Direct Preference Optimization: A Theoretical and Experimental Analysis

Cristian Alejandro Chávez Becerra
Giovanni Benedetti da Rosa

Master 2 - Data Science
Institut Polytechnique de Paris - École Polytechnique
Reinforcement Learning Course

Febraruary 2025

Contents

Direct Preference Optimization: A Theoretical and Experimental Analysis

Cristian Alejandro Chávez Becerra
Giovanni Benedetti da Rosa

Master 2 - Data Science
Institut Polytechnique de Paris - École Polytechnique
Reinforcement Learning Course

Abstract. In the last few years Large-scale language models (LMs) have been able to show their ability learn broad world knowledge and some reasoning skills, gaining world attention thanks to its capabilities and potential application. However, constraining or modifying their behavior remains challenging due to the inherently unsupervised nature of their training. One approach to addressing this issue is Reinforcement Learning from Human Feedback (RLHF), which leverages human preferences to train a reward model that guides the fine-tuning of the language model. The goal is to align the model's responses with human expectations while preventing excessive deviation from the original distribution, avoiding issues such as mode collapse. However, the RLHF process is often unstable, as training a separate reward model introduces additional complexity and can lead to unintended behaviors.
In this report it will be analysed an approach called Direct Preference Optimization that introduces a new parameterization of the reward model in Reinforcement Learning from Human Feedback (RLHF). This reformulation allows for the extraction of the optimal policy in a closed form, effectively solving the RLHF problem using only a classification loss. In the following sections it will be analyzed the step by step of the most relevant equations of the paper, including the Bradley-Terry model and the reparameterization of the reward function. Additionally, we review the experiments conducted by the original authors and present the results obtained from our own implementation, discussing their implications and performance.

1 Introduction

Large-scale language models (LMs), such as GPT, LLaMA, and PaLM, have demonstrated remarkable capabilities in understanding and generating human-like text[1]. These models are initially trained using unsupervised learning on vast text corpora, allowing them to acquire broad world knowledge and reasoning abilities. However, aligning these models with human intent and ethical considerations requires additional fine-tuning. One of the methods to allign these models is using Reinforcement Learning (RL).

Reinforcement Learning is one of the three fundamental paradigms of machine learning, alongside supervised learning and unsupervised learning. The core idea behind RL is that an agent (which could be a neural network, for instance) interacts with an environment by taking actions in response to its observations of the environment's current state. In some cases, the environment provides feedback to the agent by returning a reward, which indicates how appropriate the chosen action was. The agent's objective is to maximize the cumulative reward over time. However, defining an explicit reward function is not always feasible. This challenge arises for several reasons—one of which is that the problem's complexity may make it difficult to model the reward function explicitly. Despite this limitation, humans can often compare and rank different actions for a given state, even without explicitly defining a numerical reward function.

Different approaches leverage the advantage of utilizing human feedback (preferences) to train reward functions, which are then used to train the agent. One such approach is Reinforcement Learning from Human Feedback (RLHF)[2], which involves training both a reward model (which can also be a neural network) and the policy that the agent follows. While RLHF has been effective

in aligning reinforcement learning agents with human preferences, it introduces several challenges. Training a separate reward model adds significant complexity and increases computational costs, making the overall process more resource-intensive. Additionally, the optimization process can be unstable, as errors in the reward model may lead to reward hacking, where the agent learns to exploit the reward function in unintended ways, resulting in misaligned behavior. Furthermore, RLHF relies on reinforcement learning algorithms like Proximal Policy Optimization (PPO)[3], which require substantial computational power and careful hyperparameter tuning to ensure stability.

To address these challenges, Direct Preference Optimization(DPO) [4], proposes a more efficient and stable approach to fine-tuning LMs using human preferences. Unlike RLHF, which trains both a reward model and a policy, DPO directly optimizes the policy without needing an explicit reward function. This can be done by reformulating the preference learning problem: instead of using reinforcement learning to train a model to maximize a learned reward, DPO expresses the preference loss as a function of the policy itself. Given a dataset of human preferences over model responses, DPO can therefore optimize a policy using a simple binary cross entropy objective, producing the optimal policy to an implicit reward function fit to the preference data.

2 Related Works

In this section, we will dive into the basic ideas of works related with DPO. The first work to be explained will be Reinforcement Learning from Human Feedback. In the section 3, the study of the pipeline of this work will be detailed, since the reparametrization techniques that ends up in the loss function of DPO comes from this work's equations

2.1 Deep Reinforcement Learning from Human Preferences

Sometimes, reward functions for reinforcement learning models are hard to define for complex tasks. However, given two possible solutions for a task we may still have an idea of which of the two is the best approach, without explictly defining a reward function. That is the essential idea of the paper of Christiano et al [2], using preferences (feedback) from humans to train reward functions and reinforcement learning models.

At a time t the agent (reinforcement learning model) receives an observation $o_t \in \mathcal{O}$ from the environment and then sends an action $a_t \in \mathcal{A}$ to the environment. Usually the environment would also supply a reward $r_t \in \mathcal{R}$ to maximize. Since sometimes this reward is hard to properly define, in this case preferences between trajectory segments will be provided. A trajectory segment is a sequence of observations and actions, $\tau = ((o_0, a_0), (o_1, a_1), \ldots, (o_{k-1}, a_{k-1})) \in \mathcal{O} \times \mathcal{A}^k$. $\sigma^1 \succ \sigma^2$ indicates preference of the trajectory segment σ^1 over trajectory segment σ^2. Informally, the goal of the agent is to produce trajectories which are preferred by the human.

The algorithm's behaviour can be evaluated in two ways:

Quantitative: Assuming that preferences are generated by a reward function $r : \mathcal{O} \times \mathcal{A} \to \mathbb{R}$ if

$$r(o_1^0, a_1^0) + \cdots + r(o_1^{k-1}, a_1^{k-1}) > r(o_2^0, a_2^0) + \cdots + r(o_2^{k-1}, a_2^{k-1}),$$

whenever $\sigma^1 \succ \sigma^2$. In this case, the agent ought to receive a high total reward according to r. In the case of knowing the reward function it is possible to evaluate the agent quantitatively. Ideally, the agent will achieve reward nearly as high as if it had been using RL to optimize r.

Qualitative: In the case where there is no reward function, it may be still possible to qualitatively evaluate how well the agent satisfies the human's preferences for a given task.

Method basic steps At each point in time, there is a policy $\pi : \mathcal{O} \to \mathcal{A}$ and a reward function estimate $\hat{r} : \mathcal{O} \times \mathcal{A} \to \mathbb{R}$, each parametrized by deep neural networks. These networks are updated by three processes:

1. The policy π interacts with the environment to produce a set of trajectories $\{\tau_1, \ldots, \tau_i\}$. The parameters of π are updated by a traditional reinforcement learning algorithm, in order to maximize the sum of the predicted rewards $r_t = \hat{r}(o_t, a_t)$.
2. A pairs of segments (τ_1, τ_2) is selected from the trajectories $\{\tau_1, \ldots, \tau_i\}$ produced in step 1, and sent to a human for comparison.
3. The parameters of the mapping $\hat{r}$ are optimized via supervised learning to fit the comparisons collected from the human so far.

Preference Elicitation The human judgments are recorded in a database D of triples (τ_1, τ_2, μ), where τ_1 and τ_2 are the two segments and μ is a distribution over $\{1, 2\}$ indicating which segment the user preferred. If the human selects one segment as preferable, then μ puts all of its mass on that choice. If the human marks the segments as equally preferable, then μ is uniform. Finally, if the human marks the segments as incomparable, then the comparison is not included in the database.

Fitting the Reward Function A reward function estimate $\hat{r}$ can be seen as a preference-predictor if $\hat{r}$ is considered as a latent factor explaining the human's judgments and assume that the human's probability of preferring a segment i depends exponentially on the value of the latent reward summed over the length of the trajectory:

$$\hat{P}(1 \succ 2) = \frac{\exp\left(\sum_t \hat{r}(o_1^t, a_1^t)\right)}{\exp\left(\sum_t \hat{r}(o_1^t, a_1^t)\right) + \exp\left(\sum_t \hat{r}(o_2^t, a_2^t)\right)}.$$

The goal is to choose the $\hat{r}$ that minimize the cross-entropy loss between these predictions and the actual human labels:

$$\text{loss}(\hat{r}) = \sum_{(1,2,\mu)\in D} \mu(1) \log \hat{P}(1 \succ 2) + \mu(2) \log \hat{P}(2 \succ 1).$$

This follows the Bradley-Terry model [5] for estimating score functions from pairwise preferences, and is the specialization of the Luce-Shephard choice rule (Luce [6]; Shepard [7]) to preferences over trajectory segments.

Subsequently, the parameters of the policy π are updated by a traditional reinforcement learning algorithm, in order to maximize the sum of the predicted rewards from $\hat{r}$

2.2 PPO

The great majority of works that work with Reinforcement Learning techniques to align the output's with user preferences were done with Proximal Policies Algorithms(PPO)[3], as in the so called *instructGPT* [8] [9] or a similar method with TRPO(Trust Region Policy Optimization) [10] in Christiano et. al[2]. Since this method is used as a comparison baseline in the original paper, in this section, we are going to describe the main points of the algorithm, to later make a better comparison on how DPO changes the main approach.

