Understanding Reference Policies in Direct Preference Optimization

Yixin Liu[1] **Pengfei Liu**[2] **Arman Cohan**[1,3]
[1]Yale University [2]Shanghai Jiao Tong University [3]Allen Institute for AI
yixin.liu@yale.edu, pengfei@sjtu.edu.cn, arman.cohan@yale.edu

Abstract

Direct Preference Optimization (DPO) has become a widely used training method for the instruction fine-tuning of large language models (LLMs). In this work, we explore an under-investigated aspect of DPO – its dependency on the reference model or policy. Such reference policies, typically instantiated as the model to be further fine-tuned, are important since they can impose an upper limit on DPO's effectiveness. Therefore, we address three related research questions in this work. First, we explore the optimal strength of the KL divergence constraint in DPO, which penalizes deviations from the reference policy, and find that DPO is sensitive to this strength. Next, we examine the necessity of the KL-constraint from the reference policies in DPO by providing both theoretical and empirical comparisons between DPO and related learning objectives, demonstrating DPO's superiority in this controlled setting. Additionally, we investigate whether DPO benefits from stronger reference policies, finding that a stronger reference policy can lead to improved performance, but only when it is similar to the model being fine-tuned. Our findings highlight the confounding role of reference policies in DPO and offer insights for best practices, while also identifying open research questions for future studies.

1 Introduction

Recently, Direct Preference Optimization (DPO) (Rafailov et al., 2023) has become a widely used training method to align pre-trained large language models with human preferences (Ouyang et al., 2022). DPO offers a significant advantage over standard supervised fine-tuning (SFT) because it learns to distinguish the quality of various candidate outputs, rather than merely relying on a single gold reference. Specifically, derived from the KL-constrained reinforcement learning (RL) setting, the training objective of DPO implicitly learns a reward model r_θ given an input x and an output y: $r_\theta(x, y) = \beta \log \frac{p_\theta(y|x)}{p_{\text{ref}}(y|x)}$, where p_θ and p_{ref} are the distributions parameterized by the LLM being fine-tuned and the reference LLM respectively, and β controls the strength of the KL divergence regularization applied from the reference LLM.

We argue that the dependency on the reference model/policy p_{ref},[1] as demonstrated by the parameterization of r_θ, is an important yet under-explored aspect of DPO. To begin with, this dependency can lead to a discrepancy between the prediction of $r_\theta(x, y)$, the optimization target, and $p_\theta(y|x)$, the learned distribution parameterized by the fine-tuned LLM. That is, given an input x and a pair of outputs y_w and y_l between which y_w has better quality, even if the reward model correctly predicts that $r_\theta(x, y_w) > r_\theta(x, y_l)$, it does not guarantee $p_\theta(y_w|x) > p_\theta(y_l|x)$, that the fine-tuned model learns to assign a higher probability to the better output (Chen et al., 2024). Moreover, since this dependency arises from a KL divergence constraint from the reference policy, any deviations from the reference policy will be penalized. This can create a performance ceiling, as the reference policy is typically instantiated as the SFT model in practice, which is meant to be further improved using DPO.

It thus seems that removing or reducing such constraints might be beneficial. Recent work (Gorbatovski et al., 2024) does find that updating the reference policy helps improve DPO's performance. However, eliminating this dependence can actually lead to performance degradation because of model degeneration (Rafailov et al., 2023). As a result, while related studies have proposed reference-policy-free training methods that have shown superior performance than DPO (Xu et al., 2023; Hong et al., 2024; Meng et al., 2024), other forms of regularization are still required in these methods.

---

[1] We will interchangeably use the terms "model" and "policy" to reflect the contexts of both LLM fine-tuning and RL.

Findings of the Association for Computational Linguistics: NAACL 2025, pages 8037–8052
April 29 - May 4, 2025 ©2025 Association for Computational Linguistics

For example, ORPO (Hong et al., 2024) introduces a maximum likelihood estimation (MLE) objective on the positive examples, while SimPO (Meng et al., 2024) highlights the importance of length normalization of the sequence log-likelihood.

These findings suggest a complicated relationship between the reference policy and the DPO performance. Therefore, we aim to further understand the effects and the role of the reference policy in DPO. To this end, we choose the task of instruction fine-tuning of LLMs (Ouyang et al., 2022) as the test bed and two series of open-source pre-trained LLMs, Tulu 2 and Mistral (Jiang et al., 2023), to analyze the reference policy in DPO on the AlpacaEval benchmark (Li et al., 2023). Specifically, we investigate three main research questions (RQs).

First, starting from the standard DPO setting where the KL-constraint regularization is applied using the SFT reference policy, we explore RQ1: **what the optimal strength of the KL-constraint should be** (§4). Our experiments suggest that the optimal KL-constraint strength is significantly smaller than that used in previous work like Zephyr (Tunstall et al., 2023) and Tulu 2 (Ivison et al., 2023). Moreover, we find that DPO is sensitive to this KL-constraint, as setting the constraint slightly smaller than the optimal value can quickly lead to performance degradation or degeneration. We therefore conduct an in-depth analysis of this sensitivity at both sequence and token levels, which reveals that (1) a small constraint is required for the model to learn to assign higher probabilities to better outputs effectively, but (2) a small constraint also leads to more extreme probability assignments to specific tokens, likely increasing model instability and causing model degeneration.

Having analyzed the standard DPO setting, we then examine the DPO formulation itself and investigate RQ2: **whether the reference policy is a necessary regularization for DPO** (§5). To this end, we compare two alternative reward parameterizations with the DPO reward parameterization $\frac{p_\theta(y|x)}{p_{\text{ref}}(y|x)}$: (1) the posterior probability $p_\theta(y|x)$ as the reward, equivalent to a reference-policy-free setting; (2) the likelihood function $p_\theta(x|y)$ as the reward, equivalent to using the prior language modeling probability $p(y)$ as the reference policy. Notably, unlike related work (Hong et al., 2024; Meng et al., 2024) that replaces the KL-constraint with other types of regularization, we focus on a more controlled comparison under the DPO formulation, which reveals new insights. Specifically, we provide proof of equivalence between the reference-policy-free preference optimization setting and the Maximum Entropy (MaxEnt) RL setting (Ziebart et al., 2008), allowing a theoretical comparison of these objectives under the RL framework. Meanwhile, our empirical experiments reveal that the alternative objectives cannot outperform DPO with the optimal KL-constraint strength identified in RQ1. Interestingly, this optimal strength is also critical for DPO to outperform reference-policy-free methods like ORPO (Hong et al., 2024), reversing their reported superiority.

Given the importance of the reference policy in the DPO objective, the third question (RQ3) we study is: **whether DPO benefits from a stronger reference policy** (§6). In detail, rather than the standard practice of using the SFT model to be fine-tuned as the reference model, we employ two more capable LLMs as the reference model. Our experiments reveal an interesting finding: stronger reference models can indeed offer more benefits than the SFT model, but only when they are compatible with the model being fine-tuned. Specifically, this compatibility likely stems from the intrinsic similarities of models, such as model architectures or pre-training corpora. However, we also find that compatibility does not require the reference model to share the same tokenizer/action space, despite DPO's capability to model dense reward functions which requires a consistent action space as demonstrated by recent work (Rafailov et al., 2024).

