Aligning LLMs with human preferences using DPO

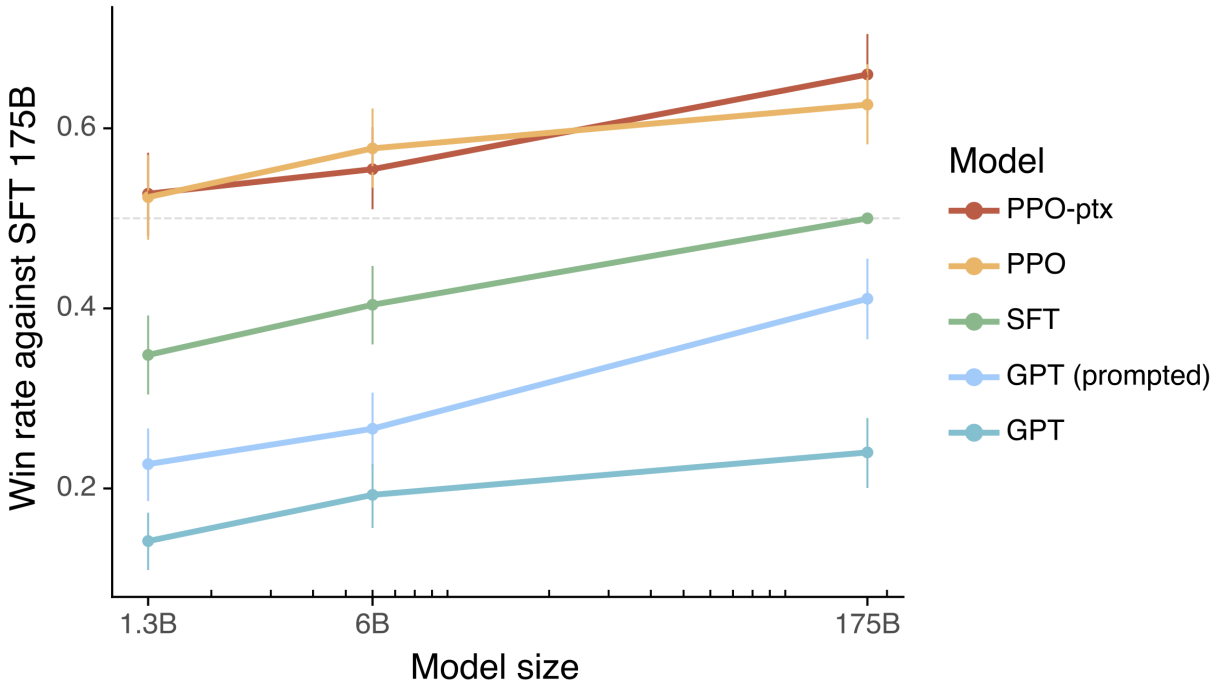
Owner	Andrii Cherniak
Tags	

Summary

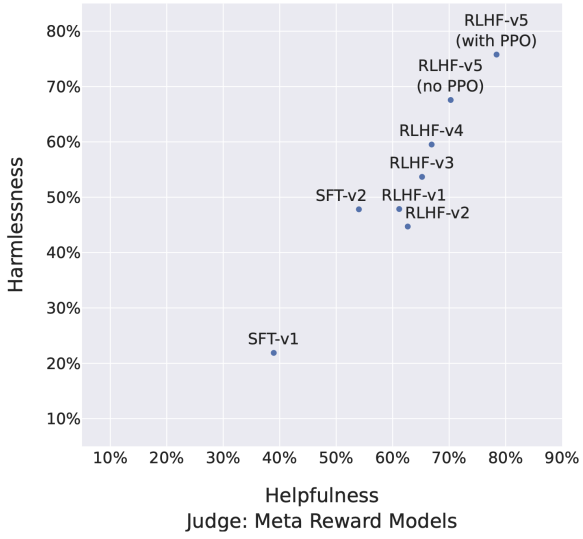
1. fine-tuned https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.2 on 12K preference pairs from https://huggingface.co/datasets/Intel/orca_dpo_pairs using 4bit QLoRA
2. found good starting values for batch size, gradient accumulator and learning rate
3. computational budget: ~ 60 hours on A100 GPU node ~ $240
4. TODO: evaluation planning

