sequence $\mathbf{s}_{h+1} = (\mathbf{x}, y_1, y_2, \dots, y_h)$.

The Max-Entropy RL setting again considers the reverse-KL regularized reward maximization problem:

$$\max_{\boldsymbol{\theta}} \mathbb{E}_{\mathbf{x}\sim\mathcal{X},\mathbf{y}\sim\pi_{\boldsymbol{\theta}}(\cdot|\mathbf{x})}[r(\mathbf{y};\mathbf{x})] - \eta^{-1}\mathbb{E}_{\mathbf{x}\sim\mathcal{X}}[\text{KL}(\pi_{\boldsymbol{\theta}}(\cdot|\mathbf{x})\|\pi_{\text{ref}}(\cdot|\mathbf{x}))]$$

$$=\mathbb{E}_{\mathbf{x}\sim\mathcal{X},\mathbf{y}\sim\pi_{\boldsymbol{\theta}}(\cdot|\mathbf{x})}[r(\mathbf{y};\mathbf{x}) + \eta^{-1}\log\pi_{\text{ref}}(\mathbf{y}|\mathbf{x})] + \eta^{-1}\mathbb{E}_{\mathbf{x}\sim\mathcal{X}}[\mathcal{H}(\pi_{\boldsymbol{\theta}}(\cdot|\mathbf{x}))].$$

We denote the optimal solution for the problem above as π^*. Rafailov et al. (2024a) showed that the Bradley-Terry preference model (E.2) can be rewritten as:

$$\mathbb{P}(\mathbf{y}_w \succ \mathbf{y}_l|\mathbf{x}) = \sigma\bigg(\eta^{-1}\sum_{h=1}^{|\mathbf{y}_w|}\log\frac{\pi^*(\mathbf{a}_h^w|\mathbf{s}_h^w)}{\pi_{\text{ref}}(\mathbf{a}_h^w|\mathbf{s}_h^w)} - \eta^{-1}\sum_{h=1}^{|\mathbf{y}_l|}\log\frac{\pi^*(\mathbf{a}_h^l|\mathbf{s}_h^l)}{\pi_{\text{ref}}(\mathbf{a}_h^l|\mathbf{s}_h^l)}\bigg),$$

where the state and action is defined as in the token-level MDP introduced above, with superscription $(\cdot)^w$ and $(\cdot)^l$ denoting if it is for the winner $\mathbf{y}_w$ or the loser $\mathbf{y}_l$. And maximizing the log likelihood with π^* replaced by $\pi_{\boldsymbol{\theta}}$ gives the DPO loss.

From now on we assume the horizon is fixed at H for simplicity. The derivation of the Max-Entropy RL formulation relies on the (soft) optimal value function Q^* and V^* as[3]:

$$V^*(\mathbf{s}_{H+1}) = r(\mathbf{s}_{H+1}) := r(\mathbf{y};\mathbf{x}), \text{ (reward at EOS)}$$

$$Q^*(\mathbf{s}_h, \mathbf{a}_h) = \eta^{-1}\log\pi_{\text{ref}}(\mathbf{a}_h|\mathbf{s}_h) + V^*(\mathbf{s}_{h+1}),$$

$$V^*(\mathbf{s}_h) = \eta^{-1}\log\sum_{\mathbf{a}}\exp\big(\eta Q^*(\mathbf{s}_h, \mathbf{a})\big), \text{ when } h \leq H.$$

Rafailov et al. (2024a) showed that the optimal policy π^* satisfies:

$$\eta^{-1}\log\pi^*(\mathbf{a}_h|\mathbf{s}_h) = Q^*(\mathbf{s}_h, \mathbf{a}_h) - V^*(\mathbf{s}_h)$$

$$= \eta^{-1}\log\pi_{\text{ref}}(\mathbf{a}_h|\mathbf{s}_h) + V^*(\mathbf{s}_{h+1}) - V^*(\mathbf{s}_h).$$