Our work emphasizes the confounding effect of the reference policy in DPO. Our findings shed light on the best practices for DPO: (1) when using the SFT model as the reference policy, a smaller yet sufficiently strong constraint typically improves performance; (2) a stronger reference policy can improve performance but should be compatible with the model to be fine-tuned. Meanwhile, it highlights open research questions that call for more theoretical analysis of the relationship between the reference policy and the DPO training dynamics.[2]

2 Preliminaries

2.1 Instruction Fine-tuning

Instruction fine-tuning aims to align the pre-trained language models with the users' intentions (Ouyang et al., 2022). The frequently used

---

[2] Our training scripts, model checkpoints, and datasets are released at `https://github.com/yale-nlp/refdpo`.

training methods of instruction fine-tuning include standard supervised fine-tuning (SFT), reinforcement learning (RL) methods (Stiennon et al., 2020; Ouyang et al., 2022) such as Proximal Policy Optimization (Schulman et al., 2017), and reward-based training methods (Rafailov et al., 2023; Yuan et al., 2023; Zhao et al., 2023a) such as Direct Preference Optimization (Rafailov et al., 2023). Both RL and reward-based training methods offer advantages over standard SFT by leveraging *human feedback*, typically in the form of human-annotated rewards assigned to various candidates, enabling more effective alignment of LLMs with human preferences. Therefore, they are widely adopted in the instruction fine-tuning of large language models (LLMs) such as GPT-4 (Achiam et al., 2023), Gemini (Team et al., 2023), and Llama-3 (Dubey et al., 2024).

2.2 Direct Preference Optimization

In Rafailov et al. (2023), DPO is proposed as an alternative to RL for training LLMs with human feedback. The DPO objective is derived from the KL-constrained reinforcement learning setting:

$$\max_{p_\theta} \mathbb{E}_{x\sim D, y\sim p_\theta(y|x)}[r(x,y)] - \beta \mathbb{D}_{\mathrm{KL}}[p_\theta(y|x)||p_{\mathrm{ref}}(y|x)], \tag{1}$$

where x is the input text, y is the output text, p_θ is the probability distribution parameterized by the LLM under fine-tuning with trainable parameters θ, p_{ref} is a reference distribution which is often instantiated by another LLM, β is a parameter controlling the strength of the KL divergence penalty:

$$\mathbb{D}_{\mathrm{KL}}[p_\theta||p_{\mathrm{ref}}] = \mathbb{E}_{y\sim p_\theta(y|x)}[\log \tfrac{p_\theta(y|x)}{p_{\mathrm{ref}}(y|x)}]. \tag{2}$$

$r(x, y)$ is the reward of y given x under the Bradley-Terry (BT) model (Bradley and Terry, 1952):

$$p(y_1 \succ y_2|x) = \tfrac{\exp(r(x,y_1))}{\exp(r(x,y_1))+\exp(r(x,y_2))}, \tag{3}$$

where $p(y_1 \succ y_2|x)$ is the probability of output y_1 being better than output y_2. This reward model can be optimized by maximum likelihood estimation:

$$\begin{aligned}\mathcal{L}(r) &= -\mathbb{E}_{(x,y_w,y_l)\sim D}[\log p(y_w \succ y_l|x)] \\ &= -\mathbb{E}_{(x,y_w,y_l)\sim D}[\log \sigma(r(x,y_w) - r(x,y_l))].\end{aligned} \tag{4}$$

Here x is an input in the dataset D, y_l and y_w is a pair of outputs between which y_w has better quality, and $\sigma(\cdot)$ is the sigmoid function.

DPO shows that for the aforementioned RL objective (Eq. 1), it is equivalent to optimizing the following objective: learning a reward model parameterized by the model's policy p_θ (and the reference policy p_{ref}) under the BT model:

$$\begin{aligned}&\mathcal{L}_{\mathrm{DPO}}(p_\theta; p_{\mathrm{ref}}) = \\ &-\mathbb{E}_{(x,y_w,y_l)\sim D}[\log \sigma(\beta \log \frac{p_\theta(y_w|x)}{p_{\mathrm{ref}}(y_w|x)} - \beta \log \frac{p_\theta(y_l|x)}{p_{\mathrm{ref}}(y_l|x)})],\end{aligned} \tag{5}$$

where a reward model is implicitly learned:

$$r_\theta(x, y) = \beta \log \tfrac{p_\theta(y|x)}{p_{\mathrm{ref}}(y|x)}. \tag{6}$$

We note that the reference policy p_{ref} used in DPO is typically the LLM after supervised fine-tuning, which is the same LLM that will be fine-tuned using DPO. For simplicity, we refer to this model as the SFT model.

2.3 Related Work on Preference Learning

Apart from DPO, various training methods of preference learning have been proposed recently. For example, IPO (Azar et al., 2024) has been proposed to address the potential limitation of DPO, which tends to overlook the KL-divergence regularization, making it easy to overfit. On the other hand, many preference learning algorithms are *reference-policy-free*. Among these, contrastive learning is a major category that interprets the model-predicted probability of an output as a quality score (Liu et al., 2022; Zhao et al., 2023b; Yuan et al., 2023; Xu et al., 2024). Other related methods also share similar training objectives that are based on the model-predicted probability (Xu et al., 2023; Hong et al., 2024; Meng et al., 2024). These methods have been shown to achieve better or comparable performance to DPO in various settings, without the need for a reference model. However, we note that additional normalization or regularization is still required for these methods. For example, an MLE objective on the positive examples is used by SLiC-HF (Zhao et al., 2023a), CPO (Xu et al., 2024), ORPO (Hong et al., 2024), among others. Meanwhile, SimPO (Meng et al., 2024) highlights the importance of length normalization of the sequence log-likelihood in improving method effectiveness. In this work, we focus on analyzing DPO since it has a more well-established theoretical background while presenting a comparison with reference-free training objectives in §5.

3 DPO for Instruction Fine-tuning

In this work, we center our analysis in the context of DPO's applications in instruction fine-tuning. Therefore, here we outline the experimental settings for our analysis in the following sections.

Datasets Following the setup of Zephyr (Tunstall et al., 2023), we use the UltraFeedback (Cui et al., 2023) dataset, specifically its binarized version,[3] for DPO fine-tuning. It contains 64K data examples consisting of a user instruction and a positive-negative output pair scored by GPT-4.[4]

Models We mainly use two SFT LLMs as the starting point for DPO fine-tuning from two recent works on instruction fine-tuning. The first LLM, `mistral-7b-sft-beta`[5] from Zephyr (Tunstall et al., 2023), is fine-tuned from the Mistral 7B base model (Jiang et al., 2023) on the UltraChat (Ding et al., 2023) dataset.[6] We will refer to this model as **`mistral-7b`** for simplicity. The second LLM, **`tulu-2-7b`**[7] from Tulu 2 (Ivison et al., 2023), is fine-tuned from the Llama-2-7B base model (Touvron et al., 2023) on the TULU v2 SFT mixture dataset.[8] We choose these two models for better transparency and reproducibility, since their experimental artifacts are publicly accessible.

Training Settings We follow the training recipe proposed in Tunstall et al. (2023) for the experiments, which is also adopted by Ivison et al. (2023). Specifically, the models are fine-tuned with DPO for 3 epochs on the UltraFeedback dataset, with the batch size setting to 32 and linear learning rate scheduling with warmup.[9] Tunstall et al. (2023) notes that while the models fine-tuned using DPO tend to overfit, this overfitting does not hurt the models' performance, thus they picked the final checkpoint for evaluation. However, to better understand the effects of the DPO training objective on the final model performance, we instead select the checkpoints based on their loss on the validation set of the UltraFeedback dataset.

Evaluations We mainly focus on the evaluations of the LLMs' instruction-following capabilities. To this end, we use the AlpacaEval2 (Li et al., 2023) benchmark for the main evaluation, which consists of 804 test examples for evaluating instruction-following and uses GPT-4 to evaluate the system outputs by comparing them with the outputs generated by GPT-4.[10] Notably, its length-controlled version (Dubois et al., 2024a), which mitigates the length bias of GPT-4's evaluations by predicting and canceling it, achieves a 0.98 Spearman correlation with LMSYS' Chatbot Arena (Chiang et al., 2024) evaluation results collected from crowd-sourced human participants. Therefore, we mainly report the length-controlled AlpacaEval 2 scores in the following sections.

