Preference Optimization via Contrastive Divergence: Your Policy Is Secretly an NLL Estimator

Zhuotong Chen, Fang Liu, Xuan Zhu, Haozhu Wang, Jiayu Li, Yanjun Qi, Mohammad Ghavamzadeh

Amazon Web Services
zhuotongchen@gmail.com, zhuxuan@amazon.com

Abstract

Existing studies on preference optimization (PO) have been focused on constructing pairwise preference data following simple heuristics, such as maximizing the margin between chosen and rejected responses based on human (or AI) ratings. In this work, we develop a novel PO framework that provides theoretical guidance to effectively sample rejected responses. To achieve this, we formulate PO as minimizing the negative log-likelihood (NLL) of a probability model and propose a sampling-based solution to estimate its normalization constant via contrastive divergence. We show that these estimative samples can act as rejected responses in PO. Leveraging the connection established between PO and NLL estimation, we propose a novel PO algorithm, called Monte-Carlo-based PO (MC-PO), that applies a MC kernel to sample *hard negatives* w.r.t. the log-likelihood of the target policy. Intuitively, these hard negatives represent the rejected samples that are most difficult for the current policy to differentiate. We show that MC-PO outperforms existing SOTA baselines on popular alignment benchmarks.

1 Introduction

While large language models (LLMs) learn a broad world knowledge, aligning their behavior precisely with human values is challenging due to the unsupervised nature of their training. Reinforcement learning from human feedback (RLHF) (Ouyang et al. 2022) has emerged as a class of effective algorithms to align LLMs (Schulman et al. 2017). Recent works on direct preference optimization (DPO) (Rafailov et al. 2024) and its variants (e.g., identity preference optimization (Azar et al. 2024)) directly optimize an LLM to adhere to human values, without explicit reward modeling or RL. Data for these algorithms are often collected in the form of preferences (Ziegler et al. 2019). This leads to more consistent labeling across human annotators as it reduces their cognitive load and avoids the need for absolute judgment, which can be more subjective. Existing studies on PO have predominantly considered creating pairwise preference data using simple heuristics, such as choosing a rejected response by maximizing the gap with the chosen response in terms of human (or AI) ratings (Tunstall et al. 2023; Lambert et al. 2024). In this work, we focus on improving the performance of PO by improving the quality of their preference-based training data. Specifically, we ask the question: *"How to choose rejected / dispreferred responses for PO?"*

To answer this question, we develop a novel PO framework that provides theoretical guidance on effective sampling strategies for selecting rejected responses. To achieve this, we formulate learning the optimal policy in RLHF as minimizing the negative log-likelihood (NLL) of a parametrized probability model. A key challenge in this formulation arises from the presence of a normalization constant expressed as an intractable integral. To overcome this challenge, we propose to estimate the gradient of this log-normalization constant using contrastive divergence (CD) (Hinton 2002), which leverages empirical samples to approximate gradients. We further demonstrate that this sampling-based solution is connected to the PO framework, where the generated samples can be interpreted as rejected responses for PO. This insight establishes a connection between NLL estimation and PO, allowing us to incorporate sampling strategies from the NLL literature to guide the selection of rejected responses in PO. Hence, these strategies, originally developed to improve the accuracy of the NLL gradient estimation, can improve PO performance by producing rejected samples of higher quality.

We leverage the connection between PO and NLL estimation, and propose a novel PO algorithm that applies a special kernel to generate rejected responses. We refer to our algorithm as Monte-Carlo-based Preference Optimization (MC-PO). More specifically, MC-PO selects a rejected response (or rejected responses) for an input prompt from a set of candidates by sampling in proportion to the log-likelihood of the target policy (the policy being trained). This procedure often results in generating a *hard negative*, i.e., a rejected response that closely resembles the chosen one, posing a greater challenge to the policy to differentiate between them. As illustrated in Figure 1, given a chosen response (Rank-1) and a set of rejected candidates (Rank-$2, 3, 4$) for the same input prompt, existing studies often select the most dispreferred response (e.g., Rank-4) to pair with the chosen one. MC-PO can be considered as a hybrid PO algorithm operating between offline and online approaches. In the offline PO setting, both chosen and rejected responses are fixed prior to training, whereas online PO generates training data for each batch dynamically using the current policy. MC-PO circumvents the computational overhead of generating responses

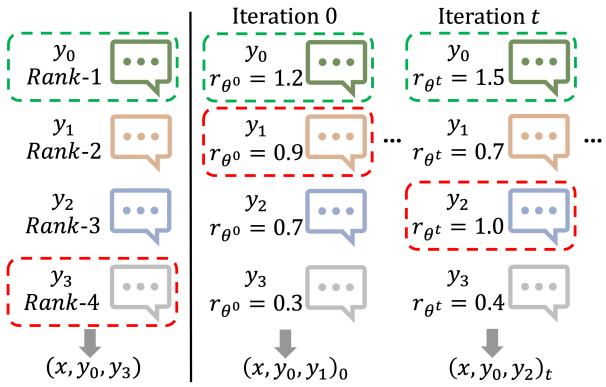


Figure 1: **Left:** Existing methods often choose a rejected response as the one that maximizes the gap with the chosen response based on human/AI ratings. **Right:** We propose theoretical guidance to sample rejected responses proportionally to the parameterized implicit reward model, which is a function of the log-likelihood of the target policy (policy being trained). Note that the sampling of rejected responses evolves during as the reward parameters get updated during training.

during training while still utilizing the current policy to select rejected responses. As shown in Figure 1, from iteration 0 to t, even for the same input prompt, MC-PO may change the rejected response as the policy evolves during training. We summarize our main contributions below.

- We propose a novel PO framework that offers theoretical guidance for designing sampling strategies to generate rejected responses. This framework is grounded in a reformulation of learning the optimal policy as an NLL estimation problem, which we solve using a CD method.
- We introduce MC-PO, a hybrid PO algorithm motivated by the connection between PO and NLL estimation. MC-PO dynamically selects a rejected response for an input prompt by sampling from a preconstructed set of candidates in proportion to their log-likelihood under the current policy. This approach avoids the computational overhead of online data generation, while still leveraging the current policy in constructing the training dataset.
- We perform benchmark evaluations and demonstrate that MC-PO outperforms existing SOTA baselines. In addition, we perform detailed ablation studies to assess the effectiveness of different sampling strategies and show that MC-PO consistently leads to improved performance.

2 Preliminaries

In this section, we briefly revisit the key concepts of PO (Section 2.1), NLL estimation and CD (Section 2.2), which we will build upon in the subsequent sections.