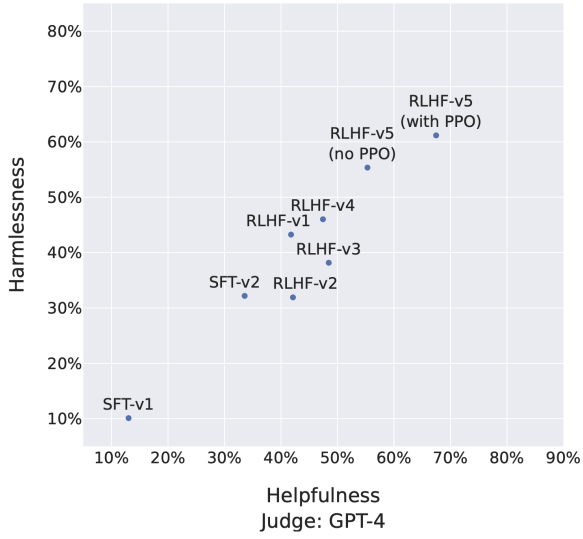
Preference alignment: motivation

We are operating in the medical field with the main principle: **do no harm**. The responses produced by our LLM must be helpful, honest, harmless, and have a tone that would resonate with pregnant women. Borrowing from https://arxiv.org/abs/2203.02155 , outputs from a much smaller LLM (1.3B parameters) may be preferred to the outputs from a much larger LLM (175B parameters) if a small model is tuned on human preference data.



Moreover with alignment we can simultaneously improve helpfulness and harmlessness for LLM https://arxiv.org/abs/2307.09288





LLM choice

The inspiration for LLM, data set choice, and evaluation thoughts comes from https://huggingface.co/Intel/neural-chat-7b-v3-1 the best scoring on LLM leaderboard.

However for the initial model the instruction-fine tuned mistral has been chosen in hopes it will perform better on QA tasks. However later we need to compare base mistral against instruction-fine-tuned version to confirm or disprove the claim.

Dataset consideration

As mentioned in https://huggingface.co/docs/trl/main/en/dpo_trainer DPO requires dataset of with the features **[prompt, chosen, rejected]** where the **prompt** contains the context inputs, **chosen** contains the corresponding chosen responses and **rejected** contains the corresponding negative (rejected) responses.

Since I am trying to re-create pipeline from Intel neural chat, I also use https://huggingface.co/datasets/Intel/orca_dpo_pairs dataset containing 12k examples from **Orca** style dataset **Open-Orca/OpenOrca** with features **[system, question, chosen, rejected]**.

Tokenizer and chat format

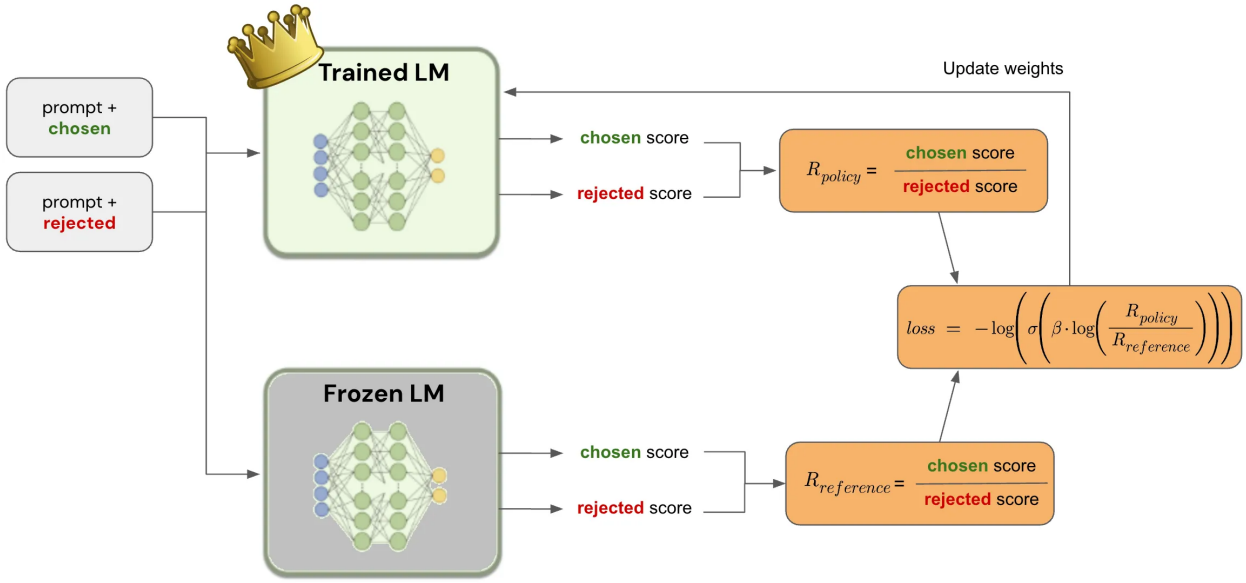
DPO trainer expects dataset with features **[prompt, chosen, rejected]** . https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.2 expects input in https://huggingface.co/docs/transformers/main/chat_templating format. Chat template for mistral specifically does not support {"role": "system" } prompt. One possible solution was inspired by https://github.com/mlabonne/llm-course/blob/main/Fine_tune_a_Mistral_7b_model_with_DPO.ipynb .