The Proximal Policy Optimization (PPO) algorithm aims to approximate the optimal policy by alternating between two main steps: sampling data through interaction with the environment and optimizing a surrogate objective function using stochastic gradient ascent[3]. The Loss that we are trying to minimize is a mixture between three terms: Clip Surogate Objective, a squared-error loss and an Entropy Bonus (Exploration Term) term.

Clipped Surrogate Policy Loss (Policy Optimization Objective) The clipped version of the Surrogate Objective Function, designed to approximate the policy while preventing excessive updates, is given in Eq. 1.By taking the minimum of the clipped and unclipped objective, we ensure a conservative (pessimistic) estimate, allowing policy updates only when they do not excessively

improve the objective but always considering negative changes.The first term inside the minimum function is L^{CPI} [10]. The second term, $\text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)A_t$, clips the probability ratio to prevent r_t from moving outside the range $[1-\epsilon, 1+\epsilon]$. This removes the incentive for large policy updates.

$$L^{\text{CLIP}}(\theta) = \mathbb{E}\left[\min\left(r_t(\theta)A_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)A_t\right)\right] \tag{1}$$

where s_t and a_t are the state and action in time t; $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ is the ratio of new to old policy probabilities; A_t is the advantage function; ϵ is the clipping parameter (e.g., 0.2); the min operator ensures that the policy does not change too drastically by clipping the ratio.

Value Function Loss (Critic Loss) The second term of the PPO loss is a squared error loss to train a value function $V(s_t)$ at each states_t:

$$L^V(\theta) = \mathbb{E}\left[\left(V_\theta(s_t) - V_t^{\text{target}}\right)^2\right] \tag{2}$$

Entropy Bonus (Exploration Term) Finally, to encourage exploration, really important part in RL algorithms[11], and prevent early policy collapse, an entropy regularization term is added.

$$L^H(\theta) = \mathbb{E}\left[-\beta H(\pi_\theta)\right] \tag{3}$$

where $H(\pi_\theta) = -\sum_a \pi_\theta(a|s)\log\pi_\theta(a|s)$ is the entropy of the policy; β is a hyperparameter that controls the strength of entropy regularization.

Final PPO Loss Function With these terms we can define thee total loss in PPO, which is a weighted sum of these terms:

$$L(\theta) = L^{\text{CLIP}}(\theta) - c_1 L^V(\theta) + c_2 L^H(\theta) \tag{4}$$

Generalized Advantage Estimation (GAE) The advantage function estimate how much better or worse taking a specific action a_t at state s_t is compared to the average action at that state, according to the current policy($A_t = Q(s_t, a_t) - V(s_t)$). In this context, for a fixed length-T trajectory segment, a discount factor(γ), and a hyperparameter λ we can define GAE [3]:

$$A_t = \sum_{l=0}^{T} (\gamma\lambda)^l \delta_{t+l}, \quad \text{where} \quad \delta_t = r_t + \gamma V(s_{t+1}) - V(s_t) \tag{5}$$

Using the previously defined equations, we can define the training process of the PPO: At each iteration N parallel actors collect T timesteps of data, resulting in a total of $N \times T$ samples. The surrogate loss is then computed on this dataset and optimized using mini-batch stochastic gradient descent (SGD)[3].

According to the authors PPO brought new advantages in the reinforcement learning process that can improve the process: while policy gradient methods used to perform one a single step update per batch(TRPO), PPO reuses data across multiple epochs and experimentally it significantly improved the performance, reducing the variance in policy updates[3].

Due to these factors, PPO became one of the most widely used methods. While it performs well in aligning models with human preferences, it requires extensive computational resources for RLHF, as it involves optimizing multiple parameters and complex reinforcement learning updates.

3 Preliminaries for Direct Preference Optimization

3.1 Placket General Model for Preferences Probabilities

Placket[12] defined a general formula to calculate the probability for a specific preference order between $1, 2, .., r$ different choices.

First of all, we must define a value β_j that will represent a quantitative value that represents how possible (or how profitable) is that we choose the option i_j between r different choices. The total sum of the resulting quantities β_j may exceed 1, but if each of them is scaled down by the same factor, then we obtain estimated probabilites p_{i_j} for choosing any option i_j.

In this case, we will assume that we know each of the probabilities p_{i_j} and the main question to solve is how to model the probability of an specific permutation (or preference) of the r different options i_j we have. A reasonable procedure is to define the probabilities of being chosen as conditional probabilities based on those which remain. Therefore, we suppose that we have $1, 2, ..., r$ different choices with associated probabilities $p_{i_1}, p_{i_2}, ..., p_{i_r}$ where the sum of all of these probabilities must be one, that is $\sum_{j=1}^{r} p_{i_j} = 1$. For instance, let us say that option 3 is chosen, then, the probabilities for the options $1, 2, ..., r$ to be chosen alone are, respectively:

$$\frac{p_{i_1}}{(1-p_{i_3})}, \frac{p_{i_2}}{(1-p_{i_3})}, ..., \frac{p_{i_r}}{(1-p_{i_3})}$$

Given this idea, it is natural to think about its extension to calculate the probability of any permutation. We can start by a simple case, the probability for a permutation ijk of only 3 elements would be:

$$P(i_i i_j i_k \mid i_1, i_2, \ldots, i_r) = p_i(\frac{p_j}{1-p_i})(\frac{p_k}{1-p_i-p_j}) \tag{6}$$

The basic idea is that, once an option is chosen, the probability space is "reduced" since this option is no longer available. In a more general way, we can define this formula for the probability of finding a specific permutation $\tau : [r] \to [r]$ for the r different choices:

$$P(\tau \mid i_1, i_2, \ldots, i_r) = \prod_{j=1}^{r} \frac{p_{i_j}}{1-\sum_{k=1}^{j-1} p_{i_k}} \tag{7}$$

where the denominator can be rewritten as:

$$1 - \sum_{k=1}^{j-1} p_{i_k} = \sum_{k=j}^{r} p_{i_k}$$

Giving as a result

$$P(\tau \mid i_1, i_2, \ldots, i_r) = \prod_{j=1}^{r} \frac{p_{i_j}}{\sum_{k=j}^{r} p_{i_k}} \tag{8}$$

since we had previously defined our values β_i for each x_i and we concluded that if we scale down the values by the same factor we would obtain the estimated probabilities p_{i_j} we will define:

$$p_{i_j} = \frac{e^{\beta_{i_j}}}{\sum_{j=1}^{r} e^{\beta_{i_i}}} \tag{9}$$

then by replacing this expression of 9 in equation 8 we obtain the general Placket General Model for Preferences Probabilites

$$\begin{aligned} P(\tau \mid i_1, i_2, \ldots, i_r) &= \prod_{j=1}^{r} \frac{\frac{e^{\beta_{i_j}}}{\sum_{a=1}^{r} e^{\beta_{i_a}}}}{\sum_{k=j}^{r} \frac{e^{\beta_{i_k}}}{\sum_{a=1}^{r} e^{\beta_{i_a}}}} \\ &= \prod_{j=1}^{r} \frac{\frac{e^{\beta_{i_j}}}{\sum_{a=1}^{r} e^{\beta_{i_a}}}}{\frac{\sum_{k=j}^{r} e^{\beta_{i_k}}}{\sum_{a=1}^{r} e^{\beta_{i_a}}}} \\ &= \prod_{j=1}^{r} \frac{e^{\beta_{i_j}}}{\sum_{k=j}^{r} e^{\beta_{i_k}}} \end{aligned} \tag{10}$$

In our context, of large language models, when presented with a prompt x and a set of K answers $y_1, \ldots, y_K$, a user would output a permutation $\tau : [K] \to [K]$, giving their ranking of the answers. We can now replace β_{y_j} by the reward $r^*(x, y_{\tau(j)})$. As a result, we would have that the probability of this output permutation (or preference) from the user is given by:

$$p^*(\tau \mid y_1, \ldots, y_K, x) = \prod_{t=1}^{K} \frac{e^{r^*(x, y_{\tau(t)})}}{\sum_{j=t}^{K} e^{r^*(x, y_{\tau(j)})}} \tag{11}$$

3.2 Bradley-Terry model

The Bradley-Terry [5] model comes from the Placket General Model for Preferences Proabilities for the specific case of $k = 2$ that is, when we only have two possible preferences.

By defining $K = 2$ in 11, we have only two possible rankings. Suppose the user ranks y_1 first and y_2 second, i.e., $\tau(1) = 1, \tau(2) = 2$. Then, the probability simplifies as follows:

$$p^*(\tau \mid y_1, y_2, x) = \left(\frac{\exp(r^*(x, y_1))}{\exp(r^*(x, y_1)) + \exp(r^*(x, y_2))}\right)\left(\frac{\exp(r^*(x, y_2))}{\exp(r^*(x, y_2))}\right) \tag{12}$$

Since for $k = 1$:

$$\frac{\exp(r^*(x, y_1))}{\exp(r^*(x, y_1)) + \exp(r^*(x, y_2))} \tag{13}$$

and for $k = 2$:

$$\frac{\exp(r^*(x, y_2))}{\exp(r^*(x, y_2))} \tag{14}$$

However, since the second fraction is always 1, the overall probability simplifies to:

$$p^*(\tau \mid y_1, y_2, x) = \frac{\exp(r^*(x, y_1))}{\exp(r^*(x, y_1)) + \exp(r^*(x, y_2))} \tag{15}$$

Using the notation of the original paper:

$$p^*(y_1 \succ y_2 \mid x) = \frac{\exp(r^*(x, y_1))}{\exp(r^*(x, y_1)) + \exp(r^*(x, y_2))} \tag{16}$$

This is the standard probability expression for choosing y_1 over y_2 called the Bradley-Terry [5] model that is used in different approaches for using human preferences in reinforcement learning models like RLHF.

