LLM Project Report
Direct Preference Optimization: Your Language Model is Secretly a Reward Model

Rebecca El Chidiac, Ruben Ifrah

December 2025

Contents

Introduction / Use of LLMs for this project

The GitHub repository for this project project can be found at: [1].

To make explicit the use of LLM's for this project, here is a detailed overview of the part of the project that were conducted using Google's Gemini:

- assistance for correct run of the code on Mesonet server: help with writing .sh jobs, along with a lot of debbuging;
- coding several helper functions and scripts such as logging of the metrics in the correct .json format or wrapping our data using a custom 'DPODataCollator' class;
- confirming the analysis we conducted on the influence of β in the optimization process during training.
- We highlight the fact that our main goal was to implement the loss, the metrics and the DPO training procedure ourselves and that Gemini was used as a "supervisor" to correct the implementation (add gradient clipping, accelerator wrapping, optimization tricks).

1 Presentation of the article

The paper [2] we worked on focuses on an alternative to Reinforcement Learning from Human Feedback (RLHF): Direct Preference Optimization (DPO). It leverages a parametrization of the reward model, enabling the extraction of the corresponding optimal policy in closed form. This approach allows the standard RLHF problem to be solved using only a simple classification loss, bypassing the complexity of PPO.

1.1 Motivation: Why Preference-Based Alignment?

Large Language Models (LLMs) have demonstrated remarkable capabilities in natural language understanding and generation. However, despite their scale and fluency, such models are not inherently aligned with human values. Pretraining objectives are typically based on next-token prediction over large amounts of data, which encourages linguistic coherence and factual recall but does not explicitly encode notions of truthfulness, safety, or usefulness from a human perspective.

As a consequence, raw pretrained models often exhibit undesirable behaviors. These include the generation of incorrect or misleading information, the production of harmful or unsafe content, and the amplification of social biases or toxic language present in the training data.

Preference-based alignment addresses this gap by directly incorporating human judgments into the training process, thereby steering LLM behavior toward human values.

1.2 RLHF framework

Reinforcement Learning from Human Feedback (RLHF) is a standard framework for aligning large language models with human preferences. It typically consists of three main stages.

1.2.1 Supervised Fine-Tuning (SFT)

Starting from a pretrained language model, supervised fine-tuning (SFT) is performed on a dataset of high-quality human demonstrations. This falls in the supervised learning context, where the data consists of pairs of *prompts* x_i and *expected outputs* y_i. The dataset $\mathcal{D} = (x_i, y_i)_{1:n}$ is typically obtained by contracting labelers who are asked to provide prompts along with expected answers [3]. This yields a policy π_{SFT} trained by maximizing the log-likelihood of human-written responses:

$$\max_{\theta_{SFT}} \mathbb{E}_{(x,y)} \left[\log \pi_{\theta_{SFT}}(y \mid x) \right]. \tag{1}$$

1.2.2 Reward Model Learning

To incorporate human preferences, the SFT policy is used to generate pairs of candidate responses $(y_1, y_2) \sim \pi_{\text{SFT}}(\cdot \mid x)$ for a given prompt x. A human annotator then selects the preferred response y_w over the less preferred one y_l.

This yields a dataset of pairwise preferences $\mathcal{D}_P = \{(x^{(i)}, y_w^{(i)}, y_l^{(i)})\}_{i=1}^N$. Following the Bradley-Terry model, human preferences are assumed to satisfy:

$$p^*(y_w \succ y_l \mid x) = \sigma\big(r^*(x, y_w) - r^*(x, y_l)\big), \tag{2}$$

where $r^*(x, y)$ is an unknown scalar reward function and $\sigma(z) = (1 + e^{-z})^{-1}$ is the logistic function. A parametric reward model $r_\phi(x, y)$ is trained by minimizing the negative log-likelihood:

$$\mathcal{L}_R(r_\phi) = -\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}_P}\left[\log \sigma\big(r_\phi(x, y_w) - r_\phi(x, y_l)\big)\right]. \tag{3}$$

1.2.3 Policy Optimization

Given the learned reward model r_ϕ, the policy is further optimized using reinforcement learning. The RLHF objective is:

$$\max_{\pi_\theta} \mathbb{E}_{y\sim\pi_\theta(\cdot|x)}[r_\phi(x, y)] - \beta\, D_{\text{KL}}(\pi_\theta(\cdot|x) \,\|\, \pi_{\text{ref}}(\cdot|x)), \tag{4}$$

where π_{ref} is typically the SFT policy and $\beta > 0$ controls the strength of the KL regularization towards the reference model. In practice, this optimization constitutes the main computational and engineering bottleneck of RLHF, typically requiring PPO (Proximal Policy Optimization).