2.1 Preference Optimization

The standard RLHF paradigm (Christiano et al. 2017; Stiennon et al. 2020) consists of two main stages: (i) learning a reward model (RM) from human preferences and (ii) policy optimization using the learned RM. Given a preference dataset $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_{0,i}, \mathbf{y}_{1,i})\}$ in which $\mathbf{x}$ is an input prompt, and $\mathbf{y}_0$ and $\mathbf{y}_1$ are the chosen and rejected responses, the RM is learned by optimizing the following cross-entropy loss:

$$\min_{\phi} \mathbb{E}_{(\mathbf{x},\mathbf{y}_0,\mathbf{y}_1)\sim\mathcal{D}}\big[-\log\sigma\big(r_\phi(\mathbf{x},\mathbf{y_0}) - r_\phi(\mathbf{x},\mathbf{y}_1)\big)\big]. \quad (1)$$

Then, the parameter of the policy is optimized as,

$$\max_{\boldsymbol{\theta}} \mathbb{E}_{\substack{\mathbf{x}\sim\rho \\ \mathbf{y}\sim\pi_{\boldsymbol{\theta}}(\cdot|\mathbf{x})}}\big[r_\phi(\mathbf{x},\mathbf{y})\big] - \beta\cdot\mathrm{KL}\big[\pi_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x})||\pi_{\mathrm{ref}}(\mathbf{y}|\mathbf{x})\big], \quad (2)$$

where ρ is the distribution over prompts and β is a hyperparameter controlling the deviation from a reference policy π_{ref} (often the SFT policy π_{SFT}). The RM is typically estimated from a finite dataset, and thus, cannot be accurate for all prompt-response pairs. The KL-divergence uses the reference policy as a guardrail to prevent the model from overfitting the estimated RM (Skalse et al. 2022).

The KL-constrained reward maximization in (2) has the following closed-form solution (Go et al. 2023):

$$\pi^*(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})}\pi_{\mathrm{ref}}(\mathbf{y}|\mathbf{x})\exp\Big(\frac{1}{\beta}r_\phi(\mathbf{x},\mathbf{y})\Big), \quad (3)$$

where the partition function $Z(\mathbf{x})$ ensures that $\pi^*(\cdot|\mathbf{x})$ is a valid conditional probability distribution. Since the response space is combinatorially large, $Z(\mathbf{x})$ is typically intractable to compute. This makes the policy representation in (3) hard to utilize in practice.

An alternative approach to the RLHF paradigm is DPO (Rafailov et al. 2024). To address the challenge of solving the RL problem in (2), DPO rearranges it to write the RM r_ϕ in terms of its corresponding optimal policy π^*, and then substitutes this expression into the RM loss (1) to estimate the optimal policy by solving the following optimization:

$$\min_{\theta} \mathop{\mathbb{E}}_{(\mathbf{x},\mathbf{y}_0,\mathbf{y}_1)\sim\mathcal{D}}\left[-\log\sigma\left(r_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{y}_0) - r_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{y}_1)\right)\right], \quad (4)$$

where $r_{\boldsymbol{\theta}} := \beta\log\frac{\pi_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x})}{\pi_{\mathrm{ref}}(\mathbf{y}|\mathbf{x})}$ is the parametrized reward model, σ is the logistic function. Using (4), the target policy $\pi_{\boldsymbol{\theta}}$ is optimized to distinguish between the chosen $\mathbf{y}_0$ and rejected $\mathbf{y}_1$ responses for each input prompt $\mathbf{x}$.

After DPO, many PO algorithms have been developed with similar style. Based on their objective functions, these algorithms can be categorized into two groups: *contrastive* and *classification*-based. Contrastive methods aim to maximize the difference in predicted likelihoods between the chosen and rejected responses. Representative contrastive-based algorithms include DPO (Rafailov et al. 2024), IPO (Azar et al. 2024), SimPO (Meng, Xia, and Chen 2024), and RPO (Liu et al. 2024b). In contrast, classification-based algorithms perform separate maximization and minimization over the chosen and rejected responses, such as KTO (Ethayarajh et al. 2024), BCO (Jung et al. 2024), and NCA (Chen et al. 2024a).

2.2 Negative Log-Likelihood Estimation and Contrastive Divergence

NLL Estimation. Unnormalized models can be used to approximate complex data distributions. However, estimating unnormalized models is not straightforward since the NLL estimation involves the typically intractable normalization constant. Given i.i.d. observations from some data distribution, we seek to approximate it with a parametric probability

model $p_{\boldsymbol{\theta}}$,

$$p_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x}) := \frac{\tilde{p}_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x})}{Z_{\boldsymbol{\theta}}(\mathbf{x})}, \text{ where } Z_{\boldsymbol{\theta}}(\mathbf{x}) = \int \tilde{p}_{\boldsymbol{\theta}}(\mathbf{y}'|\mathbf{x}) d\mathbf{y}', \tag{5}$$

where $\tilde{p}_{\boldsymbol{\theta}}$ is the unnormalized model and $Z_{\boldsymbol{\theta}}(\mathbf{x})$ is its normalization constant. The NLL estimation minimizes the negative log-likelihood of $p_{\boldsymbol{\theta}}$ to predict the ground-truth observations. Generally speaking, as the number of observations approaches infinity, the NLL estimation results in a parametric probability model that increasingly approximates the target distribution. However, computing the log-normalization constant is challenging because of the integration.

CD. Optimizing the probability model $p_{\boldsymbol{\theta}}$ in (5) requires computing the intractable gradient of the log-normalization constant. CD uses MCMC methods to estimate this gradient (Hinton 2002). Specifically, it initiates MCMC sampling from ground-truth observation rather than from a random state, which allows the sampling process to converge faster. The sampling process involves a small number of MCMC steps (often just one), making it particularly effective for probability models where the normalization constant cannot be easily computed.

3 Preference Optimization as Sampling-based Solution to NLL Estimation

In this section, we formulate learning the optimal policy in RLHF as an NLL estimation (Section 3.1). Subsequently, we adopt CD as a sampling-based solution to estimate the intractable term in this NLL estimation and its gradient (Section 3.2). This perspective enables a reinterpretation of PO as NLL estimation, and allows us to adapt sampling-based algorithms from the NLL literature to select rejected responses in PO (Section 3.3).