3.3 RLHF Pipeline

Now, we will start by reviewing more in detail the Reinfocement Learning from Human Feedback pipeline used in Ziegler et al. [9] This pipeline is conformed by 3 main steps:

Initialize the model using a supervised fine-tuned (SFT) LM We begin by fine-tuning a pre-trained language model using supervised learning using a dataset specified for the downstream tasks of interest. We will define this initial model as π^{SFT}

Preference Sampling and Reward Model Learning Now, we will prompt the model π^{SFT} with some prompts x_i, the idea is to prompt the model with the same prompt x_i to get two different answers y_1 and y_2 defining the pair $(y_1, y_2) \sim \pi^{\text{SFT}}(y \mid x)$. Subsequently, human labelers will define preferences over one of the answers denoting as $y_w \succ y_l \mid x$, where y_w and y_l means the preferred and dispreferred answer respectively. These preferences, over the different prompts, are supposed to be generated using a reward model $r^*(y, x)$ that we do not know.

There are multiple approaches to modeling human preferences. In this work, we adopt the method proposed by Bradley-Terry [5]. As already cited, if more than two responses were available, a more general framework such as the Plackett-Luce model [12] could be employed. Notably, the Bradley-Terry model is a special case of the Plackett-Luce model for $k = 2$. The Bradley-Terry approach models the probability distribution of human preferences as follows:

$$p^*(y_1 \succ y_2 \mid x) = \frac{\exp\left(r^*(x, y_1)\right)}{\exp\left(r^*(x, y_1)\right) + \exp\left(r^*(x, y_2)\right)} \tag{17}$$

This equation can also be defined in terms of the sigmoid function. The procedure starts by expanding the denominator:

$$p^*(y_1 \succ y_2 \mid x) = \frac{\exp\left(r^*(x, y_1)\right)}{\exp\left(r^*(x, y_1)\right)\left(1 + \frac{\exp(r^*(x, y_2))}{\exp(r^*(x, y_1))}\right)} \tag{18}$$

Rewriting the fraction inside the parentheses:

$$p^*(y_1 \succ y_2 \mid x) = \frac{1}{1 + \frac{\exp(r^*(x, y_2))}{\exp(r^*(x, y_1))}} \tag{19}$$

Using exponent properties:

$$\frac{\exp\left(r^*(x, y_2)\right)}{\exp\left(r^*(x, y_1)\right)} = \exp\left(r^*(x, y_2) - r^*(x, y_1)\right) \tag{20}$$

Substituting back:

$$p^*(y_1 \succ y_2 \mid x) = \frac{1}{1 + \exp\left(r^*(x, y_2) - r^*(x, y_1)\right)} \tag{21}$$

Since the sigmoid function is defined as

$$\sigma(z) = \frac{1}{1 + \exp(-z)} \tag{22}$$

Then, the equation 21 can be expressed in terms of 22 as:

$$p^*(y_1 \succ y_2 \mid x) = \sigma\left(r^*(x, y_1) - r^*(x, y_2)\right) \tag{23}$$

Defining a dataset of comparisons $D = \{(x^{(i)}, y_w^{(i)}, y_l^{(i)})\}_{i=1}^N$ sampled from p^*, the reward model $r_\phi(x, y)$ can be parametrized via maximum likelihood (or via minimum log-likelihood).

Since the goal is that the policy model, using the preferences dataset, increases the probability of answering to prompt $x^{(i)}$ with answer $y_l^{(i)}$, then, the idea is to maximize the probability assigned to human preferences. And since we do not have access to $r^*(x, y)$, we approximate it with a parametric reward model $r_\phi(x, y)$.

To estimate r_ϕ, we maximize the likelihood of the observed human preference data:

$$\max_\phi \prod_{(x, y_w, y_l) \sim D} p_\phi(y_w \succ y_l | x) \tag{24}$$

Then, taking the negative log-likelihood (to convert it into a loss function that should be minimized):

$$L_R(r_\phi, D) = -\mathbb{E}_{(x, y_w, y_l) \sim D} \log p(y_w \succ y_l \mid x) \tag{25}$$

Substituting $p(y_w \succ y_l \mid x)$ as in equation 23:

$$L_R(r_\phi, D) = -\mathbb{E}_{(x, y_w, y_l) \sim D} \log \sigma(r_\phi(x, y_w) - r_\phi(x, y_l)) \tag{26}$$

where σ is the sigmoid function. In this case, of LMs, the reward model is in fact a network that is usually initialized as the SFT model $\pi^{\text{SFT}}(y \mid x)$ adding a linear layer on top of the final transformer layer that predicts a single scalar prediction, that is, the reward value [9].

To reduce the variance on our reward model, the rewards can be normalized.

$$\mathbb{E}_{x, y \sim D}[r_\phi(x, y)] = 0$$

for all x.

The notion in this loss function can be seen as:

- If $r_\phi(x, y_w) \gg r_\phi(x, y_l)$, then $\sigma(r_\phi(x, y_w) - r_\phi(x, y_l)) \approx 1$, so $\log \sigma$ is close to 0, meaning small loss since it has the desired behaviour.
- If $r_\phi(x, y_w) \approx r_\phi(x, y_l)$, then σ is around 0.5, and $\log \sigma$ is negative, meaning higher loss.
- If $r_\phi(x, y_w) \ll r_\phi(x, y_l)$, then the loss is very large, forcing the model to correct.

This encourages the reward model to rank the preferred response y_w higher than the dispreferred one y_l.

Language Model Fine Tuning using the Reward Model During the RL phase, the Language Model can be fine tuned using our network for the reward model. According to [13], the optimization problem can be defined as:

$$\max_{\pi_\theta} \mathbb{E}_{x \sim D, y \sim \pi_\theta(y|x)} \left[r_\phi(x, y) - \beta D_{\text{KL}}\big(\pi_\theta(y \mid x) \| \pi_{\text{ref}}(y \mid x)\big) \right] \tag{27}$$

β is a parameter used so that the fine tuned model does not diverges too much from the base reference policy π_{ref}. With this constraint the generation diversity is mantained too and collapse to high-reward answers only is avoided. In practice, the language model policy π_θ is also initialized to π^{SFT}.

We can see, that this loss function consists of two components. The first one refers to the goal maximizing the rewards given by the reward function, driving the model toward preferred behaviors. The second part, since is with a negative signed (and we try to maximize the whole equation) enforces minimization of the divergence between the optimized model and the reference model, thereby maintaining stability and alignment with the desired distribution.

Since nature of language generation is discrete, the objective is not differentiable, but can be optimized via reinforcement learning. The approach used by [13],[14], [8] to define the reward function is:

$$r(x, y) = r_\phi(x, y) - \beta\big(\log \pi_\theta(y \mid x) - \log \pi_{\text{ref}}(y \mid x)\big), \tag{28}$$

maximized by the use of PPO [3].

4 Direct Preference Optimization

In this section, the DPO objective will be derived based on the previously defined optimization problem (equation 27) and the Placket-Luce Model.

4.1 Optimum of the KL-Constrained Reward Objective Function

As stated in the original paper[4], based on Eq. 27, it is possible to obtain the optimal solution for this problem.

