RoBERTo final report

Hugginface repo: DavidLacour/HyumnoZephyrmcqa2noquantV2

Margot Malis | 311645 | margot.malis@epfl.ch
Lacour David | 275503 | david.lacour@epfl.ch
Hyunmo Kang | 375619 | hyunmo.kang@epfl.ch
Jiewei Li | 384742 | jiewei.li@unil.ch
RoBERTo

Abstract

In this project, our objective was to train an AI tutor specialized in the course content at EPFL. Our goal is to assist students in their learning by providing a language model capable of generating answers to questions in science, technology, computer sciences, engineering, and mathematics. We aim for our model to offer not just answers, but also well-defined, step-by-step explanations to ensure that even complete beginners can understand the material clearly. In order to do this we split the task in 3 step. First we collected data in order to train our languages models. First, we collected data to train our language models. Next, we trained the model using Direct Policy Optimization (DPO). Additionally, we specialized it to provide highly accurate single-letter answers for multiple-choice questions. Finally, we expanded our model's capabilities to retrieve information from documents and also making it more compact.

1 Introduction

It's extremely difficult for an AI language model to provide correct answer for math, programming and science question. Even, ChatGPT4.o from OpenAI still does mistakes and is unable to answer correctly to some math, coding questions or in other fields. Why is it so hard to create a good large language model teaching assistant?

- **Complex Calculations**: Math often requires performing complex calculations accurately. Small errors in computation can lead to incorrect answers, and language models, unlike traditional computational software, aren't inherently designed to handle intricate arithmetic with perfect precision.
- **Abstract Concepts**: Many mathematical concepts are abstract and require a deep understanding of underlying principles. For instance, solving differential equations or understanding topological spaces involves more than just applying formulas—it requires comprehension of theoretical concepts that are difficult to encode into a model.
- **Debugging and Logic**: Writing correct code often involves debugging and understanding complex logic. Models may generate code that looks correct but fails to execute properly due to logical errors, missing edge cases, or incorrect assumptions about input and output. Context Understanding: Programming questions often depend on understanding the specific context of the problem, such as the requirements and constraints. Also, the AI is not updated each time a library changed. Language models can struggle to keep track of context over longer conversations or intricate problem statements.
- **Interdisciplinary Knowledge**: Science encompasses a wide range of disciplines, each with its own set of principles, terminologies, and methodologies. A model needs to be proficient across multiple areas, from physics and chemistry to biology and earth sciences, which is a vast amount of information to encode and retrieve accurately.
- **Training Data Limitations**: The training data for models like ChatGPT-4 consists of vast amounts of text from the internet. While this includes a lot of useful information, it also contains inaccuracies, ambiguities, and a lack of detailed, context-specific explanations necessary for solving complex problems. Humans make mistakes and those mistakes can me reproduced by language models.
- **Inference and Reasoning**: Language models are primarily designed to predict the next word in a sequence rather than to perform logical inference or reason through multi-step problems. This makes it difficult for them to handle problems that require deep reasoning and logical structuring. However very large model can sometimes show emergent behavior. (Wei et al., 2022)

There is a critical necessity for competent teaching assistants in STEM programs. These are complex and challenging field, requiring students to grasp intricate concepts and apply them practically. Effective teaching assistants play a pivotal role in facilitating this learning process. They provide additional support, clarify difficult topics, and offer personalized guidance, which is essential for students to succeed in such a demanding discipline. Ensuring the availability of skilled teaching assistants can significantly enhance the educational experience and outcomes for engineering students. Teacher assistant can sometimes spends hours to help a students debugs or solve hard multi-layered physics problem. A large language model that excels at helping students could alleviate some of the pressure and workload on current teaching assistants and help when there is a shortage of assistants.

2 Related Work

- **Large Language models as Chatbots:**. Modern language models like LLaMA (Touvron et al., 2023) or mistral (Jiang et al., 2023) are trained on vast amounts of diverse text data, enabling them to grasp intricate grammatical structures and semantics. Their impressive generative abilities allow these models to engage in human-like conversations, displaying extensive knowledge on a variety of topics.