3.1 RLHF as NLL Estimation

We use the expression for the optimal policy in (3) and define the following parametric probability model for π^*:

$$p_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x}) := \frac{1}{Z_{\boldsymbol{\theta}}(\mathbf{x})} \mu(\mathbf{y}|\mathbf{x}) \exp\big(r_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y})\big), \tag{6}$$

$$Z_{\boldsymbol{\theta}}(\mathbf{x}) = \int \mu(\mathbf{y}'|\mathbf{x}) \exp\big(r_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y}')\big) d\mathbf{y}',$$

where $r_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y}) := \beta \log \frac{\pi_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x})}{\pi_{\text{ref}}(\mathbf{y}|\mathbf{x})}$ is the implicit reward of DPO (Rafailov et al. 2024) and μ is a proposal distribution from which we can sample. For any set of parameters $\boldsymbol{\theta}$, we assume that $p_{\boldsymbol{\theta}}$ covers the support of π^*, such that $p_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x}) > 0$ whenever $\pi^*(\mathbf{y}|\mathbf{x}) > 0,\ \forall \mathbf{x} \sim \rho$. This sets conditions on the proposal distribution μ. For instance, in general LLM alignment setting, we can choose μ to be the reference policy which satisfies the above requirement.

To estimate π^* with $p_{\boldsymbol{\theta}}$, the NLL estimator minimizes the NLL of $p_{\boldsymbol{\theta}}$, i.e.,

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \mathbb{E}_{\mathbf{x}\sim\rho,\ \mathbf{y}\sim\pi^*(\cdot|\mathbf{x})}\big[\mathcal{L}^{\text{NLL}}(\boldsymbol{\theta}, \mathbf{x}, \mathbf{y})\big], \tag{7}$$

$$\mathcal{L}^{\text{NLL}}(\boldsymbol{\theta}, \mathbf{x}, \mathbf{y}) = -r_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y}) + \log Z_{\boldsymbol{\theta}}(\mathbf{x}).$$

Optimizing the model parameters $\boldsymbol{\theta}$ using gradient descent requires calculating the gradient of the NLL estimator, which can be derived as (see Appendix A.1 for the detailed derivation)

$$\begin{aligned} &\nabla_{\boldsymbol{\theta}} \mathcal{L}^{\text{NLL}}(\boldsymbol{\theta}, \mathbf{x}, \mathbf{y}) \\ &= -\nabla_{\boldsymbol{\theta}} r_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y}) + \nabla_{\boldsymbol{\theta}} \log Z_{\boldsymbol{\theta}}(\mathbf{x}), \\ &= -\nabla_{\boldsymbol{\theta}} r_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y}) + \mathbb{E}_{p_{\boldsymbol{\theta}}(\mathbf{y}'|\mathbf{x})}\big[\nabla_{\boldsymbol{\theta}} r_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y}')\big]. \end{aligned} \tag{8}$$

The first term on the RHS of (8) is typically easy to compute, because it is the gradient of the target policy (e.g., the LLM) which can be computed using an automatic differentiation software. However, the second term is intractable because it involves an expectation over the probability model $p_{\boldsymbol{\theta}}$ in (6). In the next section, we show how we use CD to approximate the gradient expression in (8).

Algorithm 1: MC kernel $K_{\boldsymbol{\theta}}(\mathbf{y}_z|\mathbf{x}, \mathbf{y}_0)$

Input: $\mathbf{x}$, $\mathbf{y}_0$

1: Sample $\{\mathbf{y}_i\}_{i=1}^M$ from μ
2: Compute $\{w_i\}_{i=0}^M,\ w_i = \frac{\exp(r_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y}_i))}{\sum_{j=0}^M \exp(r_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y}_j))}$
3: Sample $z \sim \text{Categorical}\big([w_0, w_1, ..., w_M]\big)$
4: **return** $\mathbf{y}_z$

3.2 A CD Solution to Solve the NLL Estimation of RLHF

We now propose a CD-based method to approximate the gradient of the NLL estimation in (8). As described in Section 2.2, CD estimates the gradient of the log-normalization constant by applying an MC kernel to generate samples. Specifically, we apply CD with the MC kernel $K_{\boldsymbol{\theta}}(\mathbf{y}'|\mathbf{x}, \mathbf{y}_0)$ defined in Algorithm 1 to estimate the gradient of the log-normalization constant in (8).

$$\mathbb{E}_{p_{\boldsymbol{\theta}}(\mathbf{y}'|\mathbf{x})}\big[\nabla_{\boldsymbol{\theta}} r_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y}')\big] \approx \mathbb{E}_{K_{\boldsymbol{\theta}}(\mathbf{y}'|\mathbf{x}, \mathbf{y}_0)}\big[\nabla_{\boldsymbol{\theta}} r_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y}')\big]. \tag{9}$$

This kernel generates samples proportionally to the implicit reward $r_{\boldsymbol{\theta}}$. As a result, these samples have high likelihood under $p_{\boldsymbol{\theta}}$. With this particular choice of kernel, we can approximate the intractable term in the NLL gradient (8) as described in the following proposition.

Proposition 3.1. *Let $\mathbf{y}_0 \sim \pi^*(\cdot|\mathbf{x})$ and $\{\mathbf{y}_i\}_{i=1}^M$ be M noise samples from a proposal distribution $\mu(\cdot|\mathbf{x})$. The CD approximation of the log-normalization constant in Eq.* (9) *can be computed as*

$$\mathbb{E}_{K_{\boldsymbol{\theta}}(\mathbf{y}'|\mathbf{x}, \mathbf{y}_0)}\big[\nabla_{\boldsymbol{\theta}} r_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y}')\big] = \nabla_{\boldsymbol{\theta}} \log \sum_{i=0}^{M} \exp\big(r_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y}_i)\big). \tag{10}$$

We report the proof of this proposition in Appendix A.2. It is straightforward to see that the approximation to (8) suggested by Proposition 3.1 is the gradient of the following loss:

$$\mathcal{L}^{\text{CD}}(\boldsymbol{\theta}, \mathbf{x}, \mathbf{y}) = -r_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y}) + \log \sum_{i=0}^{M} \exp\big(r_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y}_i)\big). \tag{11}$$

Thus, the loss in (11) can be seen as the CD approximation of the NLL estimation (7). Comparing (11) with the NLL estimation in (7), we notice that the normalization constant has been approximated as $Z_{\boldsymbol{\theta}}(\mathbf{x}) \approx \sum_{i=0}^{M} \exp\big(r_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y}_i)\big)$

using samples $\mathbf{y}_0 \sim \pi^*$ and $\{\mathbf{y}_i\}_{i=1}^M \sim \mu(\cdot|\mathbf{x})$. The CD loss, $\mathcal{L}^{\text{CD}}$, is equivalent to a cross-entropy loss that optimizes the model to classify $\mathbf{y}_0$ as the correct prediction among all $(M+1)$ samples. Specifically,

$$\mathcal{L}^{\text{CD}}(\boldsymbol{\theta}, \mathbf{x}, \mathbf{y}_0) = -\log\Big(\frac{\exp(r_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y}_0))}{\sum_{i=0}^{M} \exp\big(r_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y}_i)\big)}\Big). \quad (12)$$

In the above CD approximations to NLL and its gradient, it is assumed that the chosen response $\mathbf{y}_0$ is sampled from the optimal policy π^*, which is generally unknown. To address this, we show that when $\mathbf{y}_0$ is sampled from the probability model $p_{\boldsymbol{\theta}}$, the CD approximation leads to an unbiased estimate for the gradient of the log-normalization constant. This motivates us to sample $\mathbf{y}_0$ in proportion to the probability model $p_{\boldsymbol{\theta}}$, rather than from π^*.