β	`mistral-7b`	`tulu-2-7b`
∞ (original)	7.57	8.50
0.1 (previous work*)	13.42	9.20
0.1	14.03	9.38
0.05	13.29	9.96
0.02	16.06	**10.46**
0.01	**16.25**	7.86
0.005	12.36	[degenerate]

Table 1: Model performance on AlpacaEval2 with varying values of β (regularization strength with respect to the reference model). The DPO fine-tuning is started from `mistral-7b` and `tulu-2-7b` respectively. *: the DPO fine-tuned models in previous work, Zephyr (Tunstall et al., 2023) and Tulu-2 (Ivison et al., 2023).

4 RQ1: What Is the Optimal KL Constraint Strength for DPO?

The regularization from the reference policy in DPO is introduced by the KL divergence constraint (Eq. 2). Therefore, we start our analysis of reference policies by investigating the impact of this KL-constraint. Specifically, in KL-constrained RL and DPO, the weighting coefficient β controls the trade-off between maximizing the reward r and minimizing the deviation from the reference policy p_{ref} (Eq. 1). Therefore, we vary the value of β to understand the impact of this KL constraint.

4.1 Main Results

In Table 1, we show the models' performance on AlpacaEval2 that are fine-tuned using DPO from `mistral-7b` and `tulu-2-7b`. Following the original setting of DPO, we also used these two SFT models as the reference models. We found a similar trend with the two series of fine-tuned models – **a smaller KL constraint generally improves performance, until the constraint becomes too**

---

[3] The dataset is available at `https://huggingface.co/datasets/HuggingFaceH4/ultrafeedback_binarized`.

[4] Further details of datasets are in Appendix A.1.

[5] `https://huggingface.co/HuggingFaceH4/mistral-7b-sft-beta`

[6] `https://huggingface.co/datasets/HuggingFaceH4/ultrachat_200k`

[7] `https://huggingface.co/allenai/tulu-2-7b`

[8] `https://huggingface.co/datasets/allenai/tulu-v2-sft-mixture`

[9] More details are in Appendix A.2.

[10] `gpt-4-1106-preview` is used to generate the outputs and to perform the pairwise output comparison.

small and leads to performance degradation.[11]

The above observation can be intuitively explained by the need for the fine-tuned model to deviate from the reference model, which is the SFT model targeted for improvement, while maintaining necessary regularization. However, we note the algorithm's sensitivity to the strength of the KL constraint: while the studies by Zephyr (Tunstall et al., 2023) and Tulu 2 (Ivison et al., 2023) both set the value of β to be 0.1, the results in Table 1 suggest that a smaller optimal value of β may be more effective, yet with an increasing risk of performance degradation. Next, we present further analysis of this KL-constraint strength sensitivity.

4.2 Ranking Accuracy

Recent work (Chen et al., 2024) noted a discrepancy between the learned (implicit) reward model and the learned policy model in DPO in terms of their capabilities of correctly ranking candidate outputs according to their quality. Particularly, the ranking accuracy of the learned reward model (Eq. 6) is usually high, indicating that the better output y_w is assigned a higher reward. i.e., $r_\theta(x, y_w) > r_\theta(x, y_l)$. However, Chen et al. (2024) found that the ranking accuracy of the learned policy p_θ remains low even after DPO fine-tuning. They partly attributed this to the generally low ranking accuracy of the reference policy p_{ref}, noting that DPO fine-tuning rarely reverses the preference (i.e., $p(y_w|x)$ vs. $p(y_l|x)$) due to the KL-constraint.

We noticed a similar trend in our experiments. Specifically, Table 2 shows the ranking accuracy of the learned reward model r_θ and the policy model p_θ on the validation set of UltraFeedback. We observe a positive correlation between the strength of the KL constraint (β) and the ranking accuracy of p_θ – **a sufficiently small β is necessary to exceed the 50% ranking accuracy of the random oracle baseline**. Furthermore, a smaller β can lead to a higher policy ranking accuracy, however, as noted in §4.1, it can cause model degradation.[12]

β	`mistral-7b`		`tulu-2-7b`	
	r_θ	p_θ	r_θ	p_θ
∞ (original)	0.500	0.435	0.500	0.439
0.1	0.718	0.495	0.773	0.462
0.05	0.744	0.513	0.778	0.483
0.02	0.746	0.544	0.766	0.536
0.01	0.751	0.605	0.758	0.605
0.005	0.766	0.704	0.750	0.680

Table 2: Ranking accuracy of the learned reward model $r_\theta(x, y)$ and the policy model $p_\theta(y|x)$ on the Ultrafeedback validation set.

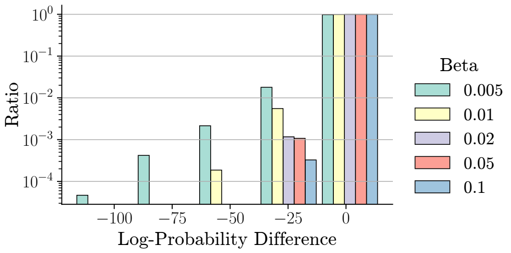

Figure 1: The average token log-probability difference (discretized) distribution between the DPO-fine-tuned and reference models (Eq. 8). The models are fine-tuned from `mistral-7b` with different values of β.

4.3 Token-Level Difference

Rafailov et al. (2024) have recently shown DPO implicitly parameterizes a dense reward function:

$$r_\theta(y^{(i)}, s^{(i)}) = \beta \log \frac{p_\theta(y^{(i)}|x, y^{(<i)})}{p_{\text{ref}}(y^{(i)}|x, y^{(<i)})}, \qquad (7)$$

where $y^{(i)}$ is the i-th token (action step) in the output y, $y^{(<i)}$ is the prefix of y before $y^{(i)}$. $s^{(i)}$ is the current "state", which is determined by the input x and the prefix $y^{(<i)}$. Intuitively, as noted by Rafailov et al. (2024), this formulation implies that DPO can learn token-level credit assignment, which is proportional to the difference between the log probabilities assigned by the trained model and the reference model. Therefore, we use the statistic, ***token log-probability difference***, to study the token-level difference between the trained and reference models:

$$\hat{r}_\theta(y^{(i)}, s^{(i)}) = \log \frac{p_\theta(y^{(i)}|x, y^{(<i)})}{p_{\text{ref}}(y^{(i)}|x, y^{(<i)})}. \qquad (8)$$

Figure 1 shows the distribution of the average log-probability difference of each token in the model's vocabulary on the UltraFeedback validation set, with models fine-tuned from `mistral-7b`. We note the following:

(1) Similar to the findings in Rafailov et al. (2024), we found the DPO fine-tuning leads to a decrease in token probabilities in general.

(2) The distribution of token-level log-probability differences is imbalanced, with a small portion of tokens receiving very different probabilities.

(3) A smaller KL constraint leads to a larger portion of tokens with large probability differences.

---

[11] Additional experiments with a 1.5B LLM are in Appendix B, which demonstrates a similar trend.

[12] Chen et al. (2024) has also noted that the ranking accuracy and the generation performance of the learned policy are not always positively correlated.

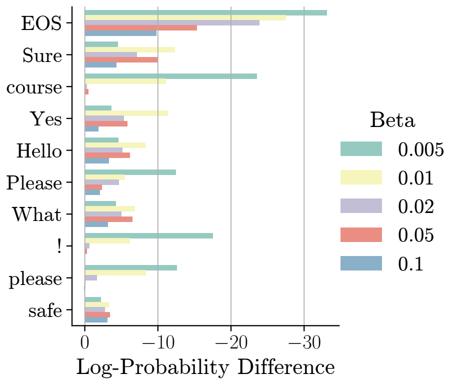

Figure 2: The tokens most downweighted on average by the models DPO-fine-tuned from `mistral-7b` with different values of β.