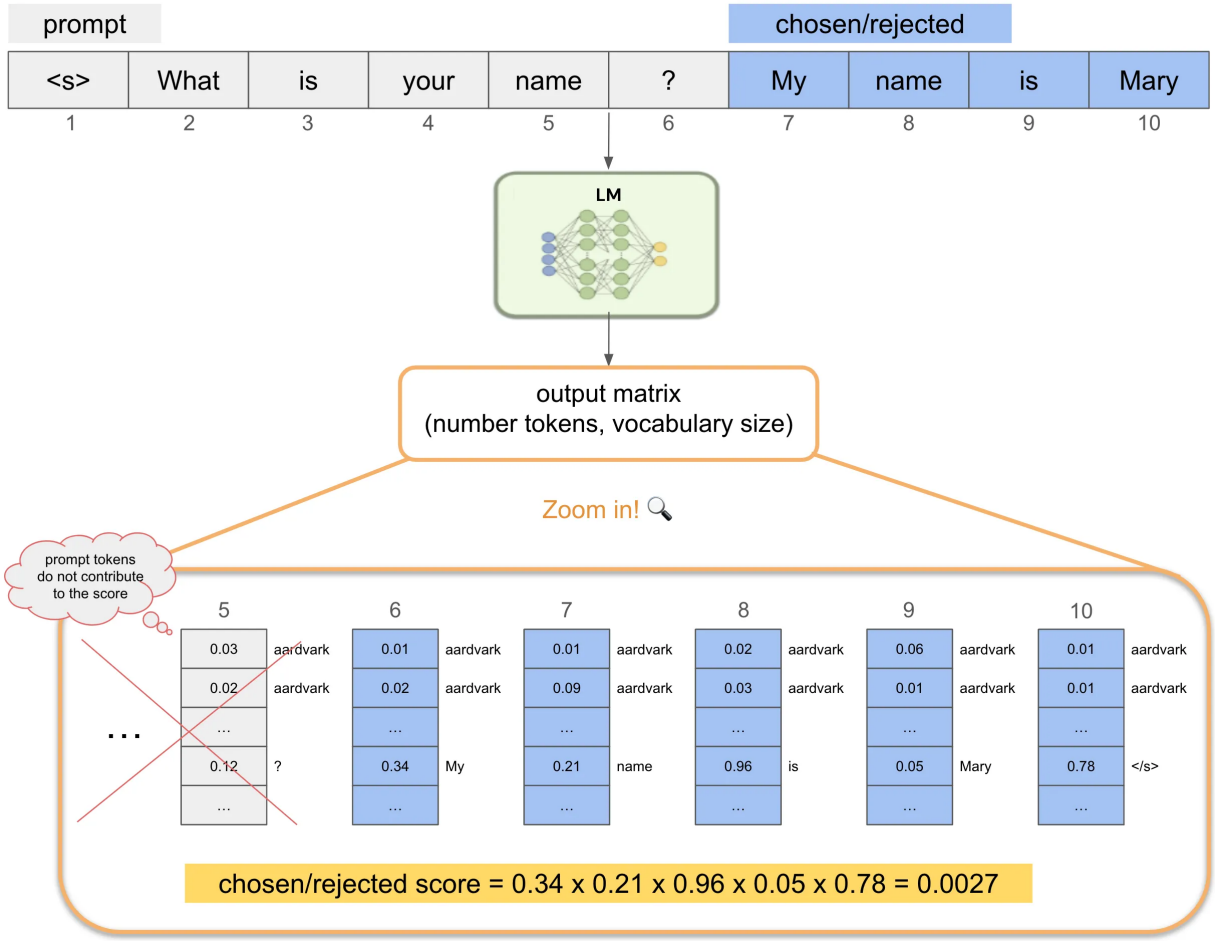
```
chat = [
    {"role": "user", "content": <systm prompt>},
    {"role": "assistant", "content": "Please ask me your questi
    {"role": "user", "content": <question>}]

prompt = tokenizer.apply_chat_template(m, tokenize=False, add_ge
chosen = <chosen response> + tokenizer.eos_token + "\n"
rejected = <rejected response> + tokenizer.eos_token + "\n"
```

where tokenizer takes care of the final formatting. However this is not set in stone and actually needs experimenting.

DPO from the bird-eye view

DPO uses two models: the **trained model (policy model)** and a copy of it called the **reference model (frozen model)**. For each datapoint, the chosen and rejected responses are scored by both the trained and the frozen language model. This score is the product of the probabilities associated with the desired response token for each step.





With the chosen and rejected responses scored, we can calculate the ratio between the scores given by the trained language model, ***R**policy*, and the ones given by the frozen language model, ***R**reference*. These ratios are then used to calculate the final loss that is used to modify the model weights in the gradient descent update.