1.3 Direct Preference Optimization (DPO)

1.3.1 Key Idea and Derivation

The authors of DPO propose a change of variables that eliminates the need for an explicit reward model. Recall the RL objective for a fixed reward function r:

$$\max_{\pi} \mathbb{E}_{y\sim\pi}[r(x, y)] - \beta D_{\text{KL}}(\pi||\pi_{\text{ref}}). \tag{5}$$

It is known (from Gibbs distributions) that the optimal solution to this KL-regularized maximization has the closed form:

$$\pi^*(y|x) \propto \pi_{\text{ref}}(y|x) \exp\left(\frac{1}{\beta} r(x, y)\right). \tag{6}$$

Specifically, we can write:

$$\pi^*(y|x) = \frac{1}{Z(x)} \pi_{\text{ref}}(y|x) \exp\left(\frac{1}{\beta} r(x, y)\right). \tag{7}$$

Taking the logarithm and rearranging terms reveals an expression for the reward function in terms of the optimal policy:

$$\begin{aligned}
\log \pi^*(y|x) &= \log \pi_{\text{ref}}(y|x) + \frac{1}{\beta} r(x, y) - \log Z(x) \\
\frac{1}{\beta} r(x, y) &= \log \frac{\pi^*(y|x)}{\pi_{\text{ref}}(y|x)} + \log Z(x) \\
r(x, y) &= \beta \log \frac{\pi^*(y|x)}{\pi_{\text{ref}}(y|x)} + \beta \log Z(x).
\end{aligned}$$

This is the crucial step: we can express the reward $r(x, y)$ solely in terms of the optimal policy π^*, the reference policy π_{ref}, and a partition function $Z(x)$ that depends only on the prompt.

Now, recall the Bradley-Terry preference model:

$$p^*(y_w \succ y_l \mid x) = \sigma\big(r^*(x, y_w) - r^*(x, y_l)\big). \tag{8}$$

If we assume our optimal policy π^* generates the rewards r^*, we can substitute the expression for $r(x, y)$ derived above into the Bradley-Terry model:

$$\begin{aligned}
r^*(x, y_w) - r^*(x, y_l) &= \left(\beta \log \frac{\pi^*(y_w|x)}{\pi_{\text{ref}}(y_w|x)} + \beta \log Z(x)\right) - \left(\beta \log \frac{\pi^*(y_l|x)}{\pi_{\text{ref}}(y_l|x)} + \beta \log Z(x)\right) \\
&= \beta \log \frac{\pi^*(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \beta \log \frac{\pi^*(y_l|x)}{\pi_{\text{ref}}(y_l|x)}.
\end{aligned}$$

Notice that the partition function $Z(x)$ cancels out! We can now simply parameterize the optimal policy π^* with our network π_θ. The probability of preferring the chosen response becomes:

$$p_\theta(y_w \succ y_l \mid x) = \sigma\left(\beta \log \frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \beta \log \frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)}\right). \tag{9}$$

Optimizing the likelihood of the preference data under this model gives the DPO loss:

$$\mathcal{L}_{\text{DPO}}(\pi_\theta) = -\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}}\left[\log \sigma\left(\beta \log \frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \beta \log \frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)}\right)\right]. \tag{10}$$

This objective corresponds to a simple binary cross-entropy loss where the "logits" are the scaled log-ratios of the policy and reference probabilities.

1.3.2 Implicit Reward Interpretation

An important consequence of DPO is that the learned policy implicitly defines a reward function:

$$\hat{r}_\theta(x, y) = \beta \log \frac{\pi_\theta(y|x)}{\pi_{\text{ref}}(y|x)}. \tag{11}$$

Therefore, optimizing the DPO objective implicitly recovers a reward function consistent with the observed human preferences. This observation motivates the paper's title: *the language model itself acts as a reward model.*

1.4 Qualitative comparison: DPO vs RLHF

Although DPO and classical RLHF are grounded in the same preference-based alignment framework, they differ substantially in their practical implementation and computational characteristics. Table 1 summarizes a qualitative comparison.