Proposition 3.2. *Let* $\hat{Z}_{\boldsymbol{\theta}}(\mathbf{x}) := \frac{1}{M+1}\sum_{i=0}^{M} \exp\big(r_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y}_i)\big)$ *be an approximation of the normalization constant* $Z_{\boldsymbol{\theta}}(\mathbf{x})$*, where* $\mathbf{y}_0 \sim p_{\boldsymbol{\theta}}(\cdot|\mathbf{x})$*. Then, we have*

$$\mathbb{E}_{p_{\boldsymbol{\theta}}(\mathbf{y}_0|\mathbf{x})\mu(\{\mathbf{y}_i\}_{i=1}^M|\mathbf{x})}\big[\nabla_{\boldsymbol{\theta}} \log \hat{Z}(\mathbf{x})\big] = \nabla_{\boldsymbol{\theta}} \log Z_{\boldsymbol{\theta}}(\mathbf{x}),$$

$$\textit{where} \quad \mu(\{\mathbf{y}_i\}_{i=1}^M|\mathbf{x}) = \prod_{i=1}^{M} \mu(\mathbf{y}_i|\mathbf{x}).$$

Notice that the additional constant $\frac{1}{M+1}$ from $\hat{Z}_{\boldsymbol{\theta}}(\mathbf{x})$ is removed by taking the gradient of log. Proposition 3.2, whose proof is reported in Appendix A.4, provides a feasible CD approximation to the NLL estimate and its gradient.

3.3 PO as Sampling-based Solution to NLL Estimation of RLHF

In this section, we show that the sampling-based solution to the NLL estimation is related to the PO formulation (4) by considering the generated samples as rejected responses. Writing the CD loss in (11) for $M = 1$, i.e., only a single sample from the proposal distribution $\mu(\cdot|\mathbf{x})$, we obtain

$$\begin{aligned} &- r_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y}_0) + \log \sum_{i=0}^{1} \exp\big(r_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y}_i)\big) \\ &= -\log \sigma\big(r_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y}_0) - r_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y}_1)\big), \end{aligned}$$

which is exactly the DPO objective in (4) with $\mathbf{y}_0$ and $\mathbf{y}_1$ as chosen and rejected responses. This shows that using a single noise sample, the sampling-based estimation of NLL is equivalent to the DPO objective. Moreover, this equivalence can be generalized to the case of $M > 1$ where multiple rejected responses are used for PO training.

This provides a novel interpretation of rejected responses in PO: *Rejected responses can be understood as samples used to estimate the normalization constant in NLL estimation.* Building on this bridge between NLL and PO, we can adapt various sampling-based algorithms from the literature of NLL estimation to select rejected samples for PO. These algorithms aim to improve the accuracy of estimating the normalization constant, thereby improving the PO performance. For example, in Proposition 3.1, the M noise samples are drawn randomly from a proposal distribution $\mu(\cdot|\mathbf{x})$. In the next section, we develop a more advanced sampling strategy and use it to implement our PO algorithm.

4 MC-PO Algorithm

Based on the connection we established between PO and sampling-based solutions to NLL estimation in Section 3, we now derive a PO algorithm that uses CD to generate the rejected responses in its training set. We refer to our algorithm as Monte-Carlo-based Preference Optimization (MC-PO). Before describing MC-PO, we state two results that highlight some characteristics of this algorithm. We first show that CD leads to an unbiased gradient estimator for the NLL estimation by setting the proposal distribution μ equal to the probability model $p_{\boldsymbol{\theta}}$.

Proposition 4.1. *Let the proposal distribution be equal to the probability model, i.e.,* $\mu = p_{\boldsymbol{\theta}}$*. Then, the CD estimator of the gradient of the log-normalization constant in* (10) *is unbiased.*

We report the proof in Appendix A.5. Proposition 4.1 suggests to draw the noise samples from the probability model $p_{\boldsymbol{\theta}}$. Recall that the kernel $K_{\boldsymbol{\theta}}(\mathbf{y}_z|\mathbf{x}, \mathbf{y}_0)$ defined in Algorithm 1 aims to approximate the sampling from $p_{\boldsymbol{\theta}}$. Following this, we apply the MC kernel defined in Algorithm 1 that samples rejected responses proportionally to the implicit reward $r_{\boldsymbol{\theta}}$. Since the implicit reward $r_{\boldsymbol{\theta}} := \beta \log \frac{\pi_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x})}{\pi_{\text{ref}}(\mathbf{y}|\mathbf{x})}$, which indicates that rejected responses with higher log-likelihood with respect to $\pi_{\boldsymbol{\theta}}$ (hard negatives) are preferred.

Another characteristic of MC-PO is related to its sampling strategy, which is proportional to the implicit reward $r_{\boldsymbol{\theta}}$. We explain the advantage of MC-PO's sampling strategy using its gradient expression. For $M = 1$, we can write the gradient of the CD loss in (11) as

$$\begin{aligned} \nabla_{\boldsymbol{\theta}} \mathcal{L}^{\text{CD}}(\boldsymbol{\theta}, \mathbf{x}, \mathbf{y}_0) = &-\sigma\big(r_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y}_1) - r_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y}_0)\big) \\ &\nabla_{\boldsymbol{\theta}}\big(r_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y}_0) - r_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y}_1)\big), \end{aligned}$$

where $\mathbf{y}_0$ and $\mathbf{y}_1$ are chosen and rejected responses, respectively. It can be seen that sampling in proportion to the implicit reward minimizes the gap between the rewards of the chosen and rejected responses, thereby increasing the magnitude of the gradient. In contrast, when the rejected response receives a low reward, the scaling factor in the gradient expression becomes small, leading to less effective parameter updates.