During training, the goal is to make sure the trained model outputs higher probabilities for preferred answers than the reference model. We also want it to output lower probabilities for rejected answers. It means we're penalizing the LLM for bad answers and rewarding it for good ones.
https://towardsdatascience.com/fine-tune-a-mistral-7b-model-with-direct-preference-optimization-708042745aac

Memory usage and QLoRA

A single A100 GPU has 80GB of memory. For DPO we need to have trained and frozen models loaded, or trained and frozen adapters + model in memory.

Without quantization two copies of mistral do not fit on GPU. A work-around was to load two 4bit quantized models and use QLoRA . QLoRA offers 33% memory savings at the cost of a 39% increase in runtime https://magazine.sebastianraschka.com/p/practical-tips-for-finetuning-llms .

Again this is not the only way to perform DPO and def least memory-efficient but this is a starting point.

Batch size	min GPU RAM, GB	max GPU RAM, GB
2	25	30
8	50	65
16	78	terminated without waiting, was too close to 80

Batch size, gradient accumulation, and learning rate

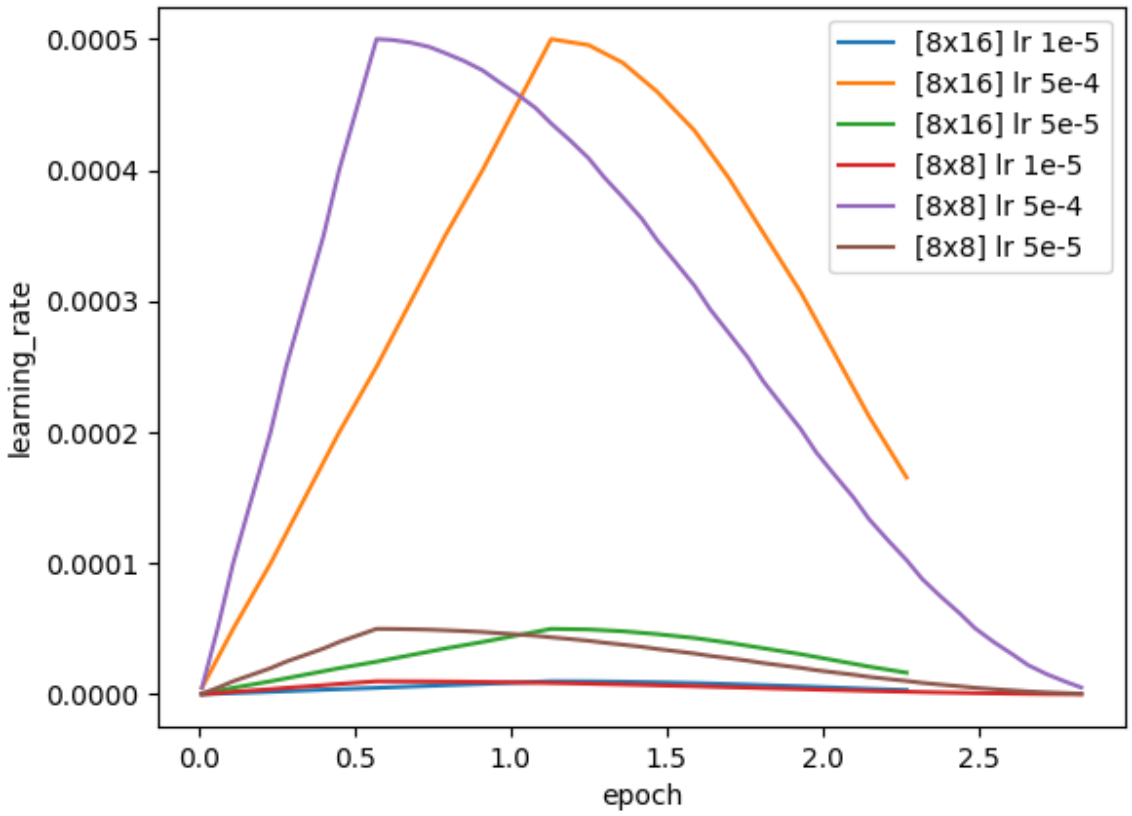
Using https://medium.com/intel-analytics-software/the-practice-of-supervised-finetuning-and-direct-preference-optimization-on-habana-gaudi2-a1197d8a3cd3 as a reference, batch size and gradient accumulation were set to 8 which effectively gives a batch size of 64, and learning rate 5e-4. Mistral utilizes https://arxiv.org/abs/1910.07467 and in their experiments the batch size was 80. Thus in theory we do not want to go low on the batch size. At the same time there is a relation between mini batch and learning rate https://arxiv.org/abs/1706.02677 : the higher the batch size the higher learning rate should be. But how sensitive are we to those parameters?