Criterion	**RLHF**	**DPO**
Training pipeline complexity	High	Low
Need for reward model	Yes	No
Stability	Sensitive / unstable	Generally stable
Compute requirements	Very high	Low–moderate
Alignment quality	High (SOTA)	Comparable, often similar
Hyperparameter tuning	Heavy	Lighter (mostly β)
Implementation difficulty	Hard (RL)	Easy (log-loss)

Table 1: Qualitative comparison between classical RLHF and Direct Preference Optimization (DPO).

2 Implementation framework

We implemented the Direct Preference Optimization (DPO) framework entirely from scratch using PyTorch, relying on the Hugging Face ecosystem while deliberately avoiding high-level "black-box" trainers such as TRL. This design choice allowed full control over the DPO objective, gradient computation, and logging, which was essential for interpretability and rigorous experimental analysis. The full implementation is publicly available on GitHub [1].

2.1 Infrastructure and design choices

All experiments were conducted on the Mesonet Juliet high-performance computing cluster. To ensure reproducibility and enable detailed post-hoc analysis of the optimization dynamics, we adopted the following infrastructure choices:

- **Streaming metric logging**: training metrics (loss, implicit rewards for chosen and rejected samples, preference margins, accuracy, and log-probability ratios) are written incrementally to local `.jsonl` files.

- **Offline analysis**: this format allows for efficient aggregation and visualization of results without requiring persistent internet connectivity or external services, which is particularly suited to HPC environments.

This setup enabled precise inspection of the effect of the DPO temperature parameter β on training stability and alignment behavior.

2.2 Dataset

We used the **HuggingFaceH4/ultrafeedback_binarized** dataset [4], a widely adopted benchmark for preference-based alignment. The dataset consists of high-quality human preference annotations expressed as pairwise comparisons.

- **Training split**: `train_prefs` (approximately 61k preference pairs).
- **Evaluation split**: `test_prefs` (approximately 2k held-out pairs).

Each instance contains a prompt along with a preferred ("chosen") and a dispreferred ("rejected") response. An example is shown below:

```
{
  "prompt": "How do I make a cake?",
  "chosen": [
    {"role": "user", "content": "How do I make a cake?"},
    {"role": "assistant", "content": "Here is a simple recipe for a sponge cake..."}
  ],
  "rejected": [
    {"role": "user", "content": "How do I make a cake?"},
    {"role": "assistant", "content": "Go buy one at the store."}
  ]
}
```

3 Experiment: the role of the β parameter

In the original DPO paper, the authors fix $\beta = 0.1$ in most experiments, with little discussion of how this parameter should scale with model capacity. The important aspect of β in DPO is that it is *not* interpreted as the weight of an explicit KL-divergence penalty, as is common in PPO-style RLHF objectives. Instead, DPO optimizes a binary classification loss over preference pairs, where β appears as a multiplicative factor inside the logit of a sigmoid function.

The DPO loss for a preference pair (y_w, y_l) can be written as

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

which shows that β directly rescales the log-probability difference that defines preference satisfaction.

Equivalently, DPO induces an *implicit reward model*

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

and trains the policy to rank preferred responses higher than rejected ones under this reward. From this perspective, β controls the saturation rate of the loss function:

- High β amplifies the log-probability difference. The sigmoid function saturates ($\sigma \to 1$) even for small margins, causing gradients to vanish early.
- Low β compresses the difference. The sigmoid remains in its linear regime, providing persistent gradients even after substantial optimization.

As a consequence, smaller β values do not constrain the policy more tightly to the reference model.

Given the absence of theoretical guidance on scaling β, we conducted an empirical investigation across two model scales: 124M (GPT-2) and 7B (Mistral) parameters.

3.1 First step: exploratory sweep on a small model (GPT-2)

To explore the stability regime of DPO training at low computational cost, we first performed a wide logarithmic sweep of β values using GPT-2 (124M parameters). The following values were tested:

$$(0.01,\ 0.03,\ 0.05,\ 0.07,\ 0.1,\ 0.15,\ 0.2,\ 0.25,\ 0.3,\ 0.4,\ 0.5,\ 0.6,\ 0.7,\ 0.8,\ 0.9,\ 1.0).$$

The results reveal strong sensitivity to the choice of β. For values $\beta \geq 0.1$, training rapidly became unstable: the preference margin collapsed and oscillated with large magnitude, often becoming strongly negative (Figure 1). This behavior indicates that GPT-2 probably lacks capacity (not enough parameters) to satisfy strong preference constraints without destabilizing optimization.