It can be verified that for $\mathbf{s}_1 = (\mathbf{x})$, we have $\eta V^*(\mathbf{s}_1) = \log\sum_{\mathbf{y}}\pi_{\text{ref}}(\mathbf{y}|\mathbf{x})\exp\big(\eta r(\mathbf{y};\mathbf{x})\big)$:

$$\exp\big(\eta V^*(\mathbf{s}_1)\big) = \sum_{\mathbf{a}_1}\exp\big(\eta Q^*(\mathbf{s}_1, \mathbf{a}_1)\big)$$

$$= \sum_{\mathbf{a}_1}\pi_{\text{ref}}(\mathbf{a}_1|\mathbf{s}_1)\exp\big(\eta V^*(\mathbf{s}_2)\big)$$

$$= \sum_{\mathbf{a}_1,\mathbf{a}_2}\pi_{\text{ref}}(\mathbf{a}_1|\mathbf{s}_1)\pi_{\text{ref}}(\mathbf{a}_2|\mathbf{s}_2)\exp\big(\eta V^*(\mathbf{s}_3)\big)$$

$$\dots$$

$$= \sum_{(\mathbf{a}_1,\mathbf{a}_2,\dots,\mathbf{a}_H)}\prod_{h=1}^{H}\pi_{\text{ref}}(\mathbf{a}_h|\mathbf{s}_h)\exp\big(\eta r(\mathbf{s}_{H+1})\big)$$

$$= \sum_{\mathbf{y}}\pi_{\text{ref}}(\mathbf{y}|\mathbf{x})\exp\big(\eta r(\mathbf{y};\mathbf{x})\big).$$

---

[3]Here we restated with the sequence-level reward $r(\mathbf{y};\mathbf{x})$. Rafailov et al. (2024a) started their derivation from a ground-truth token-level reward $r(\mathbf{s}_h, \mathbf{a}_h)$, which is under-specified due to the reward reshaping issue (Ng et al., 1999): reshaping the reward will not affect the Bradley-Terry preference probability so it is impossible to recover the ground-truth reward from the preference signal (Rafailov et al., 2024a, Section 4.2).

Going back to the SPPO objective (3.4) at t-th iteration, if we set $\pi_{\text{ref}} = \pi_t$ and $r(\mathbf{y}; \mathbf{x}) = \mathbb{P}(\mathbf{y} \succ \pi_t|\mathbf{x})$, we have $V^*(\mathbf{s}_1) = \eta^{-1} \log Z_{\pi_t}(\mathbf{x})$, and the learning objective at t-th iteration becomes:

$$\pi_{t+1} = \underset{\pi}{\arg\min} \, \mathbb{E}_{\mathbf{x} \sim \mathcal{X}, \mathbf{y} \sim \pi_t(\cdot|\mathbf{x})} \left(\log \left(\frac{\pi(\mathbf{y}|\mathbf{x})}{\pi_t(\mathbf{y}|\mathbf{x})} \right) - \left(\eta \mathbb{P}(\mathbf{y} \succ \pi_t|\mathbf{x}) - \log Z_{\pi_t}(\mathbf{x}) \right) \right)^2$$

$$= \underset{\pi}{\arg\min} \, \mathbb{E}_{\mathbf{s}_1 \sim \mathcal{X}, \mathbf{a}_h \sim \pi_t(\cdot|\mathbf{s}_h)} \left(\sum_{h=1}^{H} \log \frac{\pi(\mathbf{a}_h|\mathbf{s}_h)}{\pi^*(\mathbf{a}_h|\mathbf{s}_h)} \right)^2. \tag{A.1}$$

Similar to DPO, SPPO "secretly" encourages the policy $\pi_{\boldsymbol{\theta}}$ to converge to the optimal policy π^* at token level via the square loss form (A.1). Additionally, one may realize that minimizing the square-loss form is related to minimizing the KL divergence $\mathrm{KL}(\pi_{\boldsymbol{\theta}}\|\pi^*)$ via policy gradient:

$$\nabla_{\boldsymbol{\theta}} \mathrm{KL}(\pi_{\boldsymbol{\theta}}\|\pi^*) = \mathbb{E}_{\mathbf{s}_1 \sim \mathcal{X}, \mathbf{a}_h \sim \pi_{\boldsymbol{\theta}}(\cdot|\mathbf{s}_h)} \left[\left(\sum_{h=1}^{H} \log \frac{\pi_{\boldsymbol{\theta}}(\mathbf{a}_h|\mathbf{s}_h)}{\pi^*(\mathbf{a}_h|\mathbf{s}_h)} \right) \sum_{h=1}^{H} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\mathbf{a}_h|\mathbf{s}_h) \right]$$

$$= \mathbb{E}_{\mathbf{s}_1 \sim \mathcal{X}, \mathbf{a}_h \sim \pi_{\boldsymbol{\theta}}(\cdot|\mathbf{s}_h)} \left[\nabla_{\boldsymbol{\theta}} \left(\sum_{h=1}^{H} \log \frac{\pi_{\boldsymbol{\theta}}(\mathbf{a}_h|\mathbf{s}_h)}{\pi^*(\mathbf{a}_h|\mathbf{s}_h)} \right)^2 \right].$$

B   Approximating the Normalizing Factor

As discussed before, we replace the log-partition factor with a constant to avoid either estimating or predicting the log-partition factor. In hindsight, the approximation of the normalizing factor serves as a baseline for variance reduction, and does not need to be exact. Here we discuss the implicit assumptions and how we obtained an approximation based on different assumptions on human preference behaviour.