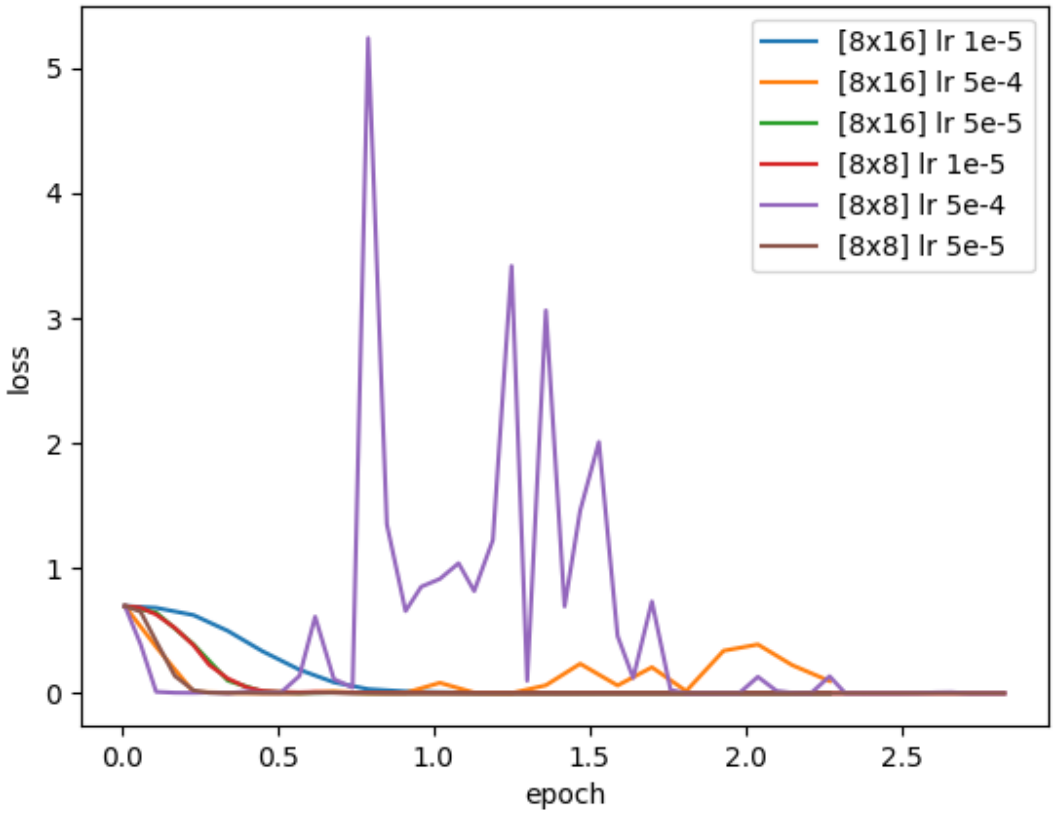
Absolute max batch size with QLoRA was 16 but it was getting very close to the max GPU utilization. Thus instead I kept batch size fixed = 8 and instead varied batch accumulation size and learning rate.

batch size	gradient accumulation	learning rate
8	8	1e-5
8	8	5e-5
8	8	5e-4
8	16	1e-5

8	16	5e-5
8	16	5e-4

LR scheduler was set to “cosine” and optimizer paged_adamw_32bit . Trining consisted of 3 epochs, and takes ~ 8-9 hours.





💡 As we can see, lr=5e-4 is probably too high when effective batch size = 64 . However the situation improves when effective batch size = 128. together, it seems that lr=5e-4 is too high. Let us analyze the rewards under more modest lr = [1e-5, 5e-5]