- **LLM chatbot for education:** The emergence of sophisticated large language models presents exciting prospects for students and educators. Many students now utilize these tools daily, and teachers are adapting their teaching methods to integrate them effectively. However, AI tools cannot completely replace human teachers in their interactions with students. Additionally, the use of AI brings potential risks, such as ethical challenges arising from a lack of transparency or misuse. (Jeon and Lee, 2023) (Kasneci et al., 2023) It is crucial to educate students about these risks, ensuring they do not become overly reliant on AI and maintain critical thinking. here is a concern that the overuse of large language models (LLMs) could potentially hinder students' development of critical reasoning skills. While LLMs can provide quick and accurate answers, relying too heavily on them may discourage students from engaging deeply with the material, thinking critically, and developing their own problem-solving abilities. It is crucial to balance the use of such technology with traditional learning methods that promote active thinking and reasoning to ensure that students continue to cultivate essential cognitive skills. Hovewer, this problem already exist when the student have accesss to exercise solutions. (Grimaldi et al., 2019) PLOS

- **Fine tuning pretrained chatbots:** Self-supervised language models of increasing scale can handle some tasks without prior examples (zero-shot) or with minimal examples (few-shot prompts). (Radford et al., 2019) (Brown et al., 2020)However, their effectiveness on downstream tasks and alignment with user intentions can be greatly enhanced by fine-tuning on datasets containing instructions and human-generated completions. (Mishra et al., 2022)This process, known as 'instruction-tuning,' helps language models generalize to new instructions outside their training set, making them more useful. (Chung et al., 2022) Despite the success of instruction tuning, it's often easier to gather relative human judgments of response quality than to collect expert demonstrations. Consequently, subsequent research has fine-tuned language models using datasets of human preferences, which has improved performance in areas such as translation, summarization, storytelling, and instruction-following. (Kreutzer et al., 2018) (Stiennon et al., 2022) (Ouyang et al., 2022)

- **RL method for finetuning LLM:** These approaches typically begin by optimizing a neural network reward function that aligns with the dataset of preferences using models like the Bradley-Terry model. The language model is then fine-tuned to maximize this reward through reinforcement learning algorithms, such as REINFORCE (Wu and Hu, 2018), proximal policy optimization (PPO), or their variants. A related research direction involves using language models fine-tuned with human feedback to generate additional synthetic preference data for specific attributes like safety, guided by weak human supervision in the form of text rubrics for annotations. This represents a convergence of research on training language models with reinforcement learning for various objectives and general methods for learning from human preferences. Despite the appeal of using relative human preferences, fine-tuning large language models with reinforcement learning poses significant practical challenges. This work proposes a theoretically sound method for optimizing relative preferences without relying on reinforcement learning.

- **Direct preference optimization:** Outside the realm of language, learning policies from preferences has been explored in both bandit and reinforcement learning contexts. Various methods have been proposed, such as contextual bandit learning, which uses preferences or rankings of actions instead of rewards. This is known as a contextual dueling bandit (CDB) (Yue et al., 2012). In scenarios lacking absolute rewards, CDB theoretical analysis replaces the concept of an optimal policy with a von Neumann winner, a policy that wins against any other policy at least 50% of the time. However, in the CDB context, preference labels are provided online, whereas learning from human preferences typically involves a fixed batch of offline preference-annotated action pairs. Similarly, preference-based reinforcement learning (PbRL) (Busa-Fekete et al., 2014) learns from binary preferences derived from an unknown 'scoring' function instead of rewards. Various algorithms for PbRL exist, including those that can reuse off-policy preference data, but they generally involve first estimating the latent scoring function (the reward model) and then optimizing it. Instead, we present a single-stage policy learning approach that directly optimizes a policy to meet the preferences, which is called Direct prefrence based optimization(DPO)(Rafailov et al., 2023).

3 Approach

3.1 recap

In order to create our teaching assistant language model, we divided the task into three steps. First, we collected data to train our language models. Next, we trained the model using Direct Preference Optimization (DPO). Additionally, we specialized it to provide highly accurate single-letter answers for multiple-choice questions. Finally, we expanded our model's capabilities to retrieve information from documents as well as making it more compact. In this section, we introduce our model as well as our training pipeline.

3.2 Model : unsloth/zephyr-sft

This model is a fine-tuned version of mistralai/Mistral-7B-v0.1 on the HuggingFaceH4/ultrachat_200k dataset. The Mistral-7B-v0.1 Large Language Model (LLM) is a pretrained generative text model with 7 billion parameters. Mistral-7B-v0.1 outperforms Llama 2 13B on all benchmarks we tested. For full details of this model please read the paper (Jiang et al., 2023) and the release blog post.