First, we use the definition of the Kullback-Leibler (KL) divergence $\mathbb{D}_{\text{KL}}\left[P\|Q\right] = \sum_i P(i) \log \frac{P(i)}{Q(i)}$:

Using this definition, we rewrite our optimization objective:

$$\max_\pi \mathbb{E}_{x\sim\mathcal{D}, y\sim\pi}\left[r(x, y)\right] - \beta\mathbb{D}_{\text{KL}}\left[\pi(y|x)\|\pi_{\text{ref}}(y|x)\right] = \max_\pi \mathbb{E}_{x\sim\mathcal{D}}\mathbb{E}_{y\sim\pi(y|x)}\left[r(x, y) - \beta \log \frac{\pi(y|x)}{\pi_{\text{ref}}(y|x)}\right] \tag{29}$$

So, by linearity of expectation, we divide the equation by $-\beta$ and changing to its dual form:

$$\min_\pi \mathbb{E}_{x\sim\mathcal{D}, y\sim\pi(y|x)}\left[\log \frac{\pi(y|x)}{\pi_{\text{ref}}(y|x)} - \frac{1}{\beta} r(x, y)\right]. \tag{30}$$

Using logarithm properties and considering the partition function of the Gibbs (Boltzmann) distribution, we can rewrite the equation by adding and subtracting $\log Z(x)$:

$$\min_\pi \mathbb{E}_{x\sim\mathcal{D}}\mathbb{E}_{y\sim\pi(y|x)}\left[\log \frac{\pi(y|x)}{\frac{1}{Z(x)}\pi_{\text{ref}}(y|x)\exp\left(\frac{1}{\beta}r(x, y)\right)} - \log Z(x)\right] \tag{31}$$

where the partition function $Z(x)$ is defined as:

$$Z(x) = \sum_y \pi_{\text{ref}}(y|x)\exp\left(\frac{1}{\beta}r(x, y)\right) \tag{32}$$

As described in the paper, let's define π^* as the optimal policy:

$$\pi^*(y|x) = \frac{1}{Z(x)}\pi_{\text{ref}}(y|x)\exp\left(\frac{1}{\beta}r(x,y)\right), \tag{33}$$

This is a valid probability distribution once $\pi^*(y|x) \geq 0, \forall y$, and $\sum_y \pi^*(y|x) = 1$. It's valuable to mention that 32, is a function only of x and the reference policy π_{ref}, but does not depend on π. Based on this, and on the definition of the KL divergent we can rewrite 31 as:

$$\min_\pi \mathbb{E}_{x\sim\mathcal{D}}\left[\mathbb{E}_{y\sim\pi(y|x)}\left[\log\frac{\pi(y|x)}{\pi^*(y|x)}\right] - \log Z(x)\right] \tag{34}$$

$$= \min_\pi \mathbb{E}_{x\sim\mathcal{D}}\left[\mathbb{D}_{\text{KL}}\left(\pi(y|x)\|\pi^*(y|x)\right) - \log Z(x)\right] \tag{35}$$

As $Z(x)$ is independent on π, we just need to worry about minimizing the KL term. Considering the definition of KL divergence, Gibb's inequality affirms that equality holds if and only if $P = Q$ almost everywhere, meaning $P(x) = Q(x)$ for all values of x in the support where $P(x) > 0$; otherwise, the KL divergence will be greater than zero. Then we can check that the optimal solution is exactly:

$$\pi_r(y|x) = \pi^*(y|x) = \frac{1}{Z(x)}\pi_{\text{ref}}(y|x)\exp\left(\frac{1}{\beta}r(x,y)\right), \tag{36}$$

With equation 36, even though we can use a Maximum Likelihood Estimator to estimate r_ϕ, it's really expensive to estimate the partition function $Z(x)$, leading to practical issues and difficult approaches[15]. Nevertheless, we can rewrite 36 to express the reward function in terms of its optimal policy π_r, the reference policy π_{ref}, and an unkown partition function $Z(x)$. To do this we start by taking the log of both sides in eq. 36, and rearranging the equation:

$$r(x,y) = \beta log\frac{\pi_r(y|x)}{\pi_{\text{ref}}(y|x)} + \beta logZ(x) \tag{37}$$

4.2 Deriving DPO Objective under the Bradley-Terry Model

As shown in the previous section, technically, the unavailable Ground truth reward $r^*(x,y)$ can be written in terms of its optimal policy(π^*) 37. Considering the definition of the Bradley-Terry preference model 16, we can rewrite the optimal reward function based on eq 37:

$$p^*(y_1 > y_2|x) = \frac{\exp\left(\beta\log\frac{\pi^*(y_1|x)}{\pi_{\text{ref}}(y_1|x)} + \beta\log Z(x)\right)}{\exp\left(\beta\log\frac{\pi^*(y_1|x)}{\pi_{\text{ref}}(y_1|x)} + \beta\log Z(x)\right) + \exp\left(\beta\log\frac{\pi^*(y_2|x)}{\pi_{\text{ref}}(y_2|x)} + \beta\log Z(x)\right)}$$

Next, we multiply the numerator and the denominator by $exp\left(-\beta logZ(x) - \beta\log\frac{\pi^*(y_1|x)}{\pi_{\text{ref}}(y_1|x)}\right)$, and is easy to see the logistic function σ:

$$p^*(y_1 > y_2|x) = \frac{1}{1 + \exp\left(\beta\log\frac{\pi^*(y_2|x)}{\pi_{\text{ref}}(y_2|x)} - \beta\log\frac{\pi^*(y_1|x)}{\pi_{\text{ref}}(y_1|x)}\right)}$$

$$= \sigma\left(\beta\log\frac{\pi^*(y_1|x)}{\pi_{\text{ref}}(y_1|x)} - \beta\log\frac{\pi^*(y_2|x)}{\pi_{\text{ref}}(y_2|x)}\right).$$

Now that the probability of human preference data in terms of the optimal policy, we can formulate a maximum likelihood objective for a parametrized policy π. Based on the log-likelihood loss defined in the previous section 26, we can write:

$$\mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}}\left[\log\sigma\left(\beta\log\frac{\pi_\theta(y_w \mid x)}{\pi_{\text{ref}}(y_w \mid x)} - \beta\log\frac{\pi_\theta(y_l \mid x)}{\pi_{\text{ref}}(y_l \mid x)}\right)\right] \tag{38}$$

4.3 Deriving DPO Objective under the Plackett -Luce Model

As already defined in the preliminaries, the Plackett-Luce model is a generalization of the Bradley-Terry model that extends its application from pairwise comparisons to full rankings, allowing for the probabilistic modeling of ordered preferences among multiple items. For a given prompt x, let $\mathcal{Y}_x = \{y_1, y_2, \ldots, y_K\}$ be the set of candidate responses. The model assigns a probability to a ranking τ over $\mathcal{Y}_x$ according to following:

$$p^*(\tau|y_1, \ldots, y_K, x) = \prod_{k=1}^{n} \frac{r^*(y_{\tau(k)}|x)}{\sum_{j=t}^{n} r^*(y_{\tau_j}|x)} \tag{39}$$

Similar to what happens with the Bradley-Taylor model the partition function $Z(x)$ cancells-out, having as a result:

$$p^*(\tau|y_1, \ldots, y_K, x) = \prod_{k=1}^{K} \frac{\exp\left(\beta\log\frac{\pi^*(y_{\tau(k)}|x)}{\pi_{\text{ref}}(y_{\tau(k)}|x)}\right)}{\sum_{j=k}^{K}\exp\left(\beta\log\frac{\pi^*(y_{\tau(j)}|x)}{\pi_{\text{ref}}(y_{\tau(j)}|x)}\right)}$$

Again, similar to the previous section considering a a Dataset$D = \{\tau^{(i)}, y_1^{(i)}, \ldots, y_K^{(i)}, x^{(i)}\}_{i=1}^N$ of prompts and user-specified rankings, we can use a parameterized model and optimize this objective with maximum-likelihood method:

$$\mathcal{L}_{\text{DPO}}(\pi_\theta, \pi_{\text{ref}}) = -\mathbb{E}_{\tau,y_1,\ldots,y_K,x\sim\mathcal{D}}\left[\log\prod_{k=1}^{K}\frac{\exp\left(\beta\log\frac{\pi_\theta(y_{\tau(k)}|x)}{\pi_{\text{ref}}(y_{\tau(k)}|x)}\right)}{\sum_{j=k}^{K}\exp\left(\beta\log\frac{\pi_\theta(y_{\tau(j)}|x)}{\pi_{\text{ref}}(y_{\tau(j)}|x)}\right)}\right] \tag{40}$$

4.4 DPO Update and Outline

To show the usability of the DPO derivations we can analyze the gradients of the previously defined losses $\mathcal{L}_{\text{DPO}}$, with respect to the parameters θ:

$$\begin{aligned}\nabla_\theta\mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = -\beta\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}}\Bigg[&\underbrace{\sigma\big(\hat{r}_\theta(x, y_l) - \hat{r}_\theta(x, y_w)\big)}_{\text{higher weight when reward estimate is wrong}} \\ &\times\Bigg(\underbrace{\nabla_\theta\log\pi(y_w|x)}_{\text{increase likelihood of } y_w} - \underbrace{\nabla_\theta\log\pi(y_l|x)}_{\text{decrease likelihood of } y_l}\Bigg)\Bigg]\end{aligned} \tag{41}$$

where $\hat{r}_\theta(x, y) = \beta\log\frac{\pi_\theta(y|x)}{\pi_{\text{ref}}(y|x)}$ is the reward implicitly defined by the language model π_θ and reference model π_{ref}.

Conceptually, the gradient of the DPO loss function $\mathcal{L}_{\text{DPO}}$ steers the model's learning process by increasing the probability of preferred completions y_w while suppressing the probability of dispreferred completions y_l. The extent of this adjustment is influenced by the implicit reward model $\hat{r}_\theta$, which assesses how strongly the model ranks y_l compared to y_w. This weighting is scaled by β, amplifying the effect when the implicit reward model misorders the completions. Furthermore, the KL constraint regulates this modification, maintaining a trade-off between adapting the model

and preserving alignment with the reference distribution. The effect of β in the process will be further discussed in details in this document.

The pipeline of the DPO algorithm is structured as follows:

1) First, we label completions with human preferences to construct the offline dataset of preferences $D = \left\{ \left(x^{(i)}, y_w^{(i)}, y_l^{(i)} \right) \right\}_{i=1}^{N}$ where the completions $y_1, y_2 \sim \pi_{\text{ref}}(\cdot|x)$ are sampled from the reference model for each prompt x.
2) Optimize the LLM π_θ to minimize $\mathcal{L}_{\text{DPO}}$ for the given reference model π_{ref}, dataset D, and desired scaling factor β.