Implementation. MC-PO can be considered as a hybrid PO algorithm as it combines aspects of both offline and online PO algorithms. In the fully offline PO setting, chosen and rejected responses are predetermined prior to training (Rafailov et al. 2024). In contrast, the online PO setting generates responses dynamically for each batch using the current checkpoint (policy) during training (Qi et al. 2024; Xiong et al. 2024). MC-PO takes a middle ground: it avoids the computational expense of generating responses during training while still leveraging the current policy (checkpoint) to select rejected responses.

Given a dataset in which each input prompt $\mathbf{x}$ is paired with a chosen response $\mathbf{y}_0$ and a set of M rejected response candidates $\{\mathbf{y}_i\}_{i=1}^M$, MC-PO applies the MC kernel defined in Algorithm 1 to select a rejected response from this set. The

MC kernel uses the current checkpoint $\pi_{\boldsymbol{\theta}}$ to compute weights $\{w_i\}_{i=0}^{M}$, and then samples from the resulting categorical distribution, as described in Algorithm 1. This process yields a rejected response as output which MC-PO pairs it with the chosen response $\mathbf{y}_0$ to form its training set for the next iteration.

5 Related Work

There has been a growing body of work focused on enhancing PO performance through improving its training data quality. An early effort in this direction introduces the use of on-policy data generation together with rejection sampling to approximate the optimal policy, thereby improving the effectiveness of DPO (Liu et al. 2024a). Building upon this idea, a series of concurrent results proposed different sampling strategies to construct pairwise preference data in an online or an iterative batch learning setting, where the target policy alternates between sampling new completions and updating via DPO training (Xiong et al. 2024; Xu et al. 2023; Dong et al. 2024; Xie et al. 2024; Guo et al. 2024; Chen et al. 2024b; Shi, Zhou, and Du 2024). These iterative methods proposed several sampling strategies to generate higher-quality preference data for DPO. For instance, (Xu et al. 2023; Dong et al. 2024) advocate using a best-of-n versus worst-of-n approach, where multiple completions are sampled and the chosen and rejected completions are selected to form a training pair. (Xie et al. 2024) introduces a value bonus mechanism that guides the choice of rejected completions by biasing toward exploratory completions, inspired by the optimism in the face of uncertainty principle in online learning. In contrast, (Guo et al. 2024) adheres to strict on-policy sampling without additional sampling heuristics, focusing on a more theoretically grounded approach. (Chen et al. 2024b) proposes a semi-supervised strategy in which a high-quality supervised fine-tuning dataset provides the chosen completions, while rejected completions are sampled from the target policy, creating pairwise preference data that blends human supervision and policy outputs.

In this work, we investigate the impact of different sampling strategies in offline PO by reformulating PO as an NLL estimation problem. This perspective reveals that rejected completions act as noise samples used to estimate the gradient of the log-normalization constant. By establishing this connection, our reformulation enables the application of various sampling techniques from the NLL literature to guide the selection of rejected completions during PO training.

6 Experiments

In this section, we present the main results of our experiments, highlight the effectiveness of MC-PO on various benchmarks (Section 6.2) and provide an in-depth understanding of the effect of different sampling strategies (Section 6.3). Additional results are presented in Appendix C.

6.1 Experimental Setup

In this section, we summarize the setup used in our experiments. More details are in Appendix B.

Models and datasets. We perform PO under **two** different setups: **(1) Base setup** considers the Mistral-7B-SFT and Llama-3.1-8B-SFT model, which has been fine-tuned using supervised next-word prediction on the TÜLU 3 SFT Mix dataset (Lambert et al. 2024). We fine-tune these models on the Nectar dataset (Zhu et al. 2023). The Nectar dataset consists of 7 ranked responses generated by different LLMs per input prompt, which creates high-quality and diverse candidate responses for sampling. For each input prompt, we consider the rank-1 response as the chosen response, and sample a rejected response from the remaining candidates. **(2) Instruct setup** uses the off-the-shelf instruction-tuned Llama-3.1-8B-Instruct model (Dubey et al. 2024) to initialize the target policy $\pi_{\boldsymbol{\theta}}$. This model has undergone extensive instruction-tuning processes, making it more expressive compared to the initial model in the base setup. We use prompts from UltraFeedback (Cui et al. 2024) and generate chosen and rejected responses using the Llama-3.1-8B-Instruct model. This makes the instruct setup closer to an on-policy setting (Tang et al. 2024). Specifically, we generate 6 responses using temperatures 0.6, 0.8, and 1 for each input prompt (2 responses for each temperature). Then, we apply the iterative pairwise ranking approach (Chen et al. 2024c) to select the chosen response and sample a rejected response from the remaining candidates with different strategies. The iterative pairwise ranking method determines the overall winner from a group of candidates by comparing two responses at a time. Each comparison is conducted using a teacher model (in this work, we use Llama-3.1-70B-Instruct).

Sampling. For both setups, we randomly sample a rejected response from all candidates for all baseline methods. The proposed MC-PO applies the kernel defined in Algorithm 1 to sample a rejected response.

Proposal distribution. The baseline setup utilizes responses from the Nectar dataset, which are pre-generated by various LLMs. In this context, the mixture of these LLMs serves as the proposal distribution. The instruct setup applies the off-the-shelf instruction-tuned Llama-3.1-8B-Instruct model model to generate multiple responses, in this case, the Llama-3.1-8B-Instruct model serves as the proposal distribution.

Evaluations. To evaluate the performance of the models after alignment, we use two popular open-ended instruction-following benchmarks: AlpacaEval 2 (Dubois et al. 2024) and Arena-Hard (Li et al. 2024). These benchmarks assess the model's versatile conversational capabilities across a wide range of queries and have been widely adopted by the community. We use win-rate as the evaluation metric. Let N_{cand}, N_{base}, and N_{tie} be the number of candidate's wins, baseline's wins, and ties, respectively. We compute the adjusted win-rate as

$$\text{Winrate} := \frac{N_{\text{cand}} + N_{\text{tie}}/2}{N_{\text{cand}} + N_{\text{base}} + N_{\text{tie}}}.$$

All the win-rate-based evaluations are done using Mistral-Large-Instruct-2407 as the judge model.

Training. All jobs are trained using full-parameter tuning. We set the effective batch size to 128 and the number of

training epochs to 2. Hyperparameter optimization is done using 7 different learning rates. All results are reported as the average of the final checkpoints across 3 random seeds, along with the standard deviation, which can effectively reduce numerical randomness (Miller 2024). Finally, each training job is done on a node of 8·A100 GPUs and a cluster of multiple AWS p4d nodes is used to complete all experiments.