We chosed this model because it is decent can be trained faster using unsloth pipeline. Unsloth. Unsloth make faster by manually deriving all compute heavy maths steps and handwriting GPU kernels, unsloth can magically make training faster without any hardware changes. More information on github

3.3 DPO Training

For DPO training we use the DPO trainer from trl, more on DPOTrainerGithub

We also use Parameter-Efficient Fine-Tuning (PEFT) Instead of fine-tuning all the parameters of a large pre-trained model, PEFT methods focus on adjusting a small subset of parameters. This significantly reduces the computational resources and time required for fine-tuning. By updating fewer parameters, the memory footprint is reduced, making it feasible to fine-tune large models on hardware with limited memory. It uses adapters, small trainable modules (adapters) between layers of the pre-trained model. During fine-tuning, only these adapter modules are trained, while the rest of the model parameters remain frozen. It also use linear algebra technics such as Low-Rank Adaptation (LoRA) which decomposes the weight updates into low-rank matrices to reduce the number of parameters that need to be updated(Hu et al., 2021).

In essence, Direct Preference Optimization aims to make decisions or predictions that align closely with what is actually preferred, using direct data on those preferences to guide the process. The data include a chosen text that represent the preferred text over a rejected text. Driven by the complexities of employing reinforcement learning algorithms on extensive tasks, like fine-tuning language models, DPO is a straightforward method for policy optimization based directly on preferences. Unlike traditional RLHF techniques that first learn a reward model and then optimize it using reinforcement learning, our approach utilizes a specific parameterization of the reward model. This unique parameterization allows for the extraction of the optimal policy in a closed form, eliminating the need for an RL training loop. (Rafailov et al., 2023)

To gain a mechanistic understanding of DPO, it is beneficial to examine the gradient of the loss function L_{DPO}. The gradient with respect to the parameters θ can be expressed as:

$$\begin{aligned}&\nabla_\theta L_{\text{DPO}}(\pi_\theta;\pi_{\text{ref}})\\ &\quad= -\beta\mathbb{E}_{(x,y_w,y_l)\sim D}\left[\sigma(\hat{r}_\theta(x,y_l)-\hat{r}_\theta(x,y_w))\right]\\ &\quad= \left(\underbrace{\nabla_\theta \log \pi(y_w|x)}_{\text{increase likelihood of } y_w} - \underbrace{\nabla_\theta \log \pi(y_l|x)}_{\text{decrease likelihood of } y_l}\right)\Bigg]\end{aligned}$$

where

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

represents the reward implicitly defined by the language model π_θ and the reference model π_{ref} . Intuitively, the gradient of the loss function L_{DPO} serves to increase the likelihood of the preferred completions y_w while decreasing the likelihood of the dispreferred completions y_l. Notably, the examples are weighted by the extent to which the implicit reward model $\hat{r}_\theta$ rates the dispreferred completions higher, scaled by β. This effectively measures how incorrectly the implicit reward model orders the completions, taking into account the strength of the KL constraint. This weighting is crucial, as a simple version of this method without the weighting coefficient can lead to the degeneration of the language model.

DPO circumvents the need to explicitly fit a reward function and avoids the complexity of performing reinforcement learning to learn the policy by employing a single maximum likelihood objective. The optimization objective in Equation 5 is akin to a Bradley-Terry model, with a reward parameterization defined as

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

This allows us to optimize our parametric model π_θ in a manner equivalent to the reward model optimization in Equation 2, facilitated by a change of variables, which represents the reward implicitly defined by the language model π_θ and the reference model π_{ref} (discussed further in Section 5). Intuitively, the gradient of the loss function L_{DPO} serves to increase the likelihood of the preferred completions y_w while decreasing the likelihood of the dispreferred completions y_l. Notably, the examples are weighted by the extent to which the implicit reward model $\hat{r}_\theta$ rates the dispreferred completions higher, scaled by β. This effectively measures how incorrectly the implicit reward model orders the completions, taking into account the strength of the KL constraint. Our experiments indicate the significance of this weighting, as a simple version of this method without the weighting coefficient can lead to the degeneration of the language model. (Rafailov et al., 2023) (Tunstall et al., 2023)

Our goal is for the model to deliver not just accurate responses, but also clear, step-by-step explanations. This is why, in our datasets, in the "chosen" section, we provide a thorough answer that is detailed, easy to comprehend, and educational. In the "rejected" section, we include answers that might be correct or somewhat incorrect but are missing adequate explanatory details. For MCQA data, we put golden answer in "chosen" section, where as wrong answer in "rejected" section.