In practice, public datasets available are used, rather than generating samples and gathering human preferences. When the π^{SFT} is available we initialize $\pi_{\text{ref}} = \pi^{SFT}$. When it's not available, We initialize π_{ref} by maximizing the likelihood of preferred completions (x, y_w), that is, $\pi_{\text{ref}} = \arg\max_\pi \mathbb{E}_{(x,y_w)\sim\mathcal{D}} [\log \pi(y_w \mid x)]$ to help mitigate the distribution shift between the true reference distribution and the one we are trying to optimize.

5 Theoretical Analysis

5.1 Your language model is a secretly reward model

As we could see in the previous sections, minimizing KL divergence with a base policy, allows to perform the RL process learning the policy using a single maximum likelihood objective. So far, we've discussed reparametrizations in the Bradley-Terry model without providing further details. As cited in the original paper, we can see that equation 37 is equivalent to a Bradley-Terry model with a reward parameterization $r^*(x, y) = \beta \log\left(\frac{\pi_\theta^*(y|x)}{\pi_{\text{ref}}(y|x)} \right)$, and we optimize our parametric model π_θ in 37 a way that is equivalent to the reward model optimization in Eq. 26, with some change of variables. Next, we are going to show that this reparameterization imposes no restrictions on the class of learned reward models and permits the exact recovery of the optimal policy. We start by a definition:

Definition 1 (Equivalent Reward Functions) *We say that two reward functions $r(x, y)$ and $r'(x, y)$ are* equivalent *if and only if*

$$r(x, y) - r'(x, y) = f(x)$$

for some function f.

Now, we are going to state two lemmas followed by their proofs based on the definition 1.

Lemma 1. Under the Plackett-Luce, and in particular the Bradley-Terry, preference framework, two reward functions from the same class induce the same preference distribution.

To prove Lemma 1, we start by considering that two rewarding functions are from the same equivalence class $r(x, y)$ and $r'(x, y)$, as defined in 1. We recall that Bradley-Taylor model is a special case, when $k = 2$ of the Plackett -Luce model, so our proof is gonna start by the most general case. For any prompt x, answers $y_1, ..., y_K$, ranking τ, and the probability distribution over rankings induced by a particular reward function $r(x, y)$ as p_r:

$$p_{r'}(\tau \mid y_1, \ldots, y_K, x) = \prod_{k=1}^{K} \frac{\exp(r'(x, y_{\tau(k)}))}{\sum_{j=k}^{K} \exp(r'(x, y_{\tau(j)}))}$$

Using the definition 1:

$$p_{r'}(\tau \mid y_1, \ldots, y_K, x) = \prod_{k=1}^{K} \frac{\exp(r(x, y_{\tau(k)}) + f(x))}{\sum_{j=k}^{K} \exp(r(x, y_{\tau(j)}) + f(x))}$$
$$= \prod_{k=1}^{K} \frac{\exp(f(x)) \exp(r(x, y_{\tau(k)}))}{\exp(f(x)) \sum_{j=k}^{K} \exp(r(x, y_{\tau(j)}))} = p_r(\tau \mid y_1, \ldots, y_K, x)$$

Lemma 2. Two reward functions from the same equivalence class induce the same optimal policy under the constrained RL problem.

Again using the definition 1, we are going to consider two reward functions from the same class $r(x, y) = r(x, y) + f(x)$, and their respective optimal policies π_r and π'_r the corresponding optimal policies. Considering the solution pf the KL-Constrained Reward Objective Function, defined in eq. 36:

$$\pi_{r'}(y \mid x) = \frac{\pi_{\text{ref}}(y \mid x) \exp\left(\frac{1}{\beta} r'(x, y)\right)}{\sum_y \pi_{\text{ref}}(y \mid x) \exp\left(\frac{1}{\beta} r'(x, y)\right)}$$

We do a very similar process ad in Lemma 1. Using the definition 1:

$$\pi_{r'}(y \mid x) = \frac{\pi_{\text{ref}}(y \mid x) \exp\left(\frac{1}{\beta} r(x, y) + f(x)\right)}{\sum_y \pi_{\text{ref}}(y \mid x) \exp\left(\frac{1}{\beta} r(x, y) + f(x)\right)}$$
$$= \frac{\pi_{\text{ref}}(y \mid x) \exp\left(\frac{1}{\beta} r(x, y)\right) \exp(f(x))}{\sum_y \pi_{\text{ref}}(y \mid x) \exp\left(\frac{1}{\beta} r(x, y)\right) \exp(f(x))}$$
$$= \pi_r(y \mid x)$$

And this conclude the proof of the lemmas.

To clarify, the first lemma was already expected to be true since in the early definitions of the Plackett-Luce family, Plackett showed that the model for ranking permutations has an under-specification issue, stating that the model for ranking permutations defines probabilities using only $r - 1$ parameters, while the number of independent probabilities required for full specification is $r! - 1$[12].

As we have seen in the Lemma 2, all reward functions within the same class result in the same optimal policy. Therefore, our main goal is simply to recover any reward function from this optimal class rather than identifying a specific one.

Based on the previously discussed results, we now present a theorem from the original paper along with its proof.

Theorem 1. *Let $\pi_{ref}(y \mid x)$ be a reference policy satisfying $\pi_{ref}(y \mid x) > 0$ for all pairs of prompts x and responses y, and let $\beta > 0$ be a strictly positive scaling parameter. Under these conditions, all reward classes consistent with the Plackett-Luce (and Bradley-Terry in particular) models can be represented with the reparameterization*

$$r(x, y) = \beta \log \frac{\pi(y \mid x)}{\pi_{\text{ref}}(y \mid x)}$$

for some model $\pi(y \mid x)$ and a given reference model $\pi_{\text{ref}}(y \mid x)$.

We start by considering a reward function $r(x, y)$ that can lead to an optimal model $_r(y|x)$ under the KL-constrained RL problem, that we already state in equation 36. Again, if we log-linearize this solution with some algebra we get to:

$$r(x, y) = \beta \log \frac{\pi_r(y \mid x)}{\pi_{\text{ref}}(y \mid x)} + \beta \log Z(x)$$

Here we notice that the partition function also depends on the reward function($Z(x) = \sum_y \pi_{\text{ref}}(y \mid x) \exp\left(\frac{1}{\beta} r(x, y)\right)$). We can perform the following transformation $r'(x, y) = f(r, \pi_{\text{ref}}, \beta)(x, y) = r(x, y) - \beta \log Z(x)$, we can check that $r'(x, y)$ is within the equivalence class of $r(x, y)$.

$$r'(x, y) = \beta \log \frac{\pi_r(y \mid x)}{\pi_{\text{ref}}(y \mid x)}$$

In the specific case, where $r(x, y)$ and $r'(x, y)$ are functions from the same class, we can say that:

$$f(r, \pi_{\text{ref}}, \beta)(x, y) = \beta \log \frac{\pi_r(y \mid x)}{\pi_{\text{ref}}(y \mid x)}$$

By Lemma 2:

$$f(r, \pi_{\text{ref}}, \beta)(x, y) = \beta \log \frac{\pi'_r(y \mid x)}{\pi_{\text{ref}}(y \mid x)} = f(r', \pi_{\text{ref}}, \beta)(x, y)$$

With this we can show that the operator f maps all reward functions from a particular equivalence class to the same reward function. Next, we have to show that for every function of an equivalent class, the reparametrized function in Theorem 1 is unique.

Proposition 1. Assume we have a reference model such that $\pi_{\text{ref}}(y \mid x) > 0$ *for all pairs of prompts* x *and answers* y, and a parameter$\beta > 0$. Then every equivalence class of reward functions, as defined in definition 1, has a unique reward function $r(x, y)$, which can be reparameterized as

$$r(x, y) = \beta \log \frac{\pi(y \mid x)}{\pi_{\text{ref}}(y \mid x)}$$

for some model $\pi(y \mid x)$.

To prove the previous proposition, we are going to proceed by contradiction. We assume two reward function from the same class $r'(x, y) = \beta \log \frac{\pi'(y|x)}{\pi_{\text{ref}}(y|x)}$ for some model $\pi'(y \mid x)$, and $r(x, y) = \beta \log \frac{\pi(y|x)}{\pi_{\text{ref}}(y|x)}$ for some model $\pi(y \mid x)$, such that $\pi \neq \pi'$. Using the definitions:

$$r'(x, y) = r(x, y) + f(x) = \beta \log \frac{\pi(y \mid x)}{\pi_{\text{ref}}(y \mid x)} + f(x)$$

Forcing everything to be inside the log we have:

$$r'(x, y) = \beta \log \frac{\pi(y \mid x) \exp\left(\frac{1}{\beta} f(x)\right)}{\pi_{\text{ref}}(y \mid x)} = \beta \log \frac{\pi'(y \mid x)}{\pi_{\text{ref}}(y \mid x)}, \forall x, \forall y$$

By the last equation we need to have $\pi(y \mid x) \exp\left(\frac{1}{\beta} f(x)\right) = \pi'(y \mid x)$. Since both $\pi(y \mid x)$ and $\pi'(y \mid x)$ are probability distributions over y, we sum over all possible values of y:

$$\sum_y \pi(y \mid x) \exp\left(\frac{1}{\beta} f(x)\right) = \sum_y \pi'(y \mid x)$$

Since $\sum_y \pi'(y \mid x) = 1$ (as π' is a probability distribution), and the same thing happens for $\sum_y \pi(y \mid x) = 1$,this simplifies to, for $\beta > 0$

$$\exp\left(\frac{1}{\beta}f(x)\right) = 1 \implies f(x) = 0$$

Thus, for the given equation to hold for all x, the function $f(x)$ must be identically zero, which contradicts our initial assumption. Therefore, $r(x, y) = r(x, y)$. This completes the proof.