Figure 2 displays the top 10 tokens with the largest probability differences that appear at least 100 times. Notably, the end-of-sequence token (EOS) is the most downweighted among them. Additionally, a smaller KL-constraint results in a larger difference for the EOS token. Consequently, we observe that a smaller KL-constraint generally leads to longer outputs, suggesting that the models have learned a preference for output length.

Discussion Our token-level analysis reveals that **as the strength of the KL-constraint decreases, the DPO-fine-tuned model begins to assign significantly different probabilities to a small subset of specific tokens compared to the reference model**. The extremeness of these log-probability differences is notable: a difference of -10 indicates that the token is downweighted by $e^{10} \approx 22000$ times on average. This partially explains the sensitivity to β, as the increasing extremity of log-probability differences can destabilize the model. Moreover, it highlights a limitation of the sequence-level KL-constraint: it does not effectively prevent extreme token-level differences.

5 RQ2: Is Reference Policy a Necessary Regularization for DPO?

We now take a step back to examine the necessity of the reference policy as a regularization in DPO. As discussed in §2.3, there are already recent studies that proposed *reference-policy-free* algorithms that replace the KL-constraint with other types of regularization. We are, however, more interested in a closer comparison between DPO and similar training objectives that are either regularization-free or maintain the KL-constraint as the regularization.

5.1 Training Objectives

To this end, we investigate two related training objectives and provide theoretical analyses in the framework established by Rafailov et al. (2023): optimizing the model using (1) the posterior probability ($p_\theta(y|x)$) and (2) the likelihood function ($p_\theta(x|y)$) as the reward function in the BT model.

Posterior probability as the reward Recall that Rafailov et al. (2023) have shown that the DPO training objective learns a reward model: $r_\theta(x, y) = \beta \log \frac{p_\theta(y|x)}{p_{\text{ref}}(y|x)}$. An alternative option is to use the posterior probability with a scaling hyperparameter β as the reward function, which does not require a reference model:

$$\tilde{r}_\theta(x, y) = \beta \log p_\theta(y|x). \tag{9}$$

The training objective under the BT model (Eq. 4) then becomes

$$\begin{aligned} &\mathcal{L}_{\text{Prob}}(p_\theta) = \\ &-\mathbb{E}_{(x,y_w,y_l)\sim D}[\log \sigma(\beta \log p_\theta(y_w|x) - \beta \log p_\theta(y_l|x))]. \end{aligned} \tag{10}$$

This objective has been investigated in recent work (Xu et al., 2024) for fine-tuning LLMs for machine translation, but in Xu et al. (2024) it is used together with an MLE loss on the positive examples as a necessary regularization.

Following the analytical framework in Rafailov et al. (2023), we note that optimizing $L_{\text{Prob}}(p_\theta)$ is equivalent to optimizing the policy model under the following RL objective:

$$\max_{p_\theta} \mathbb{E}_{x\sim D, y\sim p_\theta(y|x)}[r(x, y)] + \beta \mathrm{H}[p_\theta(y|x)], \tag{11}$$

where H is the entropy function:

$$\mathrm{H}[p_\theta(y|x)] = \mathbb{E}_{y\sim p_\theta(y|x)}[-\log p_\theta(y|x)]. \tag{12}$$

The proof for the equivalence of Eq. 10 and Eq. 11 is in Appendix C, with further analyses showing that Eq. 10 shares similar properties as DPO.

We note that Eq. 11 resembles the Maximum Entropy (MaxEnt) RL setting (Ziebart et al., 2008; Eysenbach and Levine, 2021) – apart from optimizing for a higher reward, this objective discourages the model from becoming overly deterministic.

Likelihood function as the reward Apart from the posterior probability, another candidate reward function is the likelihood function $p(x|y)$. Specifically, by Bayes' theorem, we have

$$p(y|x) \propto p(x|y)p(y). \tag{13}$$

Method	**Score**	**Accuracy**	**Length**	β
original (SFT)	7.57	0.435	205.0	-
DPO (Eq. 5)	16.25	0.605	359.4	0.01
probability (Eq. 10)	12.84	0.697	744.5	100.0
likelihood (Eq. 15)	13.63	0.602	389.3	0.01

Table 3: Performance comparison of different training objectives for fine-tuning `mistral-7b`. The best performance with the optimal β is reported. The AlpacaEval2 **score**, the **ranking accuracy** of the model's probability, and the output **length** are reported.

Under the task of instruction-following, $p(x|y)$ depicts how likely the instruction x is related to the model response y. With the above factorization, we can derive the learning objective with $p_\theta(x|y)$ as the reward function:

$$p_\theta(x|y) = \frac{p_\theta(y|x)p(x)}{p_{\text{LM}}(y)}. \tag{14}$$

Here, we use a fixed LLM to parameterize the prior probability $p(y)$, which we label $p_{\text{LM}}(y)$.[13] We will show later that $p(x)$ does not need an explicit parameterization.

Using $p_\theta(x|y)$ as the reward function with a scaling factor β, the objective under the BT model (Eq. 4) becomes

$$\begin{aligned}&\mathcal{L}_{\text{Likelihood}}(p_\theta)\\ &= -\mathbb{E}_{(x,y_w,y_l)\sim D}[\log \sigma(\beta \log p_\theta(x|y_w) - \beta \log p_\theta(x|y_l))]\\ &= -\mathbb{E}_{(x,y_w,y_l)\sim D}[\log \frac{(\frac{p_\theta(y_w|x)}{p_{\text{LM}}(y_w)})^\beta}{(\frac{p_\theta(y_w|x)}{p_{\text{LM}}(y_w)})^\beta + (\frac{p_\theta(y_l|x)}{p_{\text{LM}}(y_l)})^\beta}]\end{aligned} \tag{15}$$

Eq. 15 is equivalent to using $\frac{p_\theta(y|x)}{p_{\text{LM}}(y)}$ as the reward function, which has an intuitive interpretation – the input-dependent quality score of y is its conditional probability given x, $p_\theta(y|x)$, normalized by its unconditional probability, $p_{\text{LM}}(y)$.

Furthermore, under the analytical framework of DPO, we note that Eq. 15 is equivalent to the KL-constrained RL setting (Eq. 1) with the prior probability $p_{\text{LM}}(y)$ as the reference policy. That is,

$$\max_{p_\theta} \mathbb{E}_{x\sim D, y\sim p_\theta(y|x)}[r(x,y)] - \beta \mathbb{D}_{\text{KL}}[p_\theta(y|x)||p_{\text{LM}}(y)]. \tag{16}$$

5.2 Result Analysis

We use `mistral-7b` as the base model for fine-tuning to compare DPO with the other two reward parameterizations discussed above. Table 3 summarizes the performance of different algorithms with the optimal value of β. We found that
(1) All fine-tuning methods improve the performance of the original SFT model;

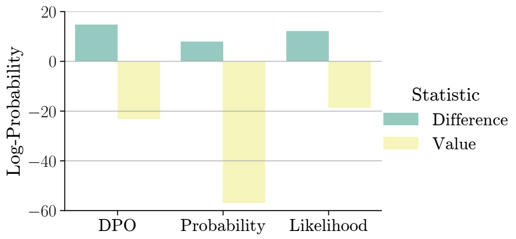

Figure 3: Average log-probability (**value**) and the average log-probability **difference** between positive and negative examples of the EOS token, as assigned by models fine-tuned with different training objectives.

(2) Using posterior probability as the reward function (Eq. 10) needs a large optimal value of β, likely because higher entropy helps prevent degeneration. (3) DPO outperforms the alternative training objectives. Notably, DPO also outperforms the reference-policy-free ORPO method (Hong et al., 2024), which scored 14.7 on the AlpacaEval2 dataset under a comparable setting.[14] This is despite ORPO's superior performance to Zephyr (Tunstall et al., 2023), which uses a suboptimal DPO configuration with β set to 0.1, as shown in Table 1.