Training is done mainly on google colab, we trained other models like distillgpt2 on the scitas cluster but they were less performants.

3.4 Generation

3.4.1 Generation to the learning students

For generating an answer to the learning students we use the generating function with standard parameters:

- `input_ids`
  The input token IDs.
- `max_new_token`
  The maximum new token of the generated text.
- `num_return_sequences = 1`
  The number of generated sequences to return. Here, it is set to 1.
- `no_repeat_ngram_size = 2`
  This parameter ensures that no 2-grams (pairs of tokens) are repeated in the generated text, promoting diversity.
- `early_stopping = True`
  This parameter stops the generation early if an end-of-sequence token is generated before reaching the `max_new_token`.

The function then decodes and returns the generated text as a string. The no repeat n-grams is crucial otherwise the model will keep repeating the same thing. We could improve this by using n-beans (checkking more that one answers in parrallel) and tweaking the temperature. ("temperature" is a hyperparameter that controls the randomness of predictions made by the model.)

3.4.2 Generation for mcqa

Initially, we considered generating the MCQA answer by outputting the token (A, B, C, D) with the highest probability based on token ID on last token of generated output. However, we ultimately decided to use the generated text instead, as it occasionally yielded higher accuracy during evaluation. This could be because it is predicting token with space like "A " instead of "A" or other reasons. We generate a response and we use a regular expression to search for the pattern "Answer: " followed by a letter (A, B, C, or D) in the generated text.

3.4.3 Generation for mcqa, using RAG

To implement RAG(Retrieval Augmented Generation), we first gathered database relevent for STEM subjects, as in below. Then we devided this database into chunck of size 1000, and embedded using huggingface embedding with model name'sentence-transformers/all-mpnet-base-v2'. Then we load them into FAISS vector database. Then we queried the database to retrieve relevant top-4 relevant chunks, and concatanate context in front of questions. Then we generated mcqa answers as above.

3.5 Quantization

To efficiently manage the computational and memory requirements of the unsloth/zephyr-sft model, which is approximately 30GB in size (fullsize), we employed 4-bit quantization. This technique involves representing the model's weights using 4 bits, significantly reducing the model's memory footprint. By leveraging Hugging Face's and AutoGPTQ support for 4-bit quantization, we achieved a more manageable model size without substantial loss in performance, facilitating deployment on resource-constrained hardware. GPTQ is way to quantize already finetuned moel. (Frantar et al., 2022) . This is the solution we went for, even though many quantization methods are available, each with its unique advantages and disadvantages. (Dettmers et al., 2022) (Xiao et al., 2022) (Chavan et al., 2024) Additionally, 2-bit methods using values like -1, 0, and 1 are also emerging. (Ma et al., 2024)

4 Experiments

4.1 Data

4.1.1 Open question dataset

Those dataset are for DPO training to train our model to be a good teaching assistant. We want our model to provide not only accurate answers but also clear, step-by-step explanations. In the "chosen" entry, we include a comprehensive answer that is detailed, easy to understand, and educational. In the "rejected" entry, we place answers that might be correct or slightly incorrect but lack sufficient explanatory detail.

In addition to student data provided, we also collected open source preference data, but meticuolsy chose STEM related ones. Also we tried to minimize unbalance of subjects, so that for instnace most data are just about mathematics. In total, we had 23584 preference pairs.

- **student data:**
  This is dataset generated by students using chatGPT API on questions from EPFL courses. Both better and worse answeres were generated, and annotated by each students. It contains 12955 preference pair from 730 distinct prompts. However, it might contains some mistakes.
- **Preference pair dataset in Huggingface:**
  In addition to student data, we used opensource preference pair related to STEM.
- **distilabel-math-preference-dpo:** It consists 2418 preference pair where prompt was highschool level math questions without needing much calculation.
- **OpenHermesPreferences:** We chose preference pair from just 2 sources, 2000 pairs from *Camel AI* and 2000 pairs from *glaive-code-assist*. *Camel AI* has subject mostly related to STEM ranging from math, physics, chemistry, biology, coding, but also had some social science. *glaive-code-assist* was about coding.

- **distilabel-capybara-dpo-7k-binarized:** We only choose pairs from *Theorem QA*, which has 574 preference pair about college level mathematics.

- **Preference pair generated using gold answer and prompt dataset** We also collected preference pair where we get prompt and gold answer from open source dataset, and we generate worse answer using chatGPT API.

