Progress Report: DPO Alignment, Quantization and RAG Planning

Charles Girardot | 316073 | charles.girardot@epfl.ch
Jacopo Ferro | 299301 | jacopo.ferro@epfl.ch
Yiwei Liu | 369958 | yiw.liu@epfl.ch
Marco Scialanga | 369469 | marco.scialanga@epfl.ch
Pastanoodles

1 Introduction

In this progress report, we will focus on the DPO alignment process that we brought forward for Milestone 2. In Section 2, we will describe the preference dataset that we use for DPO and the MCQ dataset that we plan to use in Milestone 3 for further fine tuning. In Section 3, we will introduce the model that we used and the loss function associated with DPO. In Section 5, we will present how we implemented the DPO training and the results we obtained. Finally, in Sections 4 and 6, we will outline how our model will be polished with the quantization and RAG specializations.

2 Dataset

2.1 DPO Dataset

For DPO Alignment, we need preference data, i.e. a dataset with three columns: *prompt*, *chosen*, *rejected*. First, we took into consideration the dataset that was developed in M1. Upon inspection, we noticed that the various prompts appeared several times (many of them around 12-17 times each). To avoid redundancy and excessive repetitiveness of our data, we filtered the dataset to contain at most 6 instances of each prompt, reducing the number of rows to $10,611$. In addition, we utilized the `datasets` library to load the following preference datasets (for some datasets we only used a fraction of the rows, because we did not want our final training set to be too large due to time constraints): chatml-OpenHermes2.5-dpo-binarized-alpha (sampled 60% of the rows); Neural-DPO (sampled 60% of the rows); orca_math_dpo (a merge of some questions from orca_dpo_pairs and distilabel-math-preference-dpo, sampled $5k$ out of $\approx 15k$ rows); py-dpo-v0.1 (CS questions). Finally, we combined all these datasets into one and ended up with $26,900$ rows. Since the tokenizer associated with our model of choice (`Phi-2`, more on this in Section 3) accepts inputs up to 2048 tokens, we truncated strings that exceeded that amount so that training could proceed smoothly. We shuffled the rows and divided the dataset into train, test, and evaluation with a 0.75, 0.125, 0.125 split. We used train and eval to monitor the training process, and the test set to evaluate the model once training was completed.

2.2 MCQ Dataset

In Milestone 3, we will have to train the model to correctly answer multiple choice questions with a single letter. To improve the model's technical knowledge and make it respond with the proper format, we will need additional data. The datasets we identified so far are: GPQA(Rein et al., 2023) and mmlu(Hendrycks et al., 2021). The GPQA is a challenging dataset of 448 multiple-choice questions written by domain experts contains physics. The mmlu comprises a comprehensive set of 57 multiple-choice tasks spanning various subjects, including college computer science, college mathematics, college physics and machine learning.

3 Model

For the model, we choose `Phi-2`, a pre-trained model with 2.7B parameter from Microsoft, available in the `Transformers` library. Our goal in this phase is to align the model with human and / or AI preferences through Direct Preference Optimization (DPO) (Rafailov et al., 2024). Accordingly, we aim for the model to minimize the following loss function:

$$\mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = -E_{(x,y_w,y_l)\sim\mathcal{D}}\left[\log \sigma\left(r_\theta(x, y_w) - r_\theta(x, y_l)\right)\right]$$

where $r_\theta(x, y) = \beta \log \frac{\pi_\theta(y|x)}{\pi_{\text{ref}}(y|x)}$ is the reward defined by the policy π_θ, representing the model we trained, initialized with the `Phi-2` weights, and reference policy π_{ref}, which is `Phi-2` used as reference model with frozen weights. y_w are the “chosen" responses and y_l the “rejected" ones, while x are the prompts. Thus, $\pi(y|x)$ can be interpreted as the probability of the LLM (or policy) π generating a response y given a prompt x (or taking action y in state x). To better interpret the loss above, it is useful to examine the gradient of the DPO objective with respect to the LLM / policy parameters θ:

$$\nabla_\theta \mathcal{L}_{\text{DPO}}(\pi_\theta;\pi_{\text{ref}}) = -\,\beta E_{(x,y_w,y_l)\sim\mathcal{D}}\big[\sigma(\hat{r}_\theta(x,y_l)-\hat{r}_\theta(x,y_w))\cdot \cdots (\nabla_\theta \log \pi_\theta(y_w|x) - \nabla_\theta \log \pi_\theta(y_l|x))\,\big]$$

Intuitively, this gradient increases the likelihood of the preferred completions y_w and decreases the likelihood of the dispreferred completions y_l (in the second factor), weighted by how much the implicit reward model $\hat{r}_\theta$ misestimates the preferences. In the next section, we will explain how we implemented DPO in practice.

4 Quantization

To fine-tune the model on our limited computational resources, we applied the advanced Parameter-Efficient Fine-Tuning (PEFT) method, Quantized Weight-Decomposed Low-Rank Adaptation (QDoRA), using the `peft` library. We added low rank matrices to be finetuned between our model's attention and dense layers. This technique quantizes decomposed weights and fine-tunes them through Decomposed Rank Adaptation (DoRA, (Liu et al., 2024)), merging Low-Rank Adaptation (LoRA) and Quantized Low-Rank Adaptation (QLoRA) with progressive weight decomposition strategies for optimal efficiency and performance. DoRA breaks down pre-trained weights into magnitude and direction components, then uses LoRA for directional adjustments, significantly reducing trainable parameters. We configured QDoRA with rank $r = 16$, scaling factor $\alpha = 16$, and dropout 0.05, decreasing the trainable parameters to 10,813,440. Here, r represents the rank of the low-rank matrices used to approximate the model's layers, while α adjusts the intensity of DoRA's influence on the base model. To address memory limitations, we implemented nested quantization entirely at 4-bit precision using Nested Factorization 4 (NF4) and opted for a 16-bit data type to expedite fine-tuning.