6.2 Main Results: Comparison with SOTA PO Algorithms

We compare MC-PO with several existing offline PO algorithms. These baselines belong to either contrastive or classification-based categories described in Section 2.1. The contrastive ones are **DPO** (Rafailov et al. 2024), **RPO** (Liu et al. 2024b), **EXO** (Ji et al. 2024), **SimPO** (Meng, Xia, and Chen 2024), and **CPO** (Xu et al. 2024), and the classification-based ones are **BCO** (Jung et al. 2024), **KTO** (Ethayarajh et al. 2024), **APO** (D'Oosterlinck et al. 2024), **SPPO** (Wu et al. 2024), and **NCA** (Chen et al. 2024a).

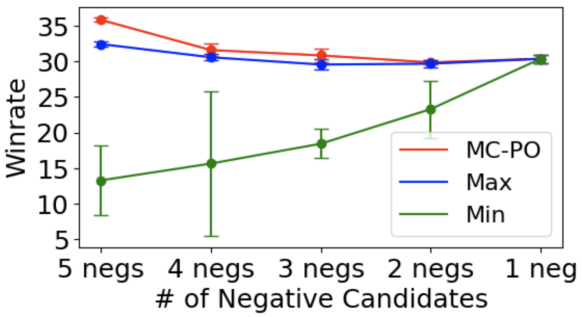
The main results are summarized in Table 1. MC-PO outperforms existing baselines in five out of six studies. Notably, in the base setup with Mistral-7B-SFT, MC-PO performs better than DPO by 4.5% and 9% in Alpaca-Eval and Arena-Hard, respectively. Using Llama-3.1-8B-SFT, MC-PO achieves win-rates 35.84% and 63.77% in Alpaca-Eval and Arena-Hard, outperforming all baselines. In the instruct setup, MC-PO outperforms almost all baselines (performs similarly to CPO), but the margin is not as significant as observed in the base setup experiments. This could be related to the low diversity of the candidates in this setup, as they are all sampled from Llama-3.1-8B-Instruct.

6.3 Analysis of Sampling Strategies in MC-PO

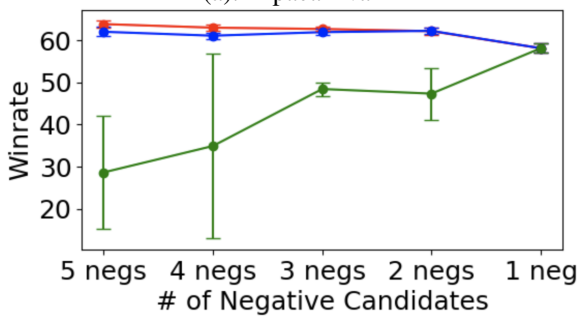
We also study how varying the sampling strategy in MC-PO impacts the performance, as it changes the quality of the preference dataset used by the algorithm. We develop Max and Min samplings as variants of MC-PO based on the kernel defined in Algorithm 1. Here, the Max (Min) sampling variant outputs the candidate with the maximum (minimum) weight, where the weight for each candidate i is calculated as $w_i = \frac{\exp\left(r_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{y}_i)\right)}{\sum_{j=0}^{M}\exp\left(r_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{y}_j)\right)}$. Moreover, we construct preference datasets with varying candidate qualities. For instance, based on the Nectar dataset, we progressively remove highly ranked responses for each input prompt. The first dataset excludes the rank-2 response, while the second one excludes both the rank-2 and rank-3 responses, resulting in lower candidate quality. The results are summarized in Figure 2, from which we observe the following insights:

(1) Sampling from the MC kernel yields the best performance. MC-PO achieves a balance between exploitation (sampling according to the categorical distribution) and exploration (retaining probabilities for choosing alternative candidates). This approach accurately estimates the gradient of the log-normalization constant, which in turn leads to improved performance.

(2) Min-based variant leads to low performance and high variance. In the NLL estimation view of PO, the rejected samples are to estimate the gradient of the log-normalization constant. While CD suggests that hard negatives yield more accurate gradient estimates, it considers the Min sampling variant as the worst sampling strategy. According to CD, Min sampling leads to inaccurate gradient estimates, resulting in lower model performance and increased variance.



(a): Alpaca-Eval



(b): Arena-Hard

Figure 2: Win-rate evaluation of the optimized Llama-3.1-8B-SFT model using MC-PO, versus its Max and Min variants. We use five modified Nectar datasets for training. By x-negs, we mean that the training dataset contains x negative candidates for each input prompt. For example, the 3-negs training dataset is constructed by removing rank-2 and rank-3 responses from Nectar.

(3) MC-PO's performance is correlated with the quality of candidates. When the preference dataset includes five high-quality candidates for each input prompt (referred to as 5-negs), both MC-PO and Max sampling strategies yield the best performance. However, as high-quality responses are eliminated from the candidate set, the performances of the models optimized with MC-PO and its Max variant decline due to the reduced quality of candidates. In the extreme case, when there is only one candidate per prompt (referred to as 1-neg), all three sampling strategies become equivalent.

6.4 Data and Ablation Analysis of MC-PO

MC-PO is robust against noisy samples. We demonstrate that MC-PO maintains high model performance even when noisy samples are included in the candidate set. Note that it has been shown that when the edit distance between pairs of responses is low, running DPO leads to a reduction in the model's likelihood for chosen responses (Pal et al. 2024).

We consider the processed Nectar dataset and inject noise

Model	Mistral-7B-SFT		Llama-3.1-8B-SFT		Llama-3.1-8B-Instruct	
Train dataset	Nectar		Nectar		Ultrafeedback (prompt only)	
Evaluation	Alpaca	Arena	Alpaca	Arena	Alpaca	Arena
DPO	25.07(±6.81)	42.01(±11.88)	33.74(±2.51)	60.25(±2.12)	64.22(±1.01)	75.88(±0.79)
RPO	15.31(±0.62)	39.18(±0.49)	32.50(±0.75)	59.20(±0.82)	51.27(±0.50)	64.74(±0.12)
EXO	21.77(±4.09)	30.63(±3.55)	26.48(±3.31)	52.89(±5.03)	64.75(±1.72)	74.93(±0.81)
SimPO	18.62(±2.64)	48.26(±3.90)	33.71(±1.41)	60.69(±1.01)	54.28(±1.48)	73.36(±1.38)
CPO	24.27(±0.39)	49.66(±0.34)	29.10(±1.01)	55.25(±0.60)	65.28(±0.54)	**77.92(±1.78)**
BCO	23.04(±0.19)	46.68(±1.62)	24.96(±1.28)	58.16(±1.76)	61.17(±1.27)	73.45(±0.54)
KTO	22.98(±0.23)	45.77(±1.85)	24.50(±1.35)	53.40(±0.75)	60.35(±0.67)	71.19(±0.49)
APO	15.79(±0.78)	35.94(±0.26)	21.13(±0.40)	53.25(±0.82)	57.54(±0.97)	70.70(±0.25)
SPPO	12.68(±0.27)	30.87(±0.67)	20.26(±0.34)	53.52(±0.56)	56.39(±0.58)	71.73(±0.62)
NCA	17.30(±0.37)	39.88(±0.80)	20.46(±0.36)	53.36(±1.25)	58.04(±0.42)	72.40(±0.23)
MC-PO	**30.86(±0.91)**	**52.75(±2.00)**	**35.84(±0.31)**	**63.77(±0.81)**	**66.90(±0.74)**	76.71(±0.24)