- **camel-AI/physics:** We thought there is not much additional data on physics, which has big portion of STEM university courses, so we collected another 2000 pairs. These questions was about serious college level physics questions.

- **STEM-AI-mtl/Electrical-engineering:** This consits 1131 preference pair of college level electrical engineering questions.

- **garage-bAInd/Open-Platypus:** This consits 15,584 gold answer and prompt. We only choose data from *scibench* and *ARB* which has 320 pairs broadely about college level STEM subjects.

4.1.2 MCQADataset

In addition to 11114 pairs of MCQA pairs, we also used additional open source data. In total, we used 23101 pairs of MCQA pairs. For additional data that has only golden answer, we selected randomly between A D that is not answer and put is as rejected answer.

- **student data:** This is dataset generated by students using chatGPT API on questions from EPFL courses. Both better and worse answers were generated, and annotated by each students. It containes 11114 pairs of MCQA answers.

- **cais/mmlu:** It consists questions relevent from high-school college level courses. We only chose STEM related subjects among those. We used 4200 pairs from this dataset. We randomly assigned choice that is not given answer as rejected answer.

- **allenai/ai2_arc:** It consists questions relevent from various STEM subjects. WE used the whole dataset, which consits two subjects ['ARC-Challenge', 'ARC-Easy'], in total 7787 pairs. We randomly assigned choice that is not given answer as rejected answer.

4.1.3 RAG Database

By investigating student MCQ dataset, we plotted subject each question in. We used chatGPT API to get one subject for one question. By investigating frequency of each subject, we noticed subject such as Cryptography, Statistics, Machine learning, Cyber Security, Linguistics(NLP) are important. We gathered various open source text data about these subjects. The database was in .jsonl file with 0.12gb in size.

- **Textbook:** We gathered open source textbook about these subjects. On total 60 textbooks were gathered. These include openstax textbooks.

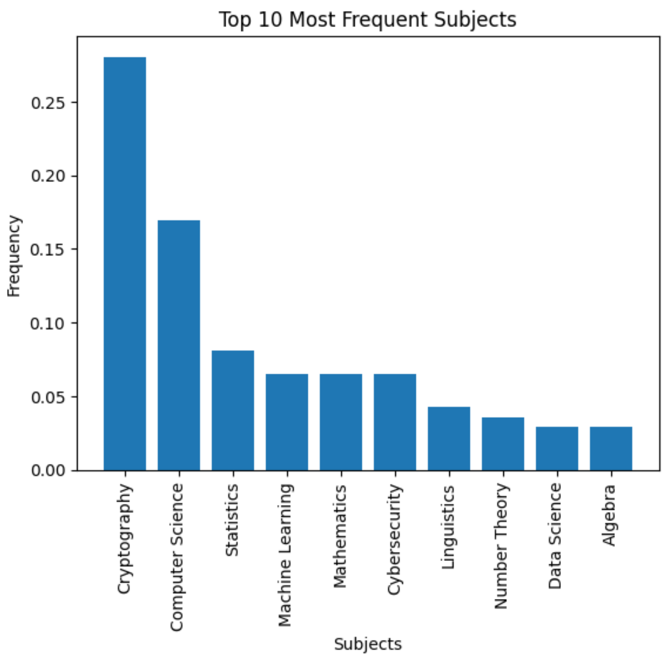


Figure 1: Top 10 Most Frequent Subjects

- **Lecture Notes:** We gathered 12 different lecture notes, mostly around modern computer science courses such as Natural Language Processing.

- **Wiki corpus:** We gathered "rag-datasets/rag-mini-wikipedia", and "rag-datasets/rag-mini-bioasq" from Hugginface.

4.2 Evaluation method

- **Open questions:** Our primary evaluation method for open questions is accuracy of selecting prefered answer among pairs. This is done by calculating log probability of each answer, from same prompt.

  We also report 3 different types of evaluation metric, BERT score, LLM based score(GEval), and human based preference. We choose 20 prompt and gold answer from sciBENCH dataset, and compare it with generated answer. For 20 prompt left for evaluation in student dataset, since it doesn't have gold answer we only report human based preference.

- **MCQA questions:**For mcqa questions, we directly calculated accuracy comparing with golden answer. We compare base models, DPO-finetuned model, DPO + RAG model, and DPO + Quantization model.