5 Preliminary Training Results

To train the model with DPO, we use the `DPOTrainer` from Huggingface. We run the DPO for three epochs, with a batch size of 1 and gradient accumulation of 4. For the optimizer, we used AdamW (Loshchilov and Hutter, 2017) with learning rate `5e-5`. To choose all the parameters above, including those of DoRA, we looked at successful DPO runs in the literature (Rafailov et al., 2024), online (Labonne), in our team members' previous experience (Farinella et al.), and by trial and error. Upon inspection, we noticed that the train loss and accuracy (i.e., the frequency at which the model gave higher rewards to "chosen" responses over "ejected" ones) stopped improving after about 1.5 epochs, so we interrupted training. Refer to A for the training plots. For our best model, we chose the best performing checkpoint on our test set using accuracy as a metric (since that is what the model will be evaluated on). This checkpoint was obtained a bit after the first epoch, exactly where the train accuracy peaked (Figure 2), with a test accuracy of ≈ 0.783.

6 RAG

For RAG, we plan to develop a dataset, using `langchain`, with pdfs containing computer science and physics knowledge, since these are the topics our model will be evaluated on. We plan to include textbooks and slides from the listed EPFL courses as well, using the `PyPDFDirectoryLoader`. We will split these documents into chunks of fixed size with the `RecursiveCharacterTextSplitter` with some overlap to avoid losing context. To implement the dataset with proper indexing and allow for fast retrieval, we will use `Chroma` (LangChain). With `Chroma`, we will be able to see the source of each context to easily check if our retriever is working properly, thanks to the `.metadata.get("source")` method. We will modify these chunks by standard text preprocessing techniques such as making all letters lower case, removing stop words, and lemmatize every word (i.e., turn inflected or variant forms of the same lemma into the same word). By doing so, the retriever will find the top-k chunks for the given prompt, using either minimum euclidean distance or maximum cosine similarity between the prompt and the chunks. Once the context is retrieved, we will input it into the prompt with the following template (we assume the question contains the options): Considering this context:" + context + Answer the following question:" + question + “Output one letter only corresponding to the correct answer". This prompt will be passed to our final (after MCQA training) quantized version of `Phi-2`, which will finally output the single letter answer.

References

Elsa Farinella, Robin Faro, and Marco Scialanga. Towards a factual chatbot with rlhf dpo alignment and rl-optimized rag. https://github.com/robinfaro/RL_Final_Project. Accessed: 2024-28-05.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Maxime Labonne. Fine-tune a mistral-7b model with direct preference optimization. https://towardsdatascience.com/fine-tune-a-mistral-7b-model-with-direct-preference-optimization-708042745aac. Accessed: 2024-28-05.

LangChain. Chroma. https://python.langchain.com/v0.1/docs/integrations/vectorstores/chroma/. Accessed: 2024-28-05.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. 2024. Dora: Weight-decomposed low-rank adaptation. *arXiv preprint arXiv:2402.09353*.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. 2023. Gpqa: A graduate-level google-proof qa benchmark.

A Training Plots

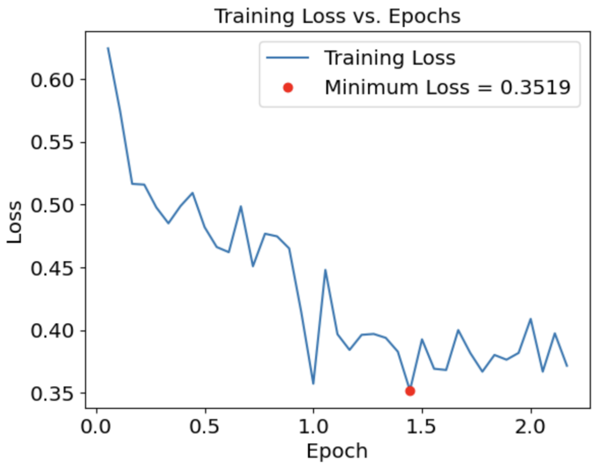

Figure 1: The DPO training loss decreasing and then plateauing.

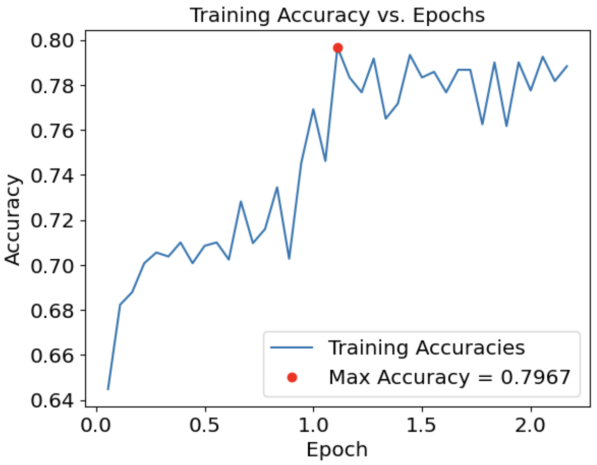

Figure 2: The DPO training accuracy increasing to nearly 80% and then plateauing.