Case Study To further understand the difference in these training objectives, we present a case study with the EOS token, as we have found in §4.3 that DPO training can significantly alter model behavior regarding the EOS token. Therefore, here we provide a further investigation. Specifically, we calculate two statistics: (1) the average log-probability assigned to the EOS token; (2) the average difference in the log-probability assigned to the EOS token between positive (y_w) and negative (y_l) examples, illustrating how the model behaves differently in these two scenarios with the EOS token.

Figure 3 presents these two statistics of models fine-tuned with different reward function parameterizations (averaged over different values of β). We note the difference between DPO and the fine-tuning method that uses posterior probability as the reward function: the former achieves a larger distinction between positive and negative examples in terms of the probability assignment to the EOS token, while the latter shows a smaller distinction, despite generally assigning more extreme probabilities to the EOS token, which leads to significantly longer outputs. This suggests that **the KL-constraint from the reference policy in DPO helps to stabilize the model behavior**. We present

---

[13] We use the SFT model in this work.

[14] The result is available at `https://tatsu-lab.github.io/alpaca_eval/`.

a qualitative case study of the outputs generated with different training objectives in Appendix D.

6 RQ3: Does DPO Benefit from Stronger Reference Policies?

In DPO, the reference policy is instantiated with the SFT model for further fine-tuning. Previous sections have highlighted the significance of reference policies in DPO. We now explore whether stronger reference policies enhance DPO.

6.1 Experimental Settings

We choose two LLMs as the stronger reference policies in DPO for the fine-tuning of `mistral-7b` and `tulu-2-7b`: (1) `mistral-7b-instruct-v0.2`,[15] (2) `llama-3-70b-instruct`.[16] These two models are more capable than the base models, achieving length-controlled AlpacaEval2 scores of 20.55 and 34.40 respectively. We refer to them as `mistral-v0.2` and `llama3` for simplicity.

6.2 Result Analysis

Table 4 displays model performance after DPO fine-tuning across different β values and reference models. For easier comparison, Table 5 lists the optimal performance using different reference models. We note the following:

(1) **A stronger reference policy in DPO fine-tuning can improve DPO's effectiveness**. Specifically, the model fine-tuned from `mistral-7b` performs best with `mistral-v0.2` as the reference, and the model fine-tuned from `tulu-2-7b` performs best with `llama3` as the refererence.

(2) However, **a stronger reference policy in DPO fine-tuning does not always lead to better performance**. Notably, DPO fine-tuning of `tulu-2-7b` with `mistral-v0.2` as the reference policy did not improve `tulu-2-7b`'s original performance.

(3) **The optimal KL constraint strength is larger with a stronger, suitable reference policy.** In particular, as shown in Table 4, the optimal value of β for `mistral-7b` and `tulu-2-7b` is 1.0 with their respective suitable reference policies. Conversely, when the SFT model is used as the reference policy, the optimal β is 0.01 or 0.02, as shown in Table 1.[17]

β	mistral-7b		tulu-2-7b	
	mistral-v0.2	llama3	mistral-v0.2	llama3
10.0	18.74	13.29	7.61	9.79
1.00	**20.25**	9.59	**7.85**	**11.17**
0.10	19.58	10.99	[degenerate]	10.31
0.01	17.18	**15.37**	[degenerate]	9.16
0.005	15.34	11.70	[degenerate]	3.29

Table 4: Model performance on AlpacaEval2 with varying values of β fine-tuned from `mistral-7b` and `tulu-2-7b`. Two reference models, `mistral-v0.2` and `llama3` are used.

	original	self	mistral-v0.2	llama3
mistral-7b	7.57	16.25	20.25	15.37
tulu-2-7b	8.50	10.46	7.85	11.17

Table 5: Model performance on AlpacaEval2. The best performance across different values of β with different reference models is compared, including the SFT model itself (self), `mistral-v0.2`, and `llama3`.

6.3 Discussion

Our investigation reveals while a stronger reference model can be helpful in DPO, it is not always the case. One possible explanation is that the reference model should be similar enough to the model under training to be compatible – `mistral-7b` performs better with `mistral-v0.2` as the reference since they are fine-tuned from the same base model, while `tulu-2-7b` performs better with `llama3` because `tulu-2-7b` is fine-tuned from `llama-2-7b`.

Regarding model similarity and the choice of the reference policy, we acknowledge the need for analytical tools and call for further research in this area. The KL divergence is a potential method, but estimating it is challenging due to the large sampling space, leading to high variance.

We note an interesting aspect in the fine-tuning of `tulu-2-7b` using `llama3` as the reference policy. Specifically, these models use different tokenizers, meaning that they do not share the same token-level action space. The fact that this combination yields further improvement suggests that action spaces do not always need to be aligned in DPO, despite related work proving that DPO can implicitly parameterize a dense reward function, which requires a consistent action space (Rafailov et al., 2024).

7 Conclusion

In this work, we studied the effects of reference policies in DPO, which can stabilize the training

---

[15] `https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.2`

[16] `https://huggingface.co/meta-llama/Meta-Llama-3-70B-Instruct`

[17] Additional experiments with a DPO-fined LLM as the reference policy are in Appendix E.

while introducing a potential performance upperbound. We found that DPO's performance is sensitive to the strength of constraints from the reference policy and demonstrated the benefits of using reference policies as a regularization in DPO. We also showed that a stronger reference policy can enhance DPO performance, but only when the reference policy is compatible. Our findings highlight the confounding role of reference policies in DPO, providing empirical insights and emphasizing the need for a more in-depth theoretical analysis. We call for future work to provide more theoretical and empirical guidelines for choosing the constraint strength from the reference policy, as well as understanding the similarity and compatibility between the model to be trained and the reference model.

Limitations

Evaluations We mainly use the AlpacaEval2 benchmark to evaluate the instruction-following capabilities of LLMs in this work, on which the LLM-based evaluation achieves a high correlation with human evaluations (Dubois et al., 2024a). The test examples on this benchmark cover a wide range of instruction types (Dubois et al., 2024b), which enhances the generalizability of the evaluation results. However, we acknowledge that the LLM-based automatic evaluation has limitations, such as low self-consistency rates in their predictions and a preference for the LLMs' own outputs (Liu et al., 2023; Wang et al., 2023; Panickssery et al., 2024). Therefore, while we did not perform human evaluations because the large number of analyses we conducted makes it cost-ineffective, we acknowledge its importance in terms of achieving more faithful evaluation results.

Model Scale The LLMs we studied in this work primarily consist of around 7 billion parameters. Due to computing resource constraints, we did not investigate LLMs with more parameters. Therefore, whether our observations are applicable on a larger scale remains to be investigated.

Acknowledgements

We thank Hamish Ivison for the helpful suggestions regarding the training details for DPO. We are grateful for the TPU compute support provided by the Google TRC program and for the OpenAI API credits support provided by OpenAI's Researcher Access Program.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*.

Mohammad Gheshlaghi Azar, Zhaohan Daniel Guo, Bilal Piot, Remi Munos, Mark Rowland, Michal Valko, and Daniele Calandriello. 2024. A general theoretical paradigm to understand learning from human preferences. In *International Conference on Artificial Intelligence and Statistics*, pages 4447–4455. PMLR.

Ralph Allan Bradley and Milton E Terry. 1952. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345.

Angelica Chen, Sadhika Malladi, Lily H Zhang, Xinyi Chen, Qiuyi Zhang, Rajesh Ranganath, and Kyunghyun Cho. 2024. Preference learning algorithms do not learn preference rankings. *arXiv preprint arXiv:2405.19534*.