Figure 1: Hyperparameter sweep on GPT-2. Large values of β lead to unstable optimization and collapsing preference margins, reflecting limited model capacity.

Main takeaway: for small language models, DPO training remains stable only for very small values of β, suggesting that model capacity plays a critical role in preference optimization.

3.2 Main experiment: target model results (Mistral-7B-instruct)

We then repeated the analysis on a significantly larger and instruction-tuned model, **Mistral-7B-Instruct-v0.3**. Based on the GPT-2 sweep, we compared $\beta = 0.01$ against the commonly used default $\beta = 0.1$ and a larger value $\beta = 0.2$.

In contrast to the GPT-2 results, the larger model exhibited stable training across all tested values and benefited from stronger preference enforcement.

Configuration	Margin	Accuracy	Interpretation
Mistral ($\beta = 0.01$)	-0.14	70.0%	Under-confident preference separation.
Mistral ($\beta = 0.10$)	**+1.13**	**86.0%**	**Best trade-off between stability and separation.**
Mistral ($\beta = 0.20$)	+2.08	84.0%	Strong margins, slight over-confidence.

Table 2: Training outcomes for Mistral-7B. Unlike GPT-2, the larger model benefits from moderate to large β values.

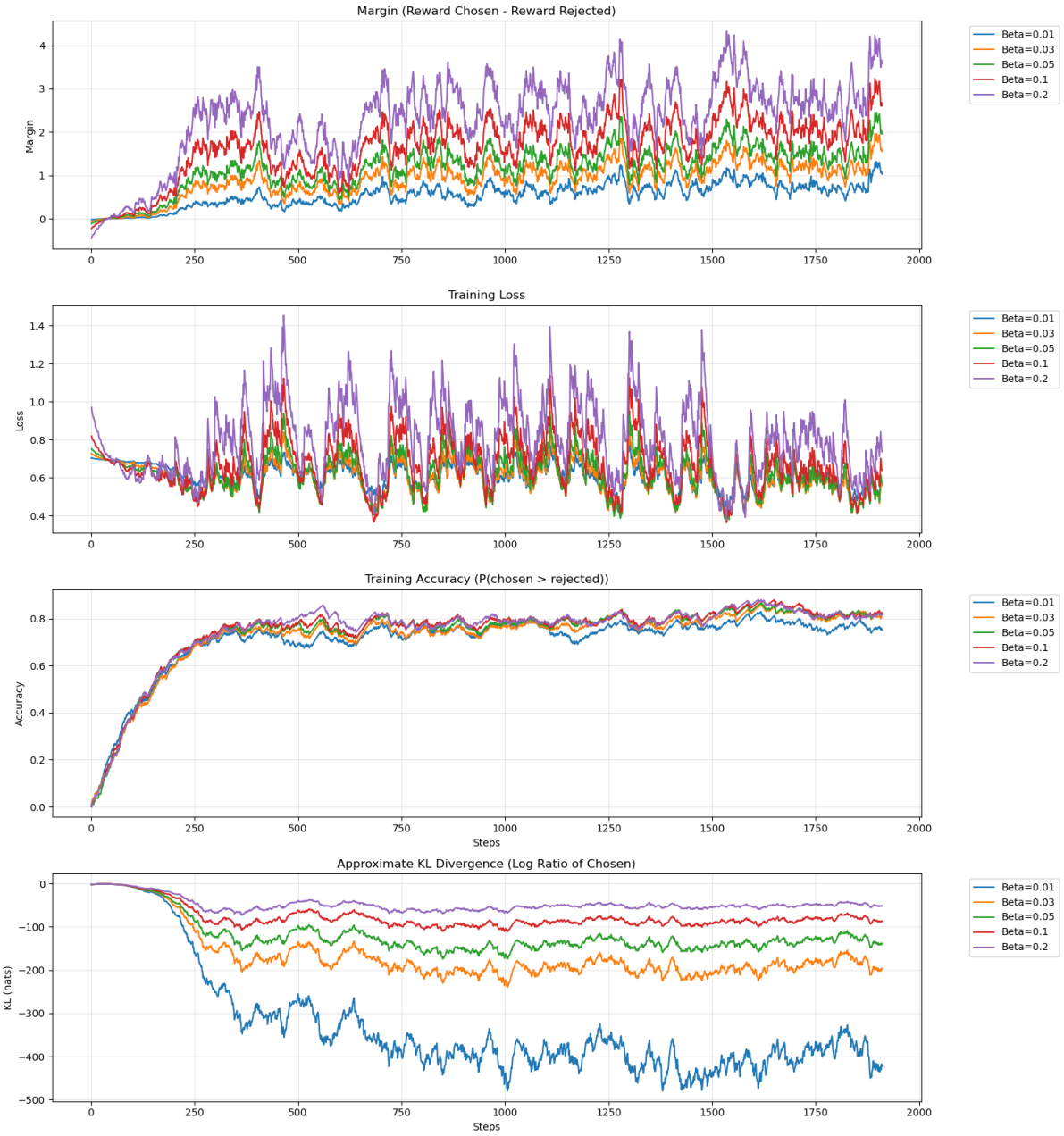

Figure 2: Training dynamics for Mistral-7B across different β values. Larger β yields stronger margins while maintaining stable optimization.

3.3 Analysis of training dynamics

Figure 2 provides a detailed view of the optimization process of DPO training for Mistral-7B across three values of the temperature parameter β. In this subsection we will try to conduct an analysis with the insights of the formulas of the DPO objective.

High β (0.2) As seen in the "Margin" plot (top), $\beta = 0.2$ (purple) achieves a high reward margin almost immediately. Because β amplifies the signal, the model satisfies the classification objective early in training. Once the sigmoid term saturates ($\sigma \approx 1$), the gradient approaches zero. Our explanation is that the model stops learning. We observe the KL divergence flattens out at roughly -50 nats. We state that the model stays close to the reference not because of a constraint, but because of "vanishing gradient".

Low β (0.01) For $\beta = 0.01$ (in blue), the effective margin is heavily compressed. The model struggles to push the sigmoid output close to 1. Consequently, the loss function never saturates, and the optimizer applies a persistent, constant gradient update throughout the entire training run. As a consequence, we observe an unbounded drift, the KL divergence dives to -450 nats, an order of magnitude larger than the high β case. Paradoxically, the "weakest" preference parameter resulted in the largest deviation from the reference model probably because the optimization never reached a stopping condition.