We first consider the case where we have K responses and then calculate the limit of $Z_{\widehat{\pi}_t^K}(\mathbf{x})$ when $K \to \infty$. We have two extreme cases:

1. The most "disordered" case: any preference is a fair coin flip

2. The most "ordered" case: there is a strict ordering among all responses.

The most "disordered" case   Specifically, we have K different responses $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_K$ for the given prompt $\mathbf{x}$. Since we consider the general preference setting, we assume that the preference probability between $\mathbf{y}_i$ and $\mathbf{y}_j$ ($i < j$) we observe is a fair coin toss:

$$\mathbb{P}(\mathbf{y}_i \succ \mathbf{y}_j|\mathbf{x}) = \begin{cases} 1, & \text{w.p. } 1/2, \\ 0, & \text{w.p. } 1/2. \end{cases}$$

Note that for simplicity, we assumed that the *preference probability* follows the Bernoulli distribution, not the *preference feedback*. The preference feedback is deterministic since the preference probability is either 0 or 1. Assuming $\mathbb{P}(\mathbf{y}_i \succ \mathbf{y}_j|\mathbf{x})$ follows any other $1/2$-mean distribution will yield the same constant.

We define the random variable $p_{i,j} := 2\mathbb{P}(\mathbf{y}_i \succ \mathbf{y}_j|\mathbf{x}) - 1$ for convenience. In total, we have $K(K-1)/2$ independent Rademacher random variable for all $i < j$, and then we have $p_{j,i} = -p_{i,j}$ for all $i > j$. For $i = j$, $p_{i,j} = 0$. We also define $X_i = \sum_{j=1}^{K} p_{i,j}/K$.

Given the setting and notations above, we have

$$\mathbb{P}(\mathbf{y}_i \succ \widehat{\pi}_t^K|\mathbf{x}) = \sum_{j=1}^{K} \mathbb{P}(\mathbf{y}_i \succ \mathbf{y}_j|\mathbf{x})/K = 1/2 + X_i.$$

Further,

$$Z_{\widehat{\pi}_t^K}(\mathbf{x}) = \sum_{i=1}^{K} \exp(\eta \mathbb{P}(\mathbf{y}_i \succ \widehat{\pi}_t^K|\mathbf{x}))/K = e^{\eta/2} \cdot \sum_{i=1}^{K} e^{\eta X_i}/K.$$

For any fixed i, we have the expectation as follows:

$$\mathbb{E}[e^{\eta X_i}] = \mathbb{E}\left[\prod_{j=1}^{K} e^{\eta p_{i,j}/K} \right] = \prod_{j=1}^{K} \mathbb{E}\left[e^{\eta p_{i,j}/K} \right] = \left(\frac{e^{\eta/K} + e^{-\eta/K}}{2} \right)^{K-1},$$

where the last equation comes from the definition of $p_{i,j}$ (note that $p_{i,i} = 0$). The variance is:

$$\text{Var}[e^{\eta X_i}] = \mathbb{E}[e^{2\eta X_i}] - \mathbb{E}[e^{\eta X_i}]^2 = \left(\frac{e^{2\eta/K} + e^{-2\eta/K}}{2} \right)^{K-1} - \left(\frac{e^{\eta/K} + e^{-\eta/K}}{2} \right)^{2K-2}.$$

Additionally, the covariance between $e^{\eta X_i}$ and $e^{\eta X_j}$ ($i \neq j$) is:

$$\begin{aligned}
\text{Cov}(e^{\eta X_i}, e^{\eta X_j}) &= \mathbb{E}[e^{\eta X_i + \eta X_j}] - \mathbb{E}[e^{\eta X_i}]\mathbb{E}[e^{\eta X_j}] \\
&= \mathbb{E}\left[\exp\left(\eta \sum_{k=1}^{K} p_{i,k}/K + \eta \sum_{l=1}^{K} p_{j,l}/K \right) \right] - \mathbb{E}[e^{\eta X_i}]\mathbb{E}[e^{\eta X_j}] \\
&= \left(\frac{e^{\eta/K} + e^{-\eta/K}}{2} \right)^{2K-4} - \mathbb{E}[e^{\eta X_i}]\mathbb{E}[e^{\eta X_j}] \\
&= \left(\frac{e^{\eta/K} + e^{-\eta/K}}{2} \right)^{2K-4} - \left(\frac{e^{\eta/K} + e^{-\eta/K}}{2} \right)^{2K-2},
\end{aligned}$$