This shows that every reward class has a unique reward function that can be represented as:$y = f(r, \pi_{\text{ref}}, \beta)$for any reward function in this class.

5.2 Baselines and Variance Analysis

Baselines Baselines are a standard variance reduction technique in the context of Policy Gradients [16].The idea is to subtract a baseline B from the reward $r(x)$, which does not introduce bias in the gradient estimates but helps to reduce variance. This adjustment prevents excessively large gradients that may arise due to high variance in the reward signal, making learning more stable and efficient.

In standard reinforcement learning (RL), the simplest form of baseline B is just the average of the rewards for the policy:

$$B_{\text{RL}} = \mathbb{E}_{x\sim\pi_\theta} r(x).$$

Note that since B_{RL} depends on θ, it has to be re-estimated after each gradient update [17].

A similar approach applies in Distributional Policy Gradients (DPG), where the baseline can be taken as the expectation of the energy-based model (EBM) reward function[18]:

$$B = \mathbb{E}_{x\sim\pi_\theta} \frac{P(x)}{\pi_\theta(x)} = \sum_x \pi_\theta(x) \frac{P(x)}{\pi_\theta(x)} = \sum_x P(x)$$

Where $P(x)$ is an Energy-Based Model (EBM) which assigns an unnormalized energy score to different states x. It is used to represent the underlying true distribution of data or optimal states in the environment.

Next, we show that substracting B from $R_\theta(x)$ does not introduce bias in the DPG gradient estimates.

The proof starts by rewriting the DPG gradient of its loss function with the added baseline B:

$$\mathbb{E}_{x\sim\pi_\theta} \left[(R_\theta(x) - B)\nabla_\theta \log \pi_\theta(x)\right]$$

And then by linearity:

$$\mathbb{E}_{x\sim\pi_\theta}[R_\theta(x)\nabla_\theta \log \pi_\theta(x)] - B\mathbb{E}_{x\sim\pi_\theta}[\nabla_\theta \log \pi_\theta(x)]$$

$$= \mathbb{E}_{x\sim\pi_\theta}[R_\theta(x)\nabla_\theta \log \pi_\theta(x)] - B\left[\sum_x \nabla_\theta \pi_\theta(x)\right].$$

In the second term, we notice that the gradient can be taken out of the summation, checking that

$$\nabla_\theta \sum_x \pi_\theta(x) = \nabla_\theta 1 = 0,$$

Therefore, the second term does not introduce bias because:

$$B \sum_{x} \nabla_\theta \pi_\theta(x) = 0,$$

which is exactly the original gradient of the DPG algorithm without the baseline. This confirms that subtracting B reduces variance without affecting the expected gradient, preserving unbiased learning.

Variance Analysis Although Actor-Critic methods are designed to reduce variance, they often suffer from instability in practice, making it challenging to train an initial model to convergence and obtain a useful policy.

RLHF trains a separate reward model that can be noisy and imprecise in some cases, which causes a high variance in the training leading to unstable training and hallucinations (patterns or objects that are nonexistent, creating nonsensical or inaccurate outputs) in the outputs of neural network policy. On the other hand, DPO implicitly uses a reward model without estimating it directly. Moreover, by avoiding explicit reward modeling, DPO simplifies hyperparameter tuning, making it easier to find optimal configurations for training the policy network.

This high variance or instability in standard RLHF models is targeted in some approaches by using human completion baselines. In contrast, the DPO reparameterization yields an implicit reward function that does not require any baselines given that the loss function does not take into account any reward function.DPO also eliminates the need for sampling from the LM during fine-tuning, simplifying training and making optimization more straightforward. By removing this sampling step, DPO reduces the additional variance introduced in standard models.

6 Experimental Analysis

In this section, we present and discuss the experimental results obtained by the authors in the original DPO paper [4], as well as a simpler experiment that we conducted ourselves. As expected, a full reproduction of the paper's results was not feasible due to limited computational resources.

6.1 Experiments from the Paper

To evaluate the validity of the method the authors proposed different tasks evaluations, in all the experiments we consider a dataset of preferences as already defined before $\mathcal{D} = \left\{ \left(x^{(i)}, y_w^{(i)}, y_l^{(i)} \right) \right\}_{i=1}^{N}$.

First, they checked how the DPO's efficiency in balancing reward maximization and KL-divergence minimization with the reference policy can be analyzed in comparison to common preference learning algorithms such as Unlikelihood[19], PPO[3], prefered -FT. In this experiment the IMDB dataset[20] was used, and they considered x a prefix of a movie review and the y is a positive sentiment. As the IMDB dataset that are no preference pairs defined they used a pre-trained classifier to stablish the pairs, and GPT-2-large to the fine-tune part. As can be seen in figure 1, DPO provides the highest expected reward score, reflecting its quality on the quality of the optimization, superating PPO and preferred-FT. It's good to notice that when comparing algorithms, it is essential to consider both the achieved reward and the KL divergence. A marginal increase in reward may not be ideal if it comes at the cost of a significantly higher KL discrepancy.To be more clear, this show that the DPO method can achieve higher rewards while keeping a controlled KL-divergence from the reference policy.

The next task that the authors decided to use for evaluating their method was a summarization task. To achieve this, they used the Reddit TL;DR (Too Long; Didn't Read) dataset [21], where x represents a forum post from Reddit, and the policy must generate a summary y capturing its

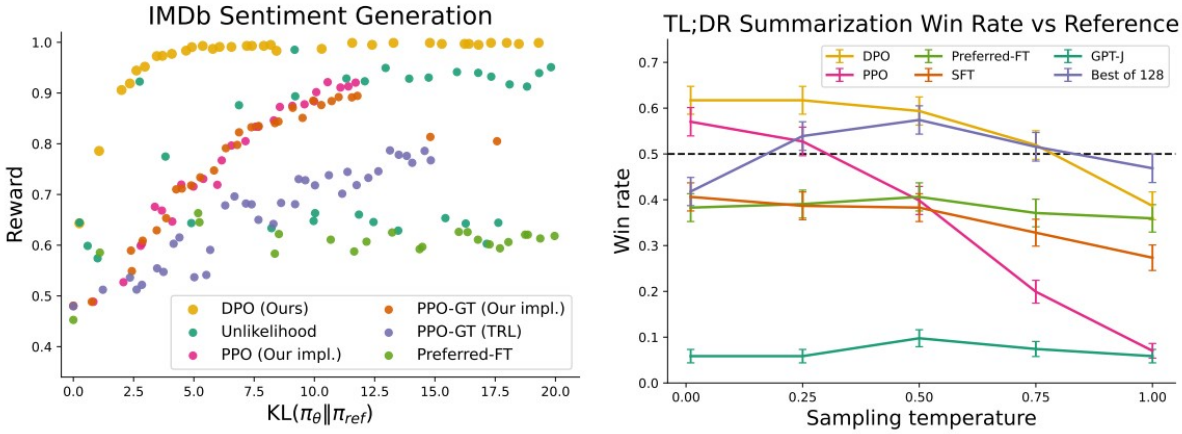

Fig. 1: Left: Expected reward vs KL to the reference policy; Right: TL;DR summarization win rates vs. human-written summaries, using GPT-4 as evaluator. Image extracted from the original paper [4]

main points. Following prior work, they leveraged human preference data collected by Stiennon et al., using a Supervised Fine-Tuning (SFT) model that was trained on human-written forum post summaries. The TRLX framework[22] was employed to apply RLHF, ensuring the generated summaries aligned with human preferences.

As in real-world the ground truth reward function is unknown, they assess the algorithms based on their win rate against a baseline policy. The authors used GPT-4 as a proxy for human evaluation, measuring summary quality in the summarization task and response helpfulness in the single-turn dialogue setting, using reference summaries in the test set as the baseline. Besides the already cited methods, in the first experiments the authors added GPT-J[23], a zero shot prompting, and Best of N- baseline—a method that selects the best response from N generated candidates, from SFT model.We can see in the right image of the figure 1, DPO achieved higher win rate values compared to the majority of other methods, except when the sampling temperature approaches one, where it is surpassed by the Best of 128. We can also check that the methods got a Win-Rate smaller than 0.5 constanltly over the sampling temperature, and the fact that PPO performance's degrades at higher tempeartures.