Table 1: Performance evaluation of the aligned models. Results are reported as win-rate against GPT-4. Each experiment is conducted using three random seeds. We report the mean and standard deviation of win-rate for AlpacaEval 2 (Dubois et al. 2024) and Arena-Hard (Li et al. 2024). The Mistral-7B-SFT and Llama-3.1-8B-SFT models are trained using the Nectar dataset (in the Base setup). Llama-3.1-8B-Instruct is trained using prompts from UltraFeedback and self-generated responses (in the Instruct setup). The highest score for each task is boldfaced.

into the candidate set by randomly switching two tokens of the chosen response for each input prompt. As shown in Table 2, training with DPO($-$) that selects the noisy sample as the rejected response leads to a degenerated model. DPO, which randomly selects a dispreferred response, is also impacted by the noise injection, but much less. MC-PO that samples a rejected response based on the log-probabilities of all candidates chooses semantically hard negatives instead of syntactically similar negatives with small edit distances, and achieves the best performance.

MC-PO benefits from sampling more negatives. In Table 3, we examine the performance of MC-PO and random sampling when the number of rejected samples is greater than 1. It is evident that random sampling does not achieve notable improvements when having more rejected samples. Conversely, the performance of MC-PO, which utilizes an MC kernel for sampling, is consistently improved as we increase the number of rejected samples.

MC-PO vs. augmented training dataset. As described earlier, MC-PO uses a preference dataset in which each prompt is paired with a chosen response and multiple rejected responses. An alternative way of using such data is to augment the DPO dataset by pairing the chosen response with each rejected response. E.g., in the processed Nectar dataset, where each prompt contains 5 candidate responses, this process increases the size of the training set by 4-fold. We trained Llama-3.1-8B-SFT with DPO and this augmented dataset, and achieved win-rates of $34.18(\pm1.26)$ and $59.62(\pm1.04)$ on Alpaca and Arena. Note that training the same model with MC-PO and its original dataset, we obtained win-rates of $35.84(\pm0.31)$ and $63.77(\pm0.81)$ on Alpaca and Arena, as reported in Table 1. Comparing these results, we notice that despite 4X increase in training time, the performance of DPO with the augmented training set is still inferior to that of MC-PO.

Model	Llama-3.1-8B	
Evaluation	Alpaca	Arena
DPO($-$)	1.08(±0.6)	3.17(±0.9)
DPO	23.62(±2.81)	50.51(±5.59)
MC-PO	**28.98(±1.34)**	**58.09(±2.63)**

Table 2: Evaluation of models trained with DPO and MC-PO when noise samples are included in the rejected responses set.

Nectar / Llama-3.1-8B-SFT			
Alpaca	$M=1$	$M=2$	$M=3$
Random	33.74(2.51)	33.73(0.49)	34.36(0.56)
MC-PO	**35.84(0.31)**	**36.73(0.59)**	**37.40(0.13)**
Arena	$M=1$	$M=2$	$M=3$
Random	60.25(2.12)	61.53(0.29)	61.16(0.69)
MC-PO	**63.77(0.81)**	**64.53(0.60)**	**66.16(0.13)**

Table 3: Comparison of random sampling and MC-PO with multiple rejected responses.

7 Conclusion, Limitations and Future Work

In this paper, we frame the alignment problem as a NLL estimation task and introduces sampling-based methods to solve PO. By establishing a connection between PO and NLL estimation, we provide theoretical justification for sampling rejected responses in PO and open the door to the development of more sophisticated sampling algorithms for improved performance. In practical scenarios, input prompts are often associated with multiple responses, for instance, rule-based reward functions can generate several chosen and rejected answers. The proposed MC-PO approach offers a principled way to sample from rejected responses, rather than indiscriminately using all available responses. As future research, we aim to showcase the benefits of utilizing a multi-step MCMC based PO solutions and will develop more efficient training algorithms to speed up these algorithms.

References

Azar, M. G.; Guo, Z. D.; Piot, B.; Munos, R.; Rowland, M.; Valko, M.; and Calandriello, D. 2024. A general theoretical paradigm to understand learning from human preferences. In *International Conference on Artificial Intelligence and Statistics*, 4447–4455. PMLR.

Chen, H.; He, G.; Yuan, L.; Cui, G.; Su, H.; and Zhu, J. 2024a. Noise Contrastive Alignment of Language Models with Explicit Rewards. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Chen, Z.; Deng, Y.; Yuan, H.; Ji, K.; and Gu, Q. 2024b. Self-Play Fine-Tuning Converts Weak Language Models to Strong Language Models. In *International Conference on Machine Learning*, 6621–6642. PMLR.

Chen, Z.; Liu, F.; Zhu, J.; Du, W.; and Qi, Y. 2024c. Towards Improved Preference Optimization Pipeline: from Data Generation to Budget-Controlled Regularization. *arXiv preprint arXiv:2411.05875*.

Christiano, P.; Leike, J.; Brown, T.; Martic, M.; Legg, S.; and Amodei, D. 2017. Deep reinforcement learning from human preferences. In *Proceedings of Neural Information Processing Systems*.

Cui, G.; Yuan, L.; Ding, N.; Yao, G.; Zhu, W.; Ni, Y.; Xie, G.; Liu, Z.; and Sun, M. 2024. UltraFeedback: Boosting Language Models with High-quality Feedback.

Dong, H.; Xiong, W.; Pang, B.; Wang, H.; Zhao, H.; Zhou, Y.; Jiang, N.; Sahoo, D.; Xiong, C.; and Zhang, T. 2024. Rlhf workflow: From reward modeling to online rlhf. *arXiv preprint arXiv:2405.07863*.

D'Oosterlinck, K.; Xu, W.; Develder, C.; Demeester, T.; Singh, A.; Potts, C.; Kiela, D.; and Mehri, S. 2024. Anchored Preference Optimization and Contrastive Revisions: Addressing Underspecification in Alignment. *CoRR*.