where the third line holds because $p_{i,i} = p_{j,j} = 0$, $p_{i,j} + p_{j,i} = 0$, and the rest terms are i.i.d.. One can check that when $K \to \infty$, we have $\mathbb{E}[e^{\eta X_i}] \to 1$, $\text{Var}[e^{\eta X_i}] \to 0$, and $\text{Cov}(e^{\eta X_i}, e^{\eta X_j}) \to 0$. By Chebyshev's inequality, $\sum_{i=1}^{K} e^{\eta X_i}/K$ will converge to 1 in probability. So we have

$$Z_{\widehat{\pi}_t^K}(\mathbf{x}) = e^{\eta/2} \cdot \sum_{i=1}^{K} e^{\eta X_i}/K \to e^{\eta/2},$$

and we can approximate $\log Z_{\widehat{\pi}_t^K}(\mathbf{x})$ with $\eta/2$.

The most "ordered" case We assume there is an ordering $\sigma(\cdot)$ among the K different responses $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_K$ for the given prompt $\mathbf{x}$. The preference probability between $\mathbf{y}_i$ and $\mathbf{y}_j$ ($i < j$) is:

$$\mathbb{P}(\mathbf{y}_i \succ \mathbf{y}_j | \mathbf{x}) = \begin{cases} 1, & \text{if } \sigma(i) < \sigma(j), \\ 0, & \text{if } \sigma(i) > \sigma(j). \end{cases}$$

Again, the preference feedback is deterministic: as long as $\mathbf{y}_i$ is ranked higher than $\mathbf{y}_j$, $\mathbf{y}_i$ will always be preferred over $\mathbf{y}_j$. The same responses still tie: $\mathbb{P}(\mathbf{y}_i \succ \mathbf{y}_i | \mathbf{x}) = 1/2$.

Without loss of generality, we can assume $\mathbf{y}_1 \prec \mathbf{y}_2 \prec \mathbf{y}_3 \prec \cdots \prec \mathbf{y}_K$. Given the setting and notations above, we have

$$\mathbb{P}(\mathbf{y}_i \succ \widehat{\pi}_t^K | \mathbf{x}) = \sum_{j=1}^{K} \mathbb{P}(\mathbf{y}_i \succ \mathbf{y}_j | \mathbf{x})/K = \frac{i - 1 + 1/2}{K},$$

because for $\mathbf{y}_i$, there are $i - 1$ responses that are strictly worse, and $\mathbf{y}_i$ ties with itself. For the normalizing factor, we have

$$\begin{aligned}
\log Z_{\widehat{\pi}_t^K}(\mathbf{x}) &= \log\left(\sum_{i=1}^{K} \exp(\eta \mathbb{P}(\mathbf{y} \succ \widehat{\pi}_t^K | \mathbf{x}))/K \right) \\
&= \log\left(\sum_{i=1}^{K} \exp\left(\eta \frac{i - 1/2}{K} \right)/K \right) \\
&\to \log\left(\int_0^1 \exp(\eta x) dx \right) \\
&= \log \frac{e^{\eta} - 1}{\eta}.
\end{aligned}$$

where the third line (limiting) can be obtained by the squeeze theorem.
For $\eta = 1$, $\log \frac{e^{\eta}-1}{\eta} \approx 0.54\eta$. For large $\eta \approx 1e3$ as we used in the experiments, we have $\log \frac{e^{\eta}-1}{\eta} \approx \eta$.

Choice of η Depending on how "disordered" the preference is, η can vary between $\eta/2$ and η. As this paper is partially motivated by human **intransitive and irrational preference behavior**, we chose to use $\eta/2$ to approximate $\log Z_{\widehat{\pi}_t^K}(\mathbf{x})$. Fine-tuning the coefficient of this constant as a hyperparameter is also an option and can help improve performance on given dataset.