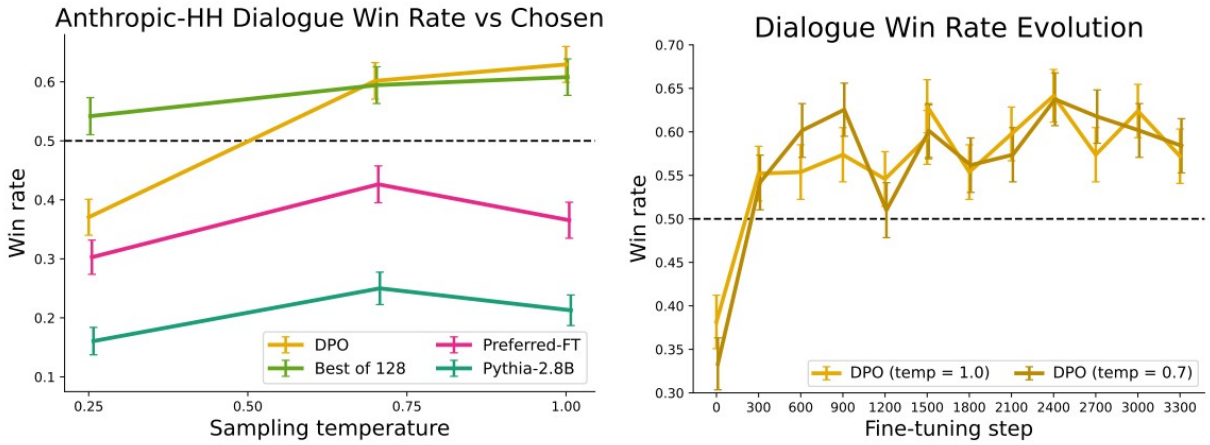

Fig. 2: Left: Win rates by GPT-4 for Anthropic-HH one-step dialogue; Right: Win Rate for different tempeartures over the fine-tuning steps. Image extracted from the original paper [4]

Next, the authors evaluate Anthropic Helpful and Harmless dialogue dataset[14] containing 170k dialogues between a human and an automated assistant. Here x is an human prompt and y a helpfull answer to the prompt. As no SFT model was available, they built their own SFT only the preferred completions. Again, using GPT-4 they computed the win rates, using the prefered answers that can be seen in the image 2. The prefered-FT and the Pythia-2.8B[24] did not performed well in this task. Among the tested methods, the only one that produced results comparable to DPO was the Best-of-128 Preferred-FT baseline. This approach, which selects the best response from 128 generated candidates, benefits from extensive sampling but comes at a significantly higher computational cost. In the right figure present in image 2, We can check that after a certain number of steps 300, DPO presents a robust and stable win rate for different temperatures.

To validate DPO under distribution shifts, the authors tested the trained policies from the Reddit dataset on news articles in the test split of the CNN/DailyMail dataset. The new input data was tested with sampling temperatures of 0 and 0.25, once again demonstrating that the DPO method outperforms PPO by 10% and 8% of win rate, respectively, as can be seen in the table 1 of the original paper [4]. The authors also verified that the GPT-4 judment is comparable with human judgments.

6.2 Our Experiments

To check the validity of the method, we built a code to perform simpler experiments than the ones done in the paper. To evaluate our model, we used a small dataset composed of simple instruction-based examples maked available by the authors of the book *Build a Large Language Model (From Scratch* [25] [26]. Each sample consists of an *instruction*, an optional *input*, a reference *output*, and two model-generated responses: a *chosen* response and a *rejected* response. For example, given the instruction *"Convert 45 kilometers to meters."*, the input is empty. The reference output is *"45 kilometers is 45000 meters."*. The model produces two responses: the chosen response *"45 kilometers is equivalent to 45000 meters."* and the rejected response *"45 kilometers is 45000 meters."*.

Due to resource limitations, the language model analyzed in this study is *GPT-2-medium*. As in all the described methods, we first build a supervised fine-tuning model. To achieve this, we utilize the *transformers* library, providing the model with the input and the prompt to predict the chosen response, using a Categorical Cross-Entropy Loss computed using difference between the predicted token probabilities and the ground truth tokens. After that, we proceed with our implementation of the DPO method: We keep as policy model *GPT-2-medium* using as reference model the previously trained SFT model. Using the previously described dataset we compute the loss of the process using the prefered and the dispreferred probabilities that the model gives to each answer as described in the theorethical part.

To evaluate the quality of the answers we employ Sentence-BERT (SBERT) to measure semantic similarity between the model-generated responses and the labeled responses. The evaluation is conducted across various sampling temperatures (β values) to analyze the impact of generation stochasticity on alignment performance. The cosine similarity of the generated responses' representations, obtained using Sentence-BERT, determines whether the generated answer is closer to the chosen response than to the rejected response. If the similarity to the chosen response is higher, we count it as a win. However, this method has a clear limitation: cosine similarity does not always accurately reflect semantic equivalence, as two sentences may have similar representations while conveying different meanings. We believe this limitation contributes to our evaluation's slightly inflated win rates. Again, since we couldn't run a huge LLM as in the original paper this solution turned to be pratical to our computational ressources.

In Figure 3, we present the mean DPO loss per epoch for different temperature values, alongside the mean KL divergence per epoch. This visualization allows us to analyze the impact of varying temperatures during the training process. as expected, in the left plot, for all temperatures the loss is descreasing.In this sense, it is also clear that for higher β values, the loss starts with higher values because the KL divergence term in the objective function has a stronger influence, forcing the model to stay closer to the reference policy. This can be seen as a regularization term.

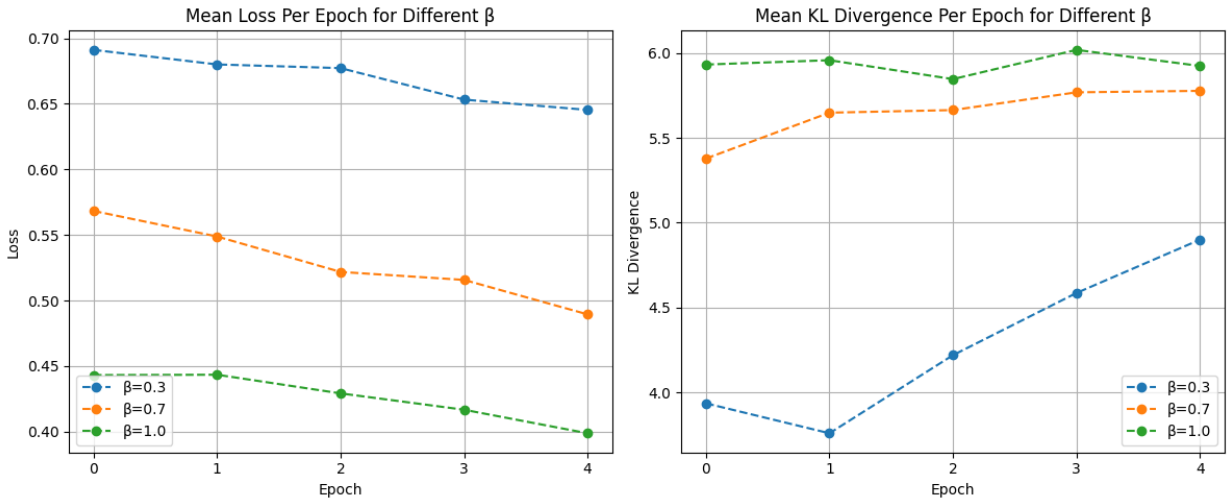

Fig. 3: Left: DPO loss over training steps per epoch; Right: KL divergence during training

A similar effect is observed in the right plot, where we can examinate the token-level KL divergence of logits between the reference model and the fine-tuned model computed across different epochs. We can also check that higher β values can lead to higher KL-divergence values, meaning that the new model diverges more from the reference model, but these values keep very similar showing the stability of the method. For $\beta = 0.3$, KL divergence increases over epochs, showing that model is diverging more from the reference model, meaning it is learning new patterns rather than just mimicking the reference model. We expect that this value reach similar values than the higher β values and can stabilize more. For $\beta = 0.7$ and $\beta = 1.0$, KL divergence is relatively stable and have very similar values.

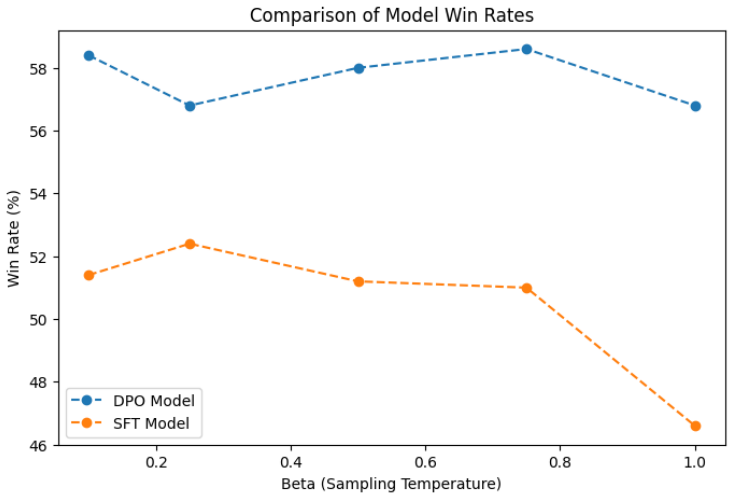

Fig. 4: Win-Rates comparisson beetween the

In image 4, we can check the difference in the win rates beetween the SFT model and the DPO alligned model. First, it is clear that the DPO alligned model achieve higher win rates values in comparisson with the SFT model. Also, we can check that, overall, the DPO model is more stable, not varying that much with the temperatures. The Win Rate of the SFT model decreases with the increasing of β, showing that it can be more sensitive to the temperature. A similar result was find in the original paper [4].