Dubey, A.; Jauhri, A.; Pandey, A.; Kadian, A.; Al-Dahle, A.; Letman, A.; Mathur, A.; Schelten, A.; Yang, A.; Fan, A.; et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

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

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

Go, D.; Korbak, T.; Kruszewski, G.; Rozen, J.; Ryu, N.; and Dymetman, M. 2023. Aligning language models with preferences through f-divergence minimization. In *Proceedings of the 40th International Conference on Machine Learning*, 11546–11583.

Guo, S.; Zhang, B.; Liu, T.; Liu, T.; Khalman, M.; Llinares, F.; Rame, A.; Mesnard, T.; Zhao, Y.; Piot, B.; et al. 2024. Direct language model alignment from online ai feedback. *arXiv preprint arXiv:2402.04792*.

Hinton, G. E. 2002. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8): 1771–1800.

Ji, H.; Lu, C.; Niu, Y.; Ke, P.; Wang, H.; Zhu, J.; Tang, J.; and Huang, M. 2024. Towards efficient and exact optimization of language model alignment. *arXiv preprint arXiv:2402.00856*.

Jung, S.; Han, G.; Nam, D. W.; and On, K.-W. 2024. Binary classifier optimization for large language model alignment. *arXiv preprint arXiv:2404.04656*.

Kwon, W.; Li, Z.; Zhuang, S.; Sheng, Y.; Zheng, L.; Yu, C. H.; Gonzalez, J. E.; Zhang, H.; and Stoica, I. 2023. Efficient Memory Management for Large Language Model Serving with PagedAttention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

Lambert, N.; Morrison, J.; Pyatkin, V.; Huang, S.; Ivison, H.; Brahman, F.; Miranda, L. J. V.; Liu, A.; Dziri, N.; Lyu, S.; et al. 2024. T\" ULU 3: Pushing Frontiers in Open Language Model Post-Training. *arXiv preprint arXiv:2411.15124*.

Li, T.; Chiang, W.-L.; Frick, E.; Dunlap, L.; Zhu, B.; Gonzalez, J. E.; and Stoica, I. 2024. From live data to high-quality benchmarks: The arena-hard pipeline.

Liu, T.; Zhao, Y.; Joshi, R.; Khalman, M.; Saleh, M.; Liu, P. J.; and Liu, J. 2024a. Statistical Rejection Sampling Improves Preference Optimization. In *The Twelfth International Conference on Learning Representations*.

Liu, Z.; Lu, M.; Zhang, S.; Liu, B.; Guo, H.; Yang, Y.; Blanchet, J.; and Wang, Z. 2024b. Provably mitigating overoptimization in rlhf: Your sft loss is implicitly an adversarial regularizer. *arXiv preprint arXiv:2405.16436*.

Meng, Y.; Xia, M.; and Chen, D. 2024. Simpo: Simple preference optimization with a reference-free reward. *arXiv preprint arXiv:2405.14734*.

Miller, E. 2024. Adding Error Bars to Evals: A Statistical Approach to Language Model Evaluations. *arXiv preprint arXiv:2411.00640*.

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*.

Pattnaik, P.; Maheshwary, R.; Ogueji, K.; Yadav, V.; and Madhusudhan, S. T. 2024. Enhancing alignment using curriculum learning & ranked preferences. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, 12891–12907.

Qi, B.; Li, P.; Li, F.; Gao, J.; Zhang, K.; and Zhou, B. 2024. Online DPO: Online Direct Preference Optimization with Fast-Slow Chasing. *arXiv preprint arXiv:2406.05534*.

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

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Shi, R.; Zhou, R.; and Du, S. S. 2024. The crucial role of samplers in online direct preference optimization. *arXiv preprint arXiv:2409.19605*.

Skalse, J. M. V.; Howe, N. H.; Krasheninnikov, D.; and Krueger, D. 2022. Defining and Characterizing Reward Gaming. In *Advances in Neural Information Processing Systems*.

Stiennon, N.; Ouyang, L.; Wu, J.; Ziegler, D.; Lowe, R.; Voss, C.; Radford, A.; Amodei, D.; and Christiano, P. 2020. Learning to summarize with human feedback. In *Proceedings of Neural Information Processing Systems*.

Tang, Y.; Guo, D. Z.; Zheng, Z.; Calandriello, D.; Cao, Y.; Tarassov, E.; Munos, R.; Pires, B. Á.; Valko, M.; Cheng, Y.; et al. 2024. Understanding the performance gap between online and offline alignment algorithms. *arXiv preprint arXiv:2405.08448*.

Tunstall, L.; Beeching, E.; Lambert, N.; Rajani, N.; Rasul, K.; Belkada, Y.; Huang, S.; von Werra, L.; Fourrier, C.; Habib, N.; et al. 2023. Zephyr: Direct distillation of lm alignment. *arXiv preprint arXiv:2310.16944*.

Wu, Y.; Sun, Z.; Yuan, H.; Ji, K.; Yang, Y.; and Gu, Q. 2024. Self-play preference optimization for language model alignment. *arXiv preprint arXiv:2405.00675*.

Xie, T.; Foster, D. J.; Krishnamurthy, A.; Rosset, C.; Awadallah, A.; and Rakhlin, A. 2024. Exploratory Preference Optimization: Harnessing Implicit Q*-Approximation for Sample-Efficient RLHF. *CoRR*.

Xiong, W.; Dong, H.; Ye, C.; Wang, Z.; Zhong, H.; Ji, H.; Jiang, N.; and Zhang, T. 2024. Iterative preference learning from human feedback: bridging theory and practice for RLHF under KL-constraint. In *Proceedings of the 41st International Conference on Machine Learning*, 54715–54754.

Xu, H.; Sharaf, A.; Chen, Y.; Tan, W.; Shen, L.; Van Durme, B.; Murray, K.; and Kim, Y. J. 2024. Contrastive preference optimization: Pushing the boundaries of llm performance in machine translation. *arXiv preprint arXiv:2401.08417*.

Xu, J.; Lee, A.; Sukhbaatar, S.; and Weston, J. 2023. Some things are more CRINGE than others: Iterative Preference Optimization with the Pairwise Cringe Loss. *arXiv preprint arXiv:2312.16682*.

Zhu, B.; Frick, E.; Wu, T.; Zhu, H.; and Jiao, J. 2023. Starling-7b: Improving llm helpfulness & harmlessness with rlaif.

Ziegler, D. M.; Stiennon, N.; Wu, J.; Brown, T. B.; Radford, A.; Amodei, D.; Christiano, P.; and Irving, G. 2019. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*.