C   Proof of Theorem 3.1

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

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

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

Lemma C.1 (Theorem 1 in Freund and 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 (C.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

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

Note that π_0 is an autoregressive model that is fully supported on a finite vocabulary ($\pi_0(y_{k+1}|\mathbf{x}, \mathbf{y}_{1:k})$ has non-zero probability for every token). Because its support is a large but finite set, $|\log \pi_0(\cdot)|$ is bounded from above. So 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

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

$\square$

D   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 and Terry, 1952), which assigns a "score" representing how preferred a given response is. More recently, Rafailov et al. (2024b) 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[4]. 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., 2024b; 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

---

[4]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.

preference data. Yuan et al. (2024) proposed Self-Rewarding Language Models, where the language model itself is used to annotate preference on its 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 Soft Policy Iteration as an idealized variant of their algorithm, which reduces to the exponential weight update rule (3.1) when constrained to the bandit setting. The main difference is that they focus on the multi-round Markov decision process (MDP) in robotic and 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 the true and predicted win rate gaps, and provided theoretical guarantees on the error of finite-sample approximation. However, their practical version still utilizes the iterative-DPO framework as in Xu et al. (2023) with the DPO loss instead of their derived 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 the use of any strong external supervision besides a small-sized reward model. Another concurrent work (Gao et al., 2024) proposed REBEL, an iterative fine-tuning framework via regressing the relative reward. When applied to the preference setting, it results in a similar algorithm to our algorithm SPPO, except that SPPO approximates the log-partition factor $\log Z_{\pi_t}(\mathbf{x})$ with a constant $\eta/2$ while REBEL regresses on the win rate difference (so that $\log Z_{\pi_t}(\mathbf{x})$ is canceled). Additionally, Calandriello et al. (2024) pointed out that optimizing 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 similar to the Nash-MD algorithm.

E   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:

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

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 $(\beta = \eta^{-1})$:

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

then we have

$$\ell_{\text{DPO}}(\mathbf{y}_w, \mathbf{y}_l, \mathbf{x}) = -\log \sigma(a - b), \quad \text{(E.2)}$$
$$\ell_{\text{IPO}}(\mathbf{y}_w, \mathbf{y}_l, \mathbf{x}) = [(a - b) - 1]^2, \quad \text{(E.3)}$$
$$\ell_{\text{KTO}}(\mathbf{y}_w, \mathbf{y}_l, \mathbf{x}) = \sigma(-a + c) + \sigma(b - c) \text{ (simplified)}, \quad \text{(E.4)}$$

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).

F Full Experiment Setup

Base Model and Datasets   We follow the experimental setup of Snorkel[5], 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[6]. 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. 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[7] (Snorkel). We follow the splitting in Snorkel for a fair comparison. Additionally, we use Llama-3-8B-Instruct[8] as a stronger base model along with the same preference dataset and data splitting.

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[9] 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,

---

[5]`https://huggingface.co/snorkelai/Snorkel-Mistral-PairRM-DPO`
[6]`https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.2`
[7]`https://huggingface.co/snorkelai/Snorkel-Mistral-PairRM-DPO`
[8]`https://huggingface.co/meta-llama/Meta-Llama-3-8B-Instruct`
[9]`https://huggingface.co/llm-blender/PairRM`

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 (3.5):

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

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 trained on the first epoch of the 20k prompts from UltraFeedback to proceed to the next iteration. For both Mistral-7B-Instruct-v0.2 and Llama-3-8B-Instruct, the global training batch size is set to 64, and η is set to $1e3$. The learning 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 are 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[10]. 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 (E.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 (E.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.

- Llama-3-8B-Instruct: Llama-3-8B-Instruct is an instruction-tuned model optimized for dialogue use cases and outperforms many of the available open-source chat models on common industry benchmarks.

Benchmarks   Following previous works, we use AlpacaEval 2.0 (Dubois et al., 2024a), Arena-Hard(Li et al., 2024), MT-Bench (Zheng et al., 2024), and Open LLM Leaderboard (Beeching et al., 2023a) 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.

- **Arena-Hard** (Li et al., 2024) is a high-quality benchmark that claims to be harder and has the highest correlation and separability to Chatbot Arena among popular open-ended LLM benchmarks including AlpacaEval 2.0. We evaluate our models Mistral-PairRM-SPPO and the baseline models.

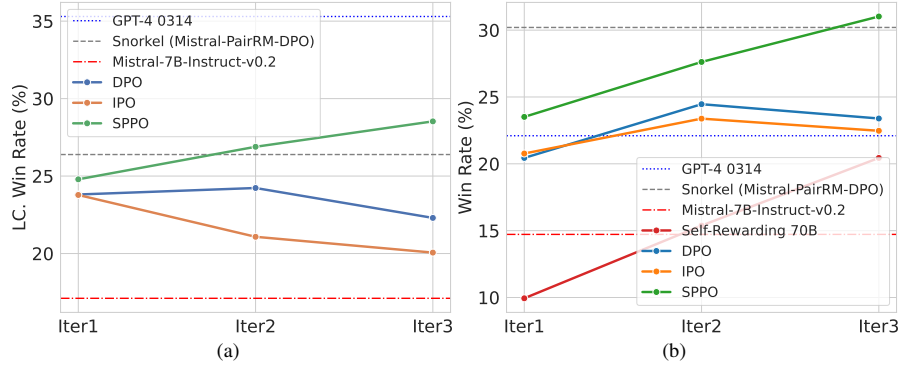[10]https://huggingface.co/snorkelai/Snorkel-Mistral-PairRM-DPO

Figure 1: 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.