Balanced β (0.1). The standard value $\beta = 0.1$ (red) strikes a balance. It provides enough signal amplification to achieve high accuracy (86%) and meaningful margins, but avoids immediate saturation. This allows the model to learn robust features without drifting endlessly into the high-KL regime observed with lower β.

Takeaway: Hyperparameter tuning in DPO is a search for the saturation sweet spot:a value of β high enough to signal clear preferences, but low enough to maintain a gradient signal throughout training. For 7B-scale models, this optimum appears robustly around $\beta = 0.1$, which is consistent with the default value used in [2].

4 Evaluation of the quality of our aligned model

To rigorously assess the aligned model, we moved beyond supervised learning training metrics (loss/accuracy) to generation quality.

4.1 LLM-as-a-Judge

Evaluating generative models is challenging due to the lack of ground truth. We implemented an "LLM-as-a-Judge" pipeline:

1. **Generation**: We sampled responses from both the SFT Baseline ('Mistral-7B-Instruct') and our DPO Candidate ('Mistral-7B-DPO-Beta0.1') on the held-out test set (200 data points)
2. **Judge**: We employed **Meta-Llama-3-8B-Instruct** as a judge.
3. **Evaluation protocol**: The judge is presented with the prompt and two anonymized responses. To ensure consistency and reproducibility, we enforce a **deterministic generation** setting for the judge ($T = 0$, `do_sample=False`), effectively treating the evaluation as a classification task rather than open-ended generation.

As a remark, as stated in [5], LLM-as-judge are subject to biases. More precisely, they tend to favor first option (response "A") when presented multiple choices questions. To mitigate this, we implemented a randomized counterbalancing strategy. For each evaluation pair, we flip a fair coin; in 50% of cases, the DPO model's response is presented as "Response A", and in the other 50% as "Response B". The final verdict is mapped back to the true model identity during post-processing.

To retrieve the preferred response from the judge's textual output, we implemented the following simple heuristic:

```
if "A" in judgment and "B" not in judgment:
    winner = "A"
elif "B" in judgment and "A" not in judgment:
    winner = "B"
else:
    winner = "Tie"
```

This logic proved robust in practice, as the instructional prompts explicitly instructed the model to output a single character. Combined with the deterministic decoding, this resulted in zero failures.