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

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

Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. 2023. Enhancing chat language models by scaling high-quality instructional conversations. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3029–3051, Singapore. Association for Computational Linguistics.

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

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

Yann Dubois, Chen Xuechen Li, Rohan Taori, Tianyi Zhang, Ishaan Gulrajani, Jimmy Ba, Carlos Guestrin, Percy S Liang, and Tatsunori B Hashimoto. 2024b. Alpacafarm: A simulation framework for methods that learn from human feedback. *Advances in Neural Information Processing Systems*, 36.

Benjamin Eysenbach and Sergey Levine. 2021. Maximum entropy RL (provably) solves some robust RL problems. In *International Conference on Learning Representations*.

Alexey Gorbatovski, Boris Shaposhnikov, Alexey Malakhov, Nikita Surnachev, Yaroslav Aksenov, Ian Maksimov, Nikita Balagansky, and Daniil Gavrilov. 2024. Learn your reference model for real good alignment. *arXiv preprint arXiv:2404.09656*.

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

Hamish Ivison, Yizhong Wang, Valentina Pyatkin, Nathan Lambert, Matthew Peters, Pradeep Dasigi, Joel Jang, David Wadden, Noah A. Smith, Iz Beltagy, and Hannaneh Hajishirzi. 2023. Camels in a changing climate: Enhancing lm adaptation with tulu 2. *Preprint*, arXiv:2311.10702.

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

Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Alpacaeval: An automatic evaluator of instruction-following models. `https://github.com/tatsu-lab/alpaca_eval`.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. 2023. G-eval: NLG evaluation using gpt-4 with better human alignment. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2511–2522, Singapore. Association for Computational Linguistics.

Yixin Liu, Pengfei Liu, Dragomir Radev, and Graham Neubig. 2022. BRIO: Bringing order to abstractive summarization. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2890–2903, Dublin, Ireland. Association for Computational Linguistics.

Yu Meng, Mengzhou Xia, and Danqi Chen. 2024. SimPO: Simple preference optimization with a reference-free reward. *arXiv preprint arXiv:2405.14734*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll 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 F Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, pages 27730–27744. Curran Associates, Inc.

Arjun Panickssery, Samuel R Bowman, and Shi Feng. 2024. LLM evaluators recognize and favor their own generations. *arXiv preprint arXiv:2404.13076*.

Rafael Rafailov, Joey Hejna, Ryan Park, and Chelsea Finn. 2024. From r to q^*: Your language model is secretly a Q-function. *arXiv preprint arXiv:2404.12358*.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. In *Thirty-seventh Conference on Neural Information Processing Systems*.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. 2020. Learning to summarize with human feedback. In *Advances in Neural Information Processing Systems*, volume 33, pages 3008–3021. Curran Associates, Inc.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

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

Lewis Tunstall, Edward Beeching, Nathan Lambert, Nazneen Rajani, Kashif Rasul, Younes Belkada, Shengyi Huang, Leandro von Werra, Clémentine Fourrier, Nathan Habib, et al. 2023. Zephyr: Direct distillation of LM alignment. *arXiv preprint arXiv:2310.16944*.

Peiyi Wang, Lei Li, Liang Chen, Zefan Cai, Dawei Zhu, Binghuai Lin, Yunbo Cao, Qi Liu, Tianyu Liu, and Zhifang Sui. 2023. Large language models are not fair evaluators. *arXiv preprint arXiv:2305.17926*.

Haoran Xu, Amr Sharaf, Yunmo Chen, Weiting Tan, Lingfeng Shen, Benjamin Van Durme, Kenton Murray, and Young Jin Kim. 2024. Contrastive preference optimization: Pushing the boundaries of llm performance in machine translation. *arXiv preprint arXiv:2401.08417*.

Jing Xu, Andrew Lee, Sainbayar Sukhbaatar, and Jason Weston. 2023. Some things are more cringe than others: Preference optimization with the pairwise cringe loss. *arXiv preprint arXiv:2312.16682*.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

Hongyi Yuan, Zheng Yuan, Chuanqi Tan, Wei Wang, Songfang Huang, and Fei Huang. 2023. RRHF: Rank responses to align language models with human feedback. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Yao Zhao, Rishabh Joshi, Tianqi Liu, Misha Khalman, Mohammad Saleh, and Peter J Liu. 2023a. SLiC-HF: Sequence likelihood calibration with human feedback. *arXiv preprint arXiv:2305.10425*.

Yao Zhao, Mikhail Khalman, Rishabh Joshi, Shashi Narayan, Mohammad Saleh, and Peter J Liu. 2023b. Calibrating sequence likelihood improves conditional language generation. In *The Eleventh International Conference on Learning Representations*.

Brian D. Ziebart, Andrew Maas, J. Andrew Bagnell, and Anind K. Dey. 2008. Maximum entropy inverse reinforcement learning. In *Proc. AAAI*, pages 1433–1438.

A Additional Experimental Details

A.1 Datasets Details

Here we outline additional details regarding the datasets we used (§3). Following the setup of Zephyr (Tunstall et al., 2023), we use the UltraFeedback (Cui et al., 2023) dataset, specifically its binarized version,[18] for DPO fine-tuning. The original UltraFeedback contains 64K data examples consisting of a user instruction and 4 candidate outputs generated by different LLMs, which are scored by GPT-4. Tunstall et al. (2023) binarized UltraFeedback to modify the data format for DPO fine-tuning, by constructing a positive-negative example pair out of the 4 candidate outputs. Specifically, the output with the highest GPT-4 score is selected as the positive example y_w used in DPO fine-tuning (Eq. 5), where the negative example y_l is randomly sampled from the rest 3 candidate outputs. We note that since UltraFeedback contains LLM-generated, GPT-4 graded outputs, the instruction fine-tuning conducted on UltraFeedback is a distillation setting from more capable LLMs, instead of learning directly from human feedback. UltraFeedback is released under the MIT license.

A.2 Additional Training Details

Here we discuss additional training details in §3. We use a linear learning rate scheduler during model training, with 10% of the steps allocated for learning rate warmup and the remainder for linearly decreasing the rate. We perform checkpoint selection based on the model validation loss on the UltraFeedback validation set, with a checkpointing interval of 500 steps. We conduct model training on 8 NVIDIA RTX 6000 Ada cards with 40GB memory each. The 3-epoch training takes around 9 hours to finish.

β	AlpacaEval2 Score
SFT	4.82
0.1	4.75
0.05	5.00
0.02	**5.89**
0.01	4.91
0.005	5.52
0.001	[degenerate]

Table 6: Model performance on AlpacaEval2 with varying values of β (regularization strength with respect to the reference model). The DPO fine-tuning is started from an SFT checkpoint fine-tuned from qwen2-1.5b.

B Additional Experiments for RQ1

In §4, we study our RQ1: what the optimal strength of the KL-constraint should be, using two series of LLMs with 7B parameters. Here, we present additional results with a 1.5B pre-trained LLM, **qwen2-1.5b** (Yang et al., 2024).[19]. The training setting for qwen2-1.5b is the same as the ones used in §4, except that we trained the SFT model using the UltraChat dataset.

Table 6 presents experimental results, showing a similar trend as the experiments in §4.1 – a small KL-constraint helps the model to achieve the optimal performance. Notably, the optimal strength of the KL-constraint identified is the same as the one for `tulu-2-7b`.

C Extending DPO: Posterior Probability as Reward Function

Following the analytical framework in Rafailov et al. (2023), we now prove the equivalence of Eq. 10 and Eq. 11 as previously discussed in §5.1.

C.1 Optimum of RL with Maximum Entropy

We will first derive the optimal policy, p^*, for the RL with Maximum Entropy setting (Eq. 11):

$$\max_p \mathbb{E}_{x\sim D, y\sim p(y|x)}[r(x,y)] + \beta \mathrm{H}[p(y|x)]. \quad (17)$$

---

[18] The dataset is available at `https://huggingface.co/datasets/HuggingFaceH4/ultrafeedback_binarized`.