- **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., 2023a) 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.

G Full Experimental Results

G.1 Additional Tables and Plots

Evaluation using GPT-4 as a judge 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), MT-Bench (Zheng et al., 2023), and Arena-Hard (Li et al., 2024) to measure the chatbot capability of our model. The results can be found in Table 2 for AlpacaEval 2.0, Figure 2 (left) for MT-Bench, and Figure 2 (right) for Arena-Hard. We found that the performance of SPPO models consistently improves throughout all iterations.

Table 2 (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, Mistral-7B-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, as illustrated in Figure 1. We also apply SPPO to a stronger baseline model, i.e., Llama-3-8B-Instruct, and the fine-tuned model Llama-3-8B-SPPO has a higher length-controlled win rate 38.77% and overall win rate 39.85%. The performance gains are more significant: 8.81% (Llama-3-8B-Instruct → Iter1), 3.42% (Iter1 → Iter2), and 3.62% (Iter2 → Iter3), summing up to a total gain of 15.85%.

Additionally, the result 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. Finally, we present the best-of-16 results for each model, selected using the PairRM reward model. We find that re-ranking with the preference model at test time can consistently improve the performance of base model (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).

Table 3: 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-SPPO Iter3 model achieves the highest LC win rate of 28.53% and a normal win rate of 31.02%. 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 and SPPO (best-of-16) achieves high win rate *without strong external supervision like GPT-4*. We additionally include the results obtained from fine-tuning Llama-3-8B-Instruct, which also show steady performance improvement.

Model	AlpacaEval 2.0		
	LC Win Rate	Win Rate	Avg. Len
Mistral-7B-Instruct-v0.2	17.11	14.72	1676
Mistral-7B-Instruct-v0.2 (best-of-16)	22.45	17.94	1529
Snorkel (Mistral-PairRM-DPO)	26.39	30.22	2736
Snorkel (Mistral-PairRM-DPO best-of-16)	29.97	34.86	2616
Self-Rewarding 70B Iter1	-	9.94	1092
Self-Rewarding 70B Iter2	-	15.38	1552
Self-Rewarding 70B Iter3	-	20.44	2552
Mistral-7B-DPO Iter1	23.81	20.44	1723
Mistral-7B-DPO Iter2	24.23	24.46	2028
Mistral-7B-DPO Iter3	22.30	23.39	2189
Mistral-7B-IPO Iter1	23.78	20.77	1693
Mistral-7B-IPO Iter2	21.08	23.38	2660
Mistral-7B-IPO Iter3	20.06	22.47	2760
Mistral-7B-SPPO Iter1	$24.79_{(+7.69)}$	$23.51_{(+8.79)}$	1855
Mistral-7B-SPPO Iter2	$26.89_{(+2.10)}$	$27.62_{(+4.11)}$	2019
Mistral-7B-SPPO Iter3	$\mathbf{28.53}_{(+1.64)}$	$\mathbf{31.02}_{(+3.40)}$	2163
Mistral-7B-SPPO Iter1 (best-of-16)	$28.71_{(+6.26)}$	$27.77_{(+9.83)}$	1901
Mistral-7B-SPPO Iter2 (best-of-16)	$31.23_{(+2.52)}$	$32.12_{(+4.35)}$	2035
Mistral-7B-SPPO Iter3 (best-of-16)	$\mathbf{32.13}_{(+0.9)}$	$\mathbf{34.94}_{(+2.82)}$	2174
Llama-3-8B-Instruct	22.92	22.57	1899
Llama-3-8B-SPPO Iter1	$31.73_{(+8.81)}$	$31.74_{(+9.17)}$	1962
Llama-3-8B-SPPO Iter2	$35.15_{(+3.42)}$	$35.98_{(+4.24)}$	2021
Llama-3-8B-SPPO Iter3	$\mathbf{38.77}_{(+3.62)}$	$\mathbf{39.85}_{(+3.87)}$	2066

Table 4: AlpacaEval 2.0 leaderboard results of both normal and length-controlled (LC) win rates in percentage (%). Mistral-7B-SPPO can outperform larger models and Mistral-7B-SPPO (best-of-16) can outperform proprietary models such as GPT-4(6/13). Llama-3-8B-SPPO exhibits even better performance.