Since we are using *GPT-2*-medium, which is not a large-scale language model, some responses after applying DPO still exhibit imperfections. However, we were able to observe certain behaviors reported in previous studies, particularly when comparing the purely supervised fine-tuned model with the one that passed through a preference optimization method.

```
Prompt: What is the opposite of 'full'?

DPO Model:
Generated: The opposite of 'full' is indeed 'empty'.
Chosen: The opposite of 'full' is 'empty'.
Rejected: Obviously, the opposite of 'full' is 'empty', get it?

SFT Model:
Generated: A 'full' answer would be an answer that only includes a complete or perfect sentence.
Chosen: The opposite of 'full' is 'empty'.
Rejected: Obviously, the opposite of 'full' is 'empty', get it?
```

In the question, despite the DPO model not generating the exact chosen answer, it produced a well-aligned response, whereas the SFT model generated a completely generic and overly verbose one, as noted in previous studies [27].

7 Discussion and Posterior Works

In our implementation, due to computational limitations, we opted to work SentenceBERT to evaluate the Win Rates which can be seen as a limitation of our evaluation. Also, it is well known that *GPT-2* is not a big LMs and it can generate verbose or hallucinate answers.

Moreover, as cited by the authors this new paper, raise several questions for future work [4] like how the DPO policy generalize out of distribution problems, and what are the DPO's fundamental limitations. In the work *Is DPO Superior to PPO for LLM Alignment? A Comprehensive Study* [28] the authors aimed to answer these questions.

To start the authors remembered that in the methods that learn from an explicit reward function, it can happen that it generates high rewards without meeting the actual human preference, sometimes generating erroneous answers. To formalize the relationship between DPO and PPO, the authors introduce a Theorem, which establishes that the class of policies found via PPO is a proper subset of those found by DPO $\mathit{\Pi}_{\text{PPO}} \subset \mathit{\Pi}_{\text{DPO}}$, meaning that DPO can recover any policy optimized by PPO, as well as adittional solutions. The proof relies on showing that the minimum DPO loss is equivalent to the minimum reward learning loss, and that for any PPO-optimized policy π_{PPO}, there exists a corresponding reward function r^* such that PPO is implicitly optimizing the same objective as DPO. The authors argue that despite the fact that "DPO avoids training the reward model, it still suffers from the misspecification issue on OOD(Out of distribuition)", due to bias. They showed the validity of the theorem through experimental evaluations. Moreover, they showed that in some cases right adjustments (batch size, advantage normalization) in PPO can lead to better results than DPO.

8 Conclusion

In this work, we have reviewed the Direct Preference Optimization (DPO) method, a novel approach for learning preferences to train more aligned and effective language models. It was shown conceptually and mathematically how the training of the policy network is simpler and faster by reducing the need of training an explicit reward function that may be hard to properly define

for complex targets. In this context, we introduced the mathematical framework of the method, along with the statistical foundations on which it is based. We showed how the previously defined Reinforcement Learning From Human Preferences can be seen as Maximum Likelihood Estimation without loss of generality. Besides that we reviewed the experiments proposed by the authors alongside with experiments generated by our implementations. Finally, we discuss the questions that still need to be answered about this method and show one of its problems with a posterior work.

9 Code Availability

Our GitHub repository for the project can be found at: `https://github.com/giovanni-br/DPO-Implementation`. The experiments can be runned in the `https://github.com/giovanni-br/DPO-Implementation/blob/main/experiments.ipynb`. The file *supervised_finetuning.py* contains the SFT training, while the *dpo_finetuning.py* contains the dpo training and *evaluation.py* the evaluation experiments.

References

1. Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners, 2023.
2. Paul Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences, 2023.
3. John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.
4. Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36, 2024.
5. R. A. Bradley and M. E. Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952. Accessed 17 Feb. 2025.
6. R. Duncan Luce. *Individual Choice Behavior: A Theoretical Analysis*. Courier Corporation, 2005. Originally published in 1959, reprinted in 2005. Accessed 17 Feb. 2025.
7. Roger N. Shepard. Stimulus and response generalization: A stochastic model relating generalization to distance in psychological space. *Psychometrika*, 22(4):325–345, 1957. Accessed 17 Feb. 2025.
8. Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback, 2022.
9. Daniel M. Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B. Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences, 2020.
10. John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. Trust region policy optimization. *CoRR*, abs/1502.05477, 2015.
11. Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018.
12. R. L. Plackett. The analysis of permutations. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 24(2):193–202, 1975. Accessed 11 Feb. 2025.
13. Natasha Jaques, J. H. Shen, Asma Ghandeharioun, Craig Ferguson, Agata Lapedriza, Norman Jones, Shixiang Shane Gu, and Rosalind Picard. Human-centric dialog training via offline reinforcement learning. *arXiv preprint arXiv:2010.05848*, 2020. Accessed 17 Feb. 2025.
14. Yuntao Bai, Andy Jones, Kamal Ndousse, Anna Askell, Nova Chen, Nicholas DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, Nicholas Joseph, Saurav Kadavath, Jackson Kernion, Tristan Conerly, Sherwin El-Showk, Nelson Elhage, Zach Hatfield-Dodds, Daniel Hernandez, Thomas Hume, Samuel Johnston, Shauna Kravec, Liane Lovitt, Neel Nanda, Catherine Olsson, Dario Amodei, Tom Brown, Jack Clark, Sam McCandlish, Chris Olah, Benjamin Mann, and Jared Kaplan. Training a helpful and harmless assistant with reinforcement learning from human feedback. 2022.
15. Dongyoung Go, Tomasz Korbak, Germán Kruszewski, Jos Rozen, Nahyeon Ryu, and Marc Dymetman. Aligning language models with preferences through f-divergence minimization, 2023.
16. Evan Greensmith, Peter Bartlett, and Jonathan Baxter. Variance reduction techniques for gradient estimates in reinforcement learning. *NeurIPS Proceedings*, 14, 2001.
17. Tomasz Korbak, Hady Elsahar, Germán Kruszewski, and Marc Dymetman. On reinforcement learning and distribution matching for fine-tuning language models with no catastrophic forgetting. *arXiv preprint arXiv:2206.00761*, 2022. Accessed 17 Feb. 2025.
18. Tetiana Parshakova, Jean-Marc Andreoli, and Marc Dymetman. Distributional reinforcement learning for energy-based sequential models. *arXiv preprint arXiv:1912.08517*, 2019. Accessed 17 Feb. 2025.
19. Sean Welleck, Ilia Kulikov, Stephen Roller, Emily Dinan, Kyunghyun Cho, and Jason Weston. Neural text generation with unlikelihood training. *arXiv preprint arXiv:1908.04319*, 2019.
20. Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.
21. Michael Völske, Martin Potthast, Shafqat Umar Syed, and Benno Stein. Tl;dr: Mining reddit to learn automatic summarization. In *Proceedings of the Workshop on New Frontiers in Summarization*, pages 59–63, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.
22. Lysandre von Werra, Julian Tow, reciprocated, Sergio Matiana, Andrew Havrilla, cat state, Lorenzo Castricato, Alan, D. V. Phung, Adit Thakur, Artem Bukhtiyarov, aaronrmm, Francesco Milo, Daniel, Dylan King, Dong Shin, Edward Kim, Jeff Wei, Manuel Romero, Nikita Pochinkov, Omar Sanseviero, Ranganath Adithyan, Simon Siu, Thomas Simonini, Vuk Blagojevic, Xinyang Song, Zach Witten, alexandremuzio, and crumb. Carperai/trlx: v0.6.0: Llama (alpaca), benchmark util, t5 ilql, tests, March 2023. Zenodo Repository.

23. Ben Wang and Aran Komatsuzaki. Gpt-j-6b: A 6 billion parameter autoregressive language model, May 2021.
24. Stella Biderman, Horace Schoelkopf, Quintin Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, M. Amin Khan, Soumya Purohit, U. S. Prashanth, Edward Raff, Aleksa Skowron, Leo Sutawika, and Oskar van der Wal. Pythia: A suite for analyzing large language models across training and scaling. 2023.
25. Sebastian Raschka. *Build a Large Language Model (From Scratch)*. Manning Publications, 2024.
26. Ahmed Allam. Unveiling the hidden reward system in language models: A dive into dpo. *Allam's Blog*, January 2024.
27. Keita Saito, Akifumi Wachi, Koki Wataoka, and Youhei Akimoto. Verbosity bias in preference labeling by large language models, 2023.
28. Shusheng Xu, Wei Fu, Jiaxuan Gao, Wenjie Ye, Weilin Liu, Zhiyu Mei, Guangju Wang, Chao Yu, and Yi Wu. Is dpo superior to ppo for llm alignment? a comprehensive study, 2024.