[19] `https://huggingface.co/Qwen/Qwen2-1.5B`

Given any reward function r, we have

$$\begin{aligned}
&\max_p \mathbb{E}_{x\sim D, y\sim p(y|x)}[r(x,y)] + \beta \mathrm{H}[p(y|x)] \\
=&\max_p \mathbb{E}_{x\sim D, y\sim p(y|x)}[r(x,y)] - \beta \log p(y|x)] \\
=&\min_p \mathbb{E}_{x\sim D, y\sim p(y|x)}[\log p(y|x) - \frac{1}{\beta} r(x,y)] \\
=&\min_p \mathbb{E}_{x\sim D, y\sim p(y|x)}[\log \frac{p(y|x)}{\frac{1}{Z(x)}\exp(\frac{1}{\beta} r(x,y))} - \log Z(x)],
\end{aligned} \tag{18}$$

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

$$Z(x) = \textstyle\sum_y \exp(\frac{1}{\beta} r(x,y)). \tag{19}$$

As $Z(x)$ is only a function of x and does not depend on p, we can define the following probability distribution:

$$p^*(y|x) = \tfrac{1}{Z(x)} \exp(\tfrac{1}{\beta} r(x,y)). \tag{20}$$

Since $Z(x)$ is not a function of y, we can reorganize the final objective of Eq. 18 as:

$$\begin{aligned}
&\min_p \mathbb{E}_{x\sim D}[\mathbb{E}_{y\sim p(y|x)}[\log \frac{p(y|x)}{p^*(y|x)}] - \log Z(x)] \\
=&\min_p \mathbb{E}_{x\sim D}[\mathbb{D}_{\mathrm{KL}}[p(y|x)||p^*(y|x)] - \log Z(x)]
\end{aligned} \tag{21}$$

Since $Z(x)$ does not depend on $p(y|x)$, the final objective of Eq. 21 is equivalent to

$$\min_p \mathbb{E}_{x\sim D}[\mathbb{D}_{\mathrm{KL}}[p(y|x)||p^*(y|x)]] \tag{22}$$

By Gibbs' inequality, the KL-divergence achieves the minimum of 0 if and only if the two distributions are identical. Therefore, we have the optimal solution of Eq. 22 (and Eq. 11) as

$$p(y|x) = p^*(y|x) = \tfrac{1}{Z(x)} \exp(\tfrac{1}{\beta} r(x,y)). \tag{23}$$

C.2 Extending DPO Objective for RL with Maximum Entropy

We can now express the reward function $r(x,y)$ under the RL with maximum entropy using the optimal policy $p^*(y|x)$ by rearranging Eq. 23:

$$r^*(x,y) = \beta \log p^*(y|x) + \beta \log Z(x) \tag{24}$$

Therefore, we know that the optimal policy p^* under the BT model (Eq. 3) satisfies the following preference model:

$$\begin{aligned}
&p^*(y_1 \succ y_2|x) \\
=&\frac{\exp(\beta \log p^*(y_1|x) + \beta \log Z(x))}{\exp(\beta \log p^*(y_1|x) + \beta \log Z(x)) + \exp(\beta \log p^*(y_2|x) + \beta \log Z(x))} \\
=&\frac{1}{1 + \exp(\beta \log p^*(y_1|x) - \beta \log p^*(y_2|x))} \\
=&\sigma(\beta \log p^*(y_1|x) - \beta \log p^*(y_2|x)).
\end{aligned} \tag{25}$$

We can now derive the maximum likelihood estimation objective based on Eq. 25 for the model p_θ:

$$\min_{p_\theta} - \mathbb{E}_{(x,y_w,y_l)\sim D}[\log \sigma(\beta \log p_\theta(y_w|x) - \beta \log p_\theta(y_l|x))] \tag{26}$$

This is exactly the objective function we defined in §5.1 that uses the posterior probability as the reward function (Eq. 10).

C.3 Further Theoretical Analysis

We continue to follow the analytical framework of DPO to discuss the theoretical properties of the training objective defined in Eq. 26 (and Eq. 10). Specifically, as we discussed in §5.1, the training objective of Eq. 26 is equivalent to a BT model with the reparameterization:

$$r_\theta(x,y) = \beta \log p_\theta(y|x). \tag{27}$$

Following the proof framework in Rafailov et al. (2023), we will now prove:
all reward classes under the BT model can be represented with the reparameterization $r(x,y) = \beta \log p(y|x)$ for some model $p(y|x)$.

Preparation Our proof uses the definition of the equivalent class of reward functions given by Rafailov et al. (2023):
"*Two reward functions $r(x,y)$ and $r'(x,y)$ are equivalent iff $r(x,y) - r'(x,y) = f(x)$ for some function f.*"

We also note a lemma given by Rafailov et al. (2023).
"*Under the Bradley-Terry preference framework, two reward functions from the same class induce the same preference distribution.*"

We now provide another lemma as an extension of a similar lemma in Rafailov et al. (2023):
Two reward functions from the same equivalent class induce the same optimal policy under the RL with Maximum Entropy setting.
Proof. Given two reward functions from the same class, such that $r'(x,y) = r(x,y) + f(x)$, from Eq. 23 we know that r' induces an optimal policy $p_{r'}$:

$$\begin{aligned}
p_{r'}(y|x) &= \frac{1}{\sum_y \exp(\frac{1}{\beta} r'(x,y))} \exp(\frac{1}{\beta} r'(x,y)) \\
&= \frac{1}{\sum_y \exp(\frac{1}{\beta}(r(x,y) + f(x)))} \exp(\frac{1}{\beta}(r(x,y) + f(x))) \\
&= \frac{1}{\exp(\frac{1}{\beta} f(x)) \sum_y \exp(\frac{1}{\beta} r(x,y))} \exp(\frac{1}{\beta} r(x,y)) \exp(\frac{1}{\beta} f(x)) \\
&= \frac{1}{\sum_y \exp(\frac{1}{\beta} r(x,y))} \exp(\frac{1}{\beta} r(x,y)) \\
&= p_r(y|x).
\end{aligned} \tag{28}$$

Proof end. As Rafailov et al. (2023) suggested, this lemma indicates that for a certain reward equivalence class, any arbitrary reward function within the class will induce the same optimal policy.

Proof For any reward function $r(x, y)$, from Appendix C.1, we know it induces the following optimal policy according to Eq. 23:

$$p_r(y|x) = \frac{1}{Z(x)} \exp(\frac{1}{\beta} r(x, y)). \tag{29}$$

Therefore, we have

$$r(x, y) = \beta \log p_r(y|x) + \beta \log Z(x), \tag{30}$$

where $Z(x) = \sum_y \exp(\frac{1}{\beta} r(x, y))$. We now define a projection f:

$$f(r; \beta) = r(x, y) - \beta \log \sum_y \exp(\frac{1}{\beta} r(x, y)), \tag{31}$$

which projects r into a new reward function:

$$\begin{aligned} r'(x, y) &= f(r; \beta)(x, y) \\ &= r(x, y) - \beta \log \sum_y \exp(\frac{1}{\beta} r(x, y)) \\ &= \beta \log p_r(x, y). \end{aligned} \tag{32}$$

We can observe that r' is within the equivalence class of r since the projector f only depends on x. Our proof is therefore complete.

D Output Examples from Different Training Objectives

We show a specific output example for each of the training objectives investigated in §5. Compared with the output generated from the DPO objective in Table 8, the posterior probability objective (Eq. 10) produces longer outputs with more complex structures that appear overly verbose in Table 9. Conversely, the likelihood function objective (Eq. 15) results in less detailed outputs, as shown in Table 10.