Model	AlpacaEval 2.0	
	LC. Win Rate	Win Rate
GPT-4 Turbo	50.0	50.0
Claude 3 Opus	40.5	29.1
Llama-3-8B-SPPO Iter3	38.8	39.9
GPT-4 0314	35.3	22.1
Llama 3 70B Instruct	34.4	33.2
Mistral-7B-SPPO Iter3 (best-of-16)	32.1	34.9
GPT-4 0613	30.2	15.8
Snorkel (Mistral-PairRM-DPO best-of-16)	30.0	34.9
Mistral Medium	28.6	21.9
Mistral-7B-SPPO Iter3	28.5	31.0
Claude 2	28.2	17.2
Snorkel (Mistral-PairRM-DPO)	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

In Table 1, we compare SPPO on the AlpacaEval 2.0 leaderboard with other state-of-the-art AI chatbots. We found our SPPO model outperforms many competing models trained on proprietary alignment data (e.g., Claude 2, Gemini Pro, & Llama 3 8B Instruct). When applied to Llama 3

| 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
Mistral-7B-DPO Iter1	7.45	6.58	7.02
Mistral-7B-DPO Iter2	7.57	6.56	7.06
Mistral-7B-DPO Iter3	7.49	6.69	7.09
Mistral-7B-SPPO Iter1	7.63	6.79	7.21
Mistral-7B-SPPO Iter2	7.90	7.08	7.49
Mistral-7B-SPPO Iter3	7.84	7.34	**7.59**

Model	Arena-Hard-Auto-v0.1
Mistral-7B-Instruct	12.6
Snorkel (Mistral-PairRM-DPO)	20.7
Mistral-7B-SPPO Iter1	18.7
Mistral-7B-SPPO Iter2	20.4
Mistral-7B-SPPO Iter3	**23.3**

Figure 2: **MT-Bench & Arena-Hard Evaluation.** Left: Mistral-7B-SPPO Iter3 outperforms all baseline models by achieving an average score of 7.59 in MT-Bench. Despite initial drops in performance in the first two iterations, SPPO Iter3 improves upon the base model by the final iteration. Right: Mistral-7B-SPPO Iter3 outperforms the baseline model Snorkel(Mistral-PairRM-DPO) in Arena-Hard. The improvement across different iterations is consistent.

Table 5: **Open LLM Leaderboard Evaluation**. SPPO fine-tuning improves the base model's performance on different tasks, reaching a state-of-the-art average score of 66.75 for Mistral-7B and 70.29 for Llama-3-8B. For Mistral-7B, 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
Mistral-7B-DPO Iter1	63.14	68.39	77.19	40.33	85.25	59.41	65.62
Mistral-7B-DPO Iter2	64.16	67.84	76.09	39.95	85.23	59.03	65.38
Mistral-7B-DPO Iter3	65.19	67.89	77.27	32.30	85.49	59.00	64.52
Mistral-7B-IPO Iter1	64.68	68.60	77.98	43.75	85.08	59.04	66.52
Mistral-7B-IPO Iter2	62.12	66.30	77.51	39.20	83.15	59.70	64.66
Mistral-7B-IPO Iter3	62.97	67.12	77.51	37.45	83.69	59.57	64.72
Mistral-7B-SPPO Iter1	65.02	69.40	77.82	43.82	85.11	58.84	66.67
Mistral-7B-SPPO Iter2	65.53	69.55	77.03	44.35	85.29	58.72	**66.75**
Mistral-7B-SPPO Iter3	65.36	69.97	76.80	42.68	85.16	58.45	66.40
Llama-3-8B-Instruct	62.29	51.65	76.09	75.89	78.73	65.59	68.37
Llama-3-8B-SPPO Iter1	63.82	54.96	76.40	75.44	79.80	65.65	69.35
Llama-3-8B-SPPO Iter2	64.93	56.48	76.87	75.13	80.39	65.67	69.91
Llama-3-8B-SPPO Iter3	65.19	58.04	77.11	74.91	80.86	65.60	**70.29**

8B Instruct, our Llama-3-8B-SPPO exhibits an even higher win rate. With test-time reranking, Mistral-7B-SPPO Iter3 (best-of-16) is even competitive to GPT-4 0613 and Llama 3 70B Instruct.

In Figure 2 (left), we evaluate the performance of SPPO on MT-Bench. We can see that Mistral-7B-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.