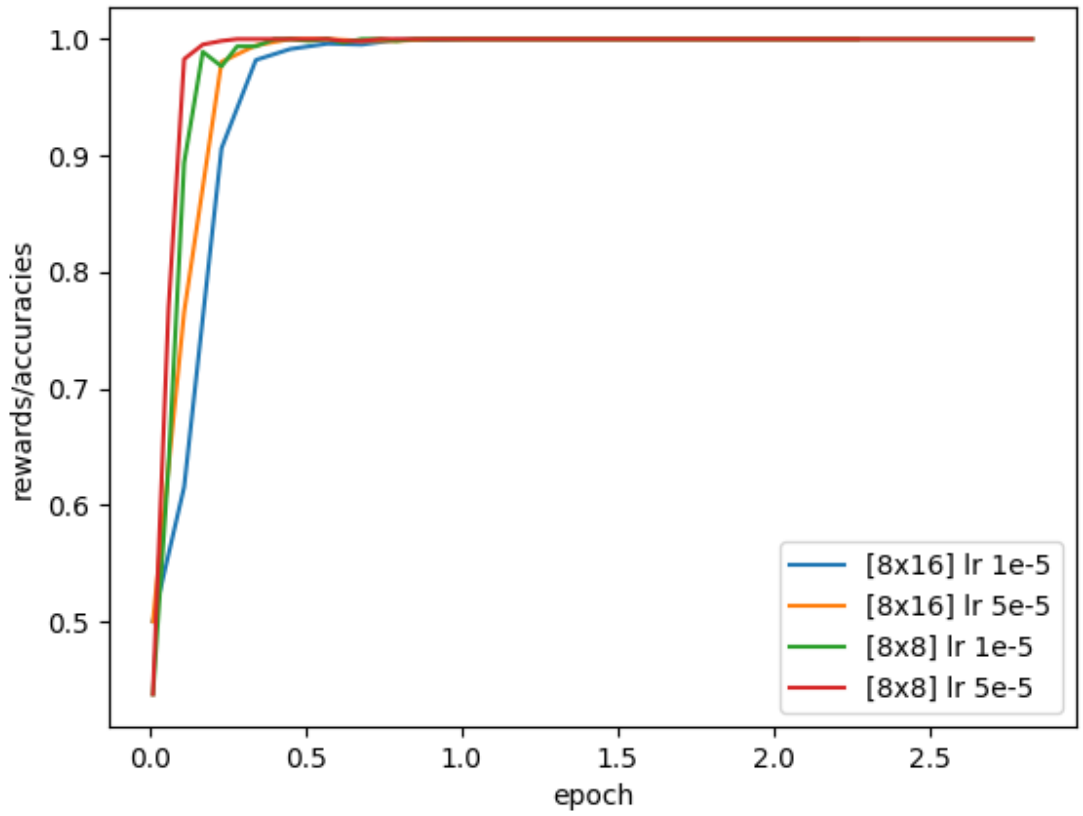
Rewards, accuracy etc

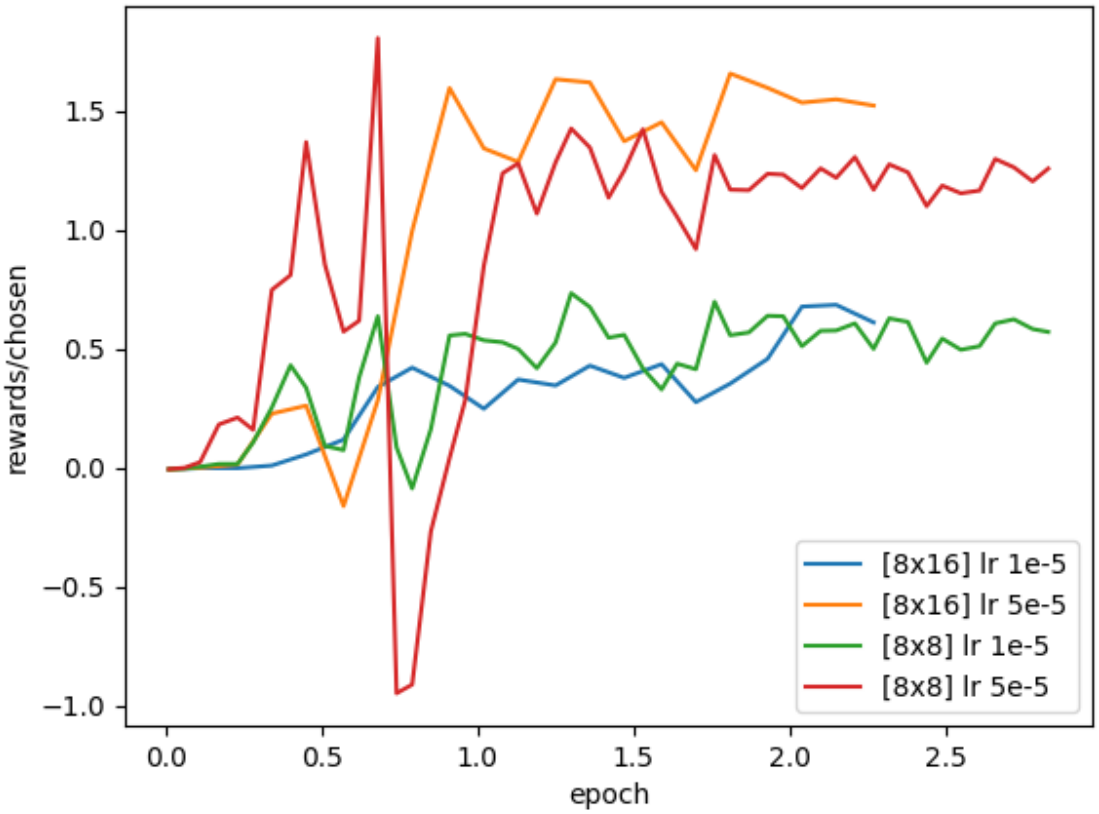
From https://github.com/huggingface/blog/blob/main/dpo-trl.md , DPO records