E Additional Experiments for RQ3

In §6, we study our RQ3: whether DPO benefits from a stronger reference policy, using two stronger reference LLMs. Here, we present additional experiments with another option of a stronger LLM – an LLM that has already been trained using DPO. Specifically, we use the optimal DPO checkpoint fine-tuned from `mistral-7b` as the reference model, which is trained with the strength of the KL-constraint set to 0.01 (§4.1). This model is then used to fine-tune `mistral-7b` again.

β	AlpacaEval2 Score
reference (β = 0.01)	16.24
10.0	15.20
1.00	**16.75**
0.10	14.99
0.01	15.07
0.005	14.80

Table 7: Model performance on AlpacaEval2 with varying values of β. The DPO fine-tuning is started from `mistral-7b`, and the reference model is the checkpoint that is already fine-tuned using DPO from `mistral-7b`.

The experimental results are presented in Table 7, aligned with our observations in §6 – the optimal strength of the KL-constraint is larger when a stronger, suitable reference policy is used. Specifically, the optimal value of β identified is 1.0, much larger than the optimal value identified in §4.1, 0.01. Interestingly, the optimal fine-tuned model under this setting outperforms the reference policy used, further demonstrating the benefit of stronger reference policies.

Instruction
I like to host guests at my home from time to time, and I am gathering recipes of different dishes and drinks to keep things interesting. I am interested in trying some Somali dishes. Can you give me a recipe for Canjeero?

Output
Certainly! Here's a traditional recipe for Canjeero (also known as Cannoli or Xaaj maanta in Somali), a popular Somali flatbread made with teff flour or a mix of teff and wheat flour.

Ingredients:
- 2 cups teff flour (you can find this at specialty stores or online)
- 1 cup wheat flour (or more teff flour if you prefer a stronger teff flavor)
- 2 tsp salt
- 1 package or 2 1/4 tsp (7g) active dry yeast
- 3 cups warm water (around 110°F)

Instructions:

1. In a large mixing bowl, combine the teff flour, wheat flour, and salt. Mix well.

2. In a separate bowl, dissolve the yeast in the warm water. Let it sit for 5 minutes until it becomes frothy.

3. Pour the yeast mixture into the flour mixture. Mix well until a smooth, sticky dough forms.

4. Let the dough rest in a warm place for 1-2 hours until it doubles in size.

5. When the dough has risen, divide it into six equal portions. Shape each portion into a ball.

6. Preheat a flat griddle or a non-stick pan over medium heat.

7. Take one ball of the dough and flatten it into a circle, about 1/4 inch thick.

8. Place the flattened dough on the hot griddle or pan. Cook for 2-3 minutes until the bottom is golden brown. Flip the Canjeero and cook for another 1-2 minutes until the other side is also golden brown.

9. Remove the Canjeero from the griddle or pan and keep it warm and moist by covering it with a clean towel or aluminum foil.

10. Repeat steps 7-9 with the remaining dough portions.

Canjeero is often served with a stew or soup. Enjoy!

Table 8: Output example with the DPO objective (Eq. 5).

Instruction

I like to host guests at my home from time to time, and I am gathering recipes of different dishes and drinks to keep things interesting. I am interested in trying some Somali dishes. Can you give me a recipe for Canjeero?

Output

Here's a traditional and delicious Somali recipe for Canjeero (also known as Canjelo or Canjif) - a popular flatbread that's often enjoyed for breakfast or as a snack. This dish is versatile and can be customized with various toppings to suit your preferences. Here's how to make it:

Ingredients:
1. 2 cups all-purpose or bread flour
2. 1/2 cup fine semolina (sooji or rava)
3. 1 tsp salt
4. 1 1/2 cups water (or as needed)
5. 2 tbsp vegetable oil (or ghee/clarified butter)
6. Toppings of your choice:
    a. Yogurt
    b. Fresh or dried fruits (bananas, mangoes, dates, raisins)
    c. Honey or sugar
    d. Cardamom powder
    e. Peanuts or other nuts
    f. Qaat (a popular Somali leaf) - optional and not commonly found outside of Somalia

Instructions:

1. In a large mixing bowl, combine the all-purpose or bread flour, semolina, and salt. Mix well to combine.

2. Gradually pour in the water while stirring with a wooden or plastic spoon. Be cautious while adding water, as the dough can become too sticky if excess water is added. Add water until the dough comes together as a smooth ball.

3. Knead the dough for 3-4 minutes until it becomes soft and elastic. Cover the bowl with a damp cloth or plastic wrap and let the dough rest for 10-15 minutes. This allows the gluten to develop, resulting in a fluffier bread.

4. Heat a non-stick or cast-iron skillet on medium-high heat. Add the vegetable oil (or ghee/clarified butter) and let it heat up.

5. Divide the dough into equal-sized balls (usually about 6-8 pieces). Flatten each ball into a disc using your palms. Make sure the disc is not too thin, as this will result in a crispier bread.

6. Once the skillet is hot, carefully place the flattened discs into the skillet. Do not overcrowd the skillet; cook the bread in batches if necessary. Cook the bread for 1-2 minutes on each side, or until golden brown spots appear. Press down lightly on the bread with a spatula to fluff it up.

7. Remove the Canjeero from the skillet and place it on a plate. Repeat the process with the remaining dough balls.

8. Serve the hot Canjeero with your choice of toppings. Here are some popular combinations:
    a. Yogurt and honey or sugar
    b. Fresh fruit and cardamom powder
    c. Qaat and peanuts
    d. Peanut butter and banana
    e. Honey and sliced almonds
    f. Cinnamon and raisins

Note: Canjeero is a versatile dish that can be enjoyed any time of day. It's often served with tea or coffee as a snack or as part of a larger meal. You can also experiment with different types of flour, such as whole-wheat or spelt, to create unique flavors.

Enjoy your homemade Canjeero! This dish is a staple in Somali cuisine and is a great way to explore the rich flavors and traditions of this vibrant culture. If you have any questions or need further assistance, please don't hesitate to ask. Happy cooking!

Table 9: Output example with the posterior probability objective (Eq. 10).

Instruction

I like to host guests at my home from time to time, and I am gathering recipes of different dishes and drinks to keep things interesting. I am interested in trying some Somali dishes. Can you give me a recipe for Canjeero?

Output

Certainly! Here’s a simple recipe for Canjeero (Somali flatbread) that serves 4-6 people:

Ingredients:
- 2 cups (240 grams) of Somali flour (also known as “soor” or “sorghum flour”)
- 2 cups (480 ml) of water
- 1/2 teaspoon of salt

Instructions:

1. In a large mixing bowl, combine the somali flour and salt.

2. Gradually add the water to the flour mixture while stirring with a wooden spoon or spatula until the mixture is smooth, free of lumps, and has a thick consistency.

3. Cover the bowl with a clean kitchen towel or plastic wrap and let it rest at room temperature for at least 1 hour (or up to 24 hours) to allow the flour to hydrate and the gluten to develop.

4. When ready to make the canjeero, preheat a non-stick pan or griddle over medium heat.

5. Using a measuring cup or ladle, pour about 1/3 to 1/2 cup of batter into the center of the pan.

6. Immediately, use the back of the ladle or a spatula to gently spread the batter into a thin, round shape (about 6-8 inches in diameter).

7. Let the canjeero cook for 1-2 minutes or until the bottom is golden brown and the top is beginning to set.

8. Flip the canjeero over and let it cook for another 30 seconds to 1 minute on the other side, or until golden brown.

9. Repeat with the remaining batter, using more oil or butter to lubricate the pan as needed.

10. Serve the canjeero warm with a side of ghee (clarified butter) or other Somali stews or sauces like “Wat” or “Xalwo.”

Enjoy your traditional Somali Canjeero!

Table 10: Output example with the likelihood function objective (Eq. 15).