Arena-Hard (Li et al., 2024) contains 500 challenging user queries and follow the same evaluation method as AlpacaEval 2.0. In Figure 2 (right), we evaluate the performance of SPPO on Arena-Hard. We can see that Mistral-7B-SPPO exhibits a steady performance gain across iterations. Mistral-7B-SPPO Iter 3 outperforms the baseline models, achieving an average score of 23.3.

Open LLM Leaderboard We further evaluate the capabilities of SPPO models using Huggingface Open LLM Leaderboard (Beeching et al., 2023b). This leaderboard encompasses 6 different datasets, each focusing on 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 5, 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
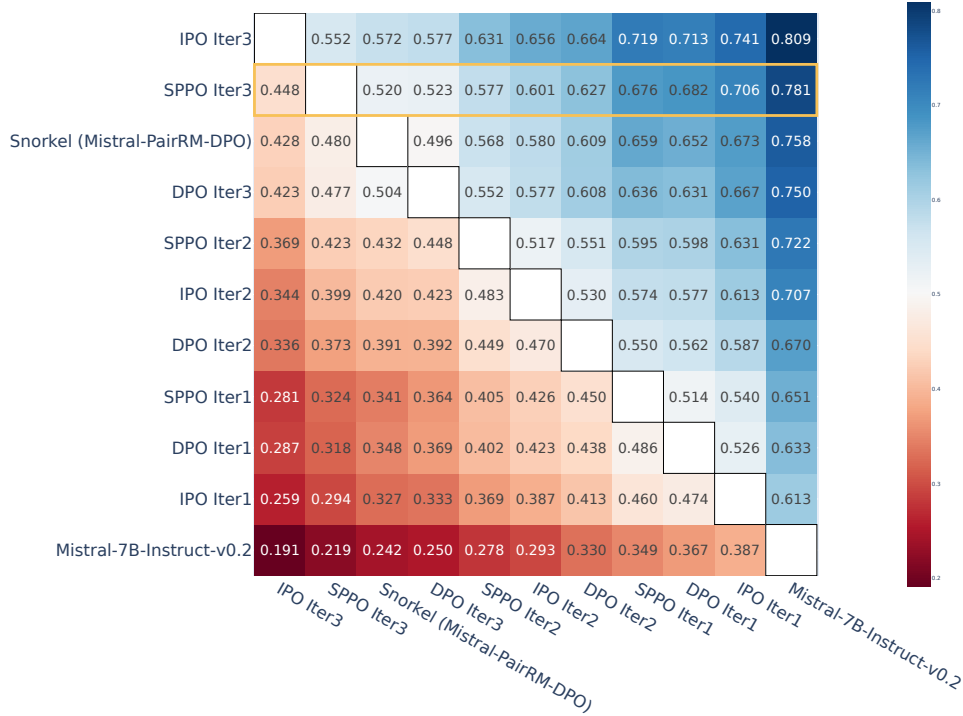
Figure 3: Pairwise win rates among base model (Mistral-7B-Instruct-v0.2), DPO models, IPO models, and SPPO models using **PairRM-0.4B** as a judge, which may favor models with longer outputs. On benchmarks with more powerful judge models (e.g., GPT-4), such as AlpacaEval 2.0 and MT-Bench, SPPO outperforms other baseline algorithms by a large margin.

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.

G.2 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 2), 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.

G.3 Ablation Study

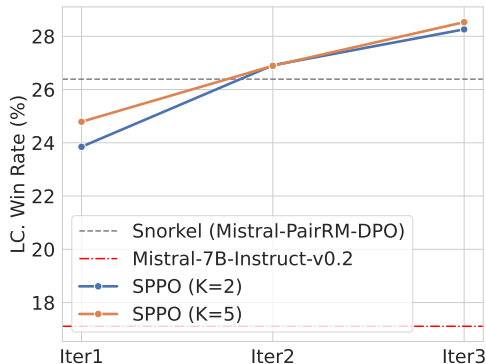| Mini-Batch Size	Iteration	AlpacaEval 2.0		Avg. Len (chars)
		Win Rate		
		LC.	Raw	
$K=2$	Iter1	23.85	23.53	1948
	Iter2	26.91	27.24	1999
	Iter3	28.26	28.22	1961
$K=5$	Iter1	24.79	23.51	1855
	Iter2	26.89	27.62	2019
	Iter3	**28.53**	**31.02**	2163

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 $\mathbb{P}(\mathbf{y} \succ \pi_t | \mathbf{x})$.

H Response Examples in Different Iterations

Table 6: 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 7: 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.