4.2 Win-rate analysis

The final metric is the win-rate of the DPO model against the SFT baseline:

$$\text{Win Rate} = \frac{\text{Wins}_{DPO}}{\text{Wins}_{DPO} + \text{Wins}_{SFT} + \text{Ties}}$$

The final evaluation was conducted on a held-out set of 200 prompts from ultrafeedback_test_prefs. The DPO model ($\beta = 0.1$) was compared against the SFT baseline using Llama-3-8B-Instruct as the judge.

Outcome	**Count**
DPO Wins	117
SFT Wins	82
Ties	1
Win Rate	**58.5%**

Table 3: Win-rate analysis of DPO vs SFT on held-out test set.

The DPO model demonstrates a clear improvement over the SFT baseline, winning in nearly 60% of cases. This confirms that even with a simple DPO implementation and standard hyperparameters ($\beta = 0.1$), we can significantly improve model alignment.

5 Limitations and discussion

While our results confirm the effectiveness of DPO, several limitations persist:

- **Scale discrepancy**: As demonstrated, small models like GPT-2 are poor predictors for the hyperparameter stability of larger models. This implies that hyperparameter tuning must be performed directly on the target scale, significantly increasing the computational cost of alignment. Still after the swept performed on a large model, we can affirm $\beta = 0.1$ is a reliable and strong value that can be transfered to the alignement of even bigger models, as suggested by the article;
- **Judge bias**: While we mitigated position bias, LLM evaluations remain imperfect proxies for human judgment. The judge model (Llama-3) may exhibit self-preference bias or favor specific stylistic patterns (like length or formatting) over actual helpfulness.
- **Compute constraints**: Due to limited GPU resources, we could not perform the exhaustive grid search typically seen in major labs. Our β exploration was limited to few discrete values, potentially missing a finer optimal point (around 0.1), which could have added value to the standard 0.1 default found in the literature.

Despite some limitations, the strong alignment performance achieved with a standard $\beta = 0.1$ validates the robustness of the DPO framework.

Future work could include:

- **comparison with RLHF**: alignment of the same Mistral-7B-Instruct model through RLHF procedure and comparison with DPO aligned model with the help of an LLM-as-a-judge to explore difference in the final aligned models. Another aspect would;
- **sensitivity of DPO to noisy labels**: a relevant future study would be to poison the labeled dataset $\mathcal{D} = (x_i, y_{w_i}, y_{l_i})$ adding random noise in the preferred response: indeed these preferences are in reality obtained through either paid labelers or directly by users on Chatbot platforms, which inevitably induce some (intentional or not) noise and bias in the quality of the preferred response.

6 Conclusion

In this project, we successfully implemented Direct Preference Optimization (DPO) "by hand" and aligned a modern 7B-parameter language model (Mistral) to human preferences. Our study offers three key takeaways for the open-source community:

1. **DPO simplicity works**: One of the reasons DPO was adopted so quickly in the last 2 years by the community definitely lies in its simplicy over RLHF heavy PPO pipeline. Our ability to implement a full end-to-end alignment that achieves 58% win-rate over SFT baseline indeed shows this important quality of the DPO method;

2. **Scale matters a lot for stability**: We showed that training stability is highly dependent on the model "capacity". While small models (GPT-2) suffer from mode collapse at standard hyperparameters, larger models (Mistral-7B) proved to be more robust;

3. **Robustness of β**: Contrary to initial fears of instability induced by the sweep on GPT-2, we found that the standard value of $\beta = 0.1$ remains a robust default for large models, providing a strong balance between reward maximization and reference regularization, which confirmed the value provided in the literature.

References

[1] `https://github.com/rubenifrah/DPO-implementation-and-finetuning`. 2025.

[2] Rafael Rafailov et al. "Direct Preference Optimization: Your Language Model is Secretly a Reward Model". In: *arXiv* (2024).

[3] Long Ouyang et al. "Training language models to follow instructions with human feedback". In: *arXiv* (2022).

[4] Anthropic. *Human Helpfulness and Harmlessness Dataset (HH).* `https://www.anthropic.com/hh-dataset`. 2023.

[5] Lianmin Zheng et al. "Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena". In: *NeurIPS* (2023).