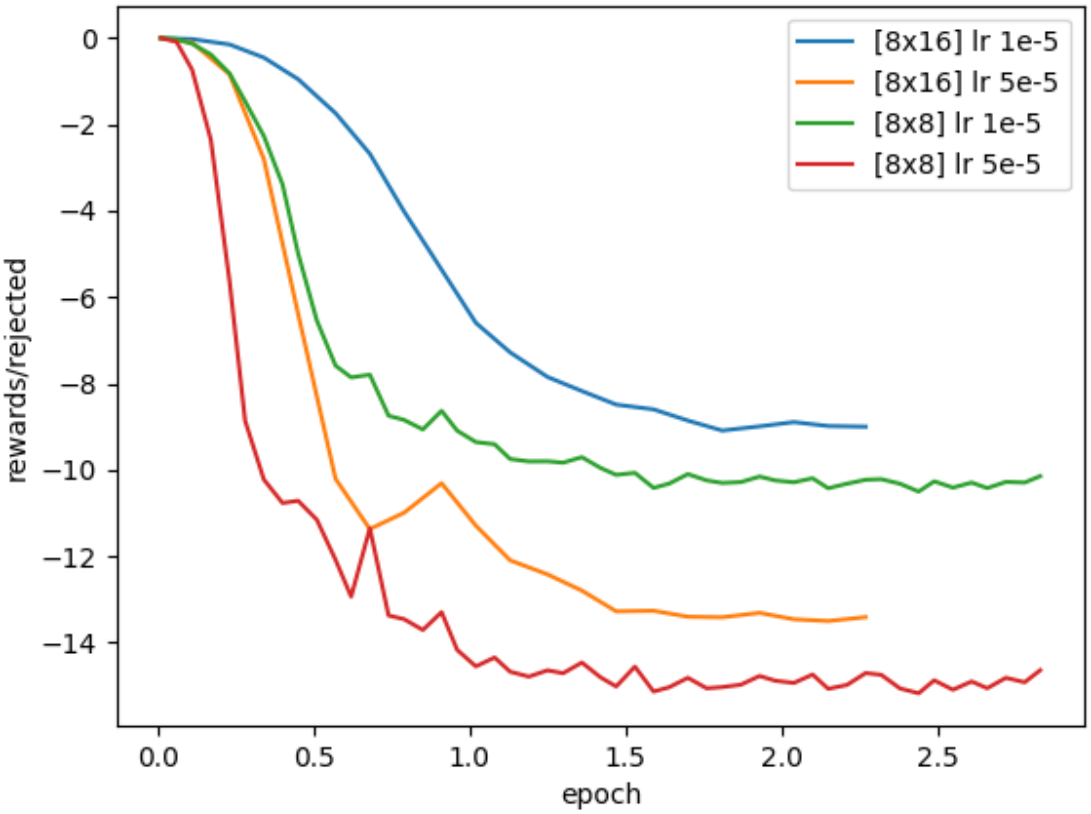
- `rewards/chosen`: the mean difference between the log probabilities of the policy model and the reference model for the chosen responses scaled by `beta`
- `rewards/rejected`: the mean difference between the log probabilities of the policy model and the reference model for the rejected responses scaled by `beta`

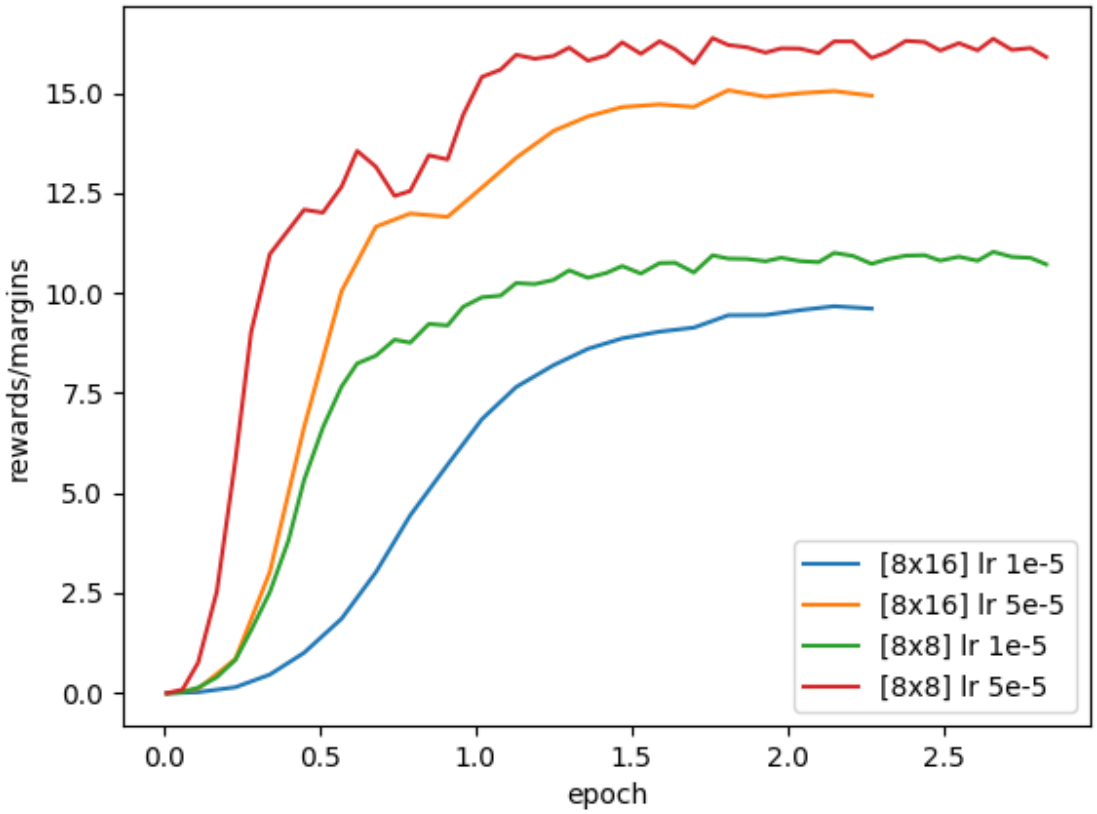
- `rewards/accuracies`: mean of how often the chosen rewards are > than the corresponding rejected rewards
- `rewards/margins`: the mean difference between the chosen and corresponding rejected rewards.

> 💡 During training we want the margins to increase and the accuracies to go to 1.0, or in other words the chosen reward to be higher than the rejected reward (or the margin bigger than zero).




rewards/chosen
epoch
1.5
1.0
0.5
0.0
−0.5
−1.0
0.0
0.5
1.0
1.5
2.0
2.5
[8x16] lr 1e-5
[8x16] lr 5e-5
[8x8] lr 1e-5
[8x8] lr 5e-5


[8x16] lr 1e-5
[8x16] lr 5e-5
[8x8] lr 1e-5
[8x8] lr 5e-5
rewards/rejected
epoch
0
−2
−4
−6
−8
−10
−12
−14
0.0
0.5
1.0
1.5
2.0
2.5



Coding: next steps

1. incorporate training / eval logging with Neptune
2. Switch 4bit → 8 bit quantization for QLoRA - can we fit in GPU ? By how much evaluation scores improve?
3. attempt to use one instance of LLM + 2 QLoRA adapters or 1 QLoRA adapter under two different names for training/reference modeling . Measure GPU usage
4. Can we skip quantization completely and use LoRA? Depends on # 3

Evaluation: next steps

1. Standard datasets.
    a. MedQA https://huggingface.co/datasets/bigbio/med_qa,

   b. PubmedQA https://huggingface.co/datasets/pubmed_qa
   c. MMLU clinical topics https://huggingface.co/datasets/cais/mmlu
   d. MedMQA https://huggingface.co/datasets/medmcqa
   e. MedicationQA https://github.com/abachaa/Medication_QA_MedInfo2019
2. use question answering / dialog completion on human preference datasets, like
   a. https://huggingface.co/datasets/Intel/orca_dpo_pairs , (test split)
   b. https://huggingface.co/datasets/Anthropic/hh-rlhf,
   c. https://github.com/glgh/awesome-llm-human-preference-datasets.
   d. compute log probabilities for chosen and rejected responses and compute how often logprob(chosen) > logprob(rejected) (desired behavior)
   e. generate answers to the questions and scored on empathy, toxicity, bias, polarity, hurtfulness using https://www.perspectiveapi.com or https://huggingface.co/spaces/evaluate-measurement/toxicity
   f. generated answers can be assessed for text quality with https://github.com/HLasse/TextDescriptives
3. Red teaming - can we provoke LLM to mis-behave and by how much this mis-behaving can be reduced with DPO
   a. https://www.anthropic.com/news/red-teaming-language-models-to-reduce-harms-methods-scaling-behaviors-and-lessons-learned
   b. https://huggingface.co/blog/red-teaming
   c. https://huggingface.co/datasets/allenai/real-toxicity-prompts
4. Take questions from doula certification doc, generate answers with untuned and tuned LLM , use LLM-as-a-judge to score the answers
5. take a sample of documents we downloaded (ACOG, AAFP, CC), generate a set of questions from those documents , and evaluate on accuracy against that document, as well as empathy, toxicity, bias, polarity, hurtfulness
6. Investigate how to incorporate RLAIF https://arxiv.org/abs/2212.08073, https://arxiv.org/pdf/2309.00267.pdf because alignment data for the medical

field will be hard to find and will be limited at first by our annotators

How does DPO affect text generation

LLMAJ evaluation

Win Rate evaluation