4.3 Baselines

We used distill-gpt2, and untrained unsloth/zephyr-sft as baseline. We chose distill-gpt2 to see effect of scale, and untrained unsloth/zephyr-sft to show the effect of finetuning and RAG.

4.4 Experimental details

- **LoRA parameters:** We trained with LoRA adapter, and parameter for LoRA was r = 64, $\alpha = 64$. We saved our adapter for each epoch, so that we can evaluate our model on each epoch, since LoRA finetuning is vulnerable to overfitting.

- **other hyperparameters:** optimizer=adamw8bit, lr=5e-6, epochs=3, batch size=2, gradient accumulation steps = 4

- **data split:** we left pairs from 20 different prompt(About 350 pairs) of open questions. For MCQ questions we used 356 paris from mcqa_example.json provided by TAs. These test sets are used to calculate accuracy for choosing prefered pair for open questions, and choosing correct answer for MCQA. Beside these testsets, we also had 100 different high quality prompt and gold answer from scibench dataset, so that we can compare quality of open generated answer with golden answer using BERT score, GEval score.

4.5 Results

4.5.1 Open question DPO results

- **Learning Curve:**

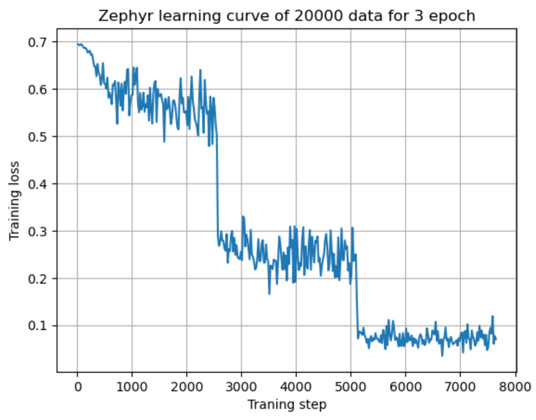


Figure 2: Training Loss

- **Accuracy of choosing prefered pair:**

	base	Epoch 1	Epoch 2	Epoch 3
Accuracy	0.52	0.58	0.60	0.57

Table 1: BERT Scores across Different Epochs.

Here base model refers to untrained zephyr-sft model. BERT score is calculated between generated answer and golden answer for 350 different test pairs.

- **Quality of generation: Human based preference**

  We matched 2 different schemes, one for generation of original model versus generation of model DPO trained on 2 epoch, and another for generation of model DPO trained on 2 epoch versus generation from 3 epochs. Four group members chose which one of two answer is better. epoch 2 model was best on performance.

	base	Epoch 2	Epoch3
base vs epoch 2	14.6%	85.4%	
epoch 2 vs epoch 3		60.8%	39.2%

Table 2: Human preference percentage

- **Quality of generation: Comparing golden answer from scibench**

  Here we report BERT score between generated answer and golden answer from scibench dataset. See Table 3 below.

	base	Epoch 1	Epoch 2	Epoch 3
f1	0.43	0.77	0.82	0.82
precision	0.46	0.77	0.82	0.82
recall	0.41	0.77	0.82	0.83

Table 3: BERT Scores across Different Epochs. Here base model referes to untrained zephyr-sft model. BERT score is calculated between generated answer and golden answer for 350 different test pairs.

We also report LLM based evaluation metric, and we specificlaly used openAI API key to use GEval. For correctness, we gave criteria="Determine whether the actual output is factually correct based on the expected output." For completeness, we gave "how thoroughly the actual output answers the question in the input prompt". For coherence, we gave "the collective quality of all sentences in the actual output". For instructiveness, we gave "how kindly the model provides instructions to the user in the actual output". Scores were scaled from 0 to 1. See Table 4 below.

	base	Epoch 1	Epoch 2	Epoch 3
correctness	0.01	0.03	0.05	0.05
completeness	0.13	0.16	0.21	0.22
coherence	0.11	0.19	0.22	0.19
instructive	0.12	0.30	0.29	0.29

Table 4: GEval Scores across Different Epochs. Scores are normalized between [0,1], where larger score means better.

- **Addendum: Distill GPT-2**

  Besdie training zephyr-sft with 7B parameter, we also trained Distill GPT-2 seperately, and we report quality of generation here. BElow is BLEURT (https://github.com/google-research/bleurt) score. BLEURT is an evaluation metric for Natural Language Generation. It takes a pair of sentences as input, a reference and a candidate, and returns a score that indicates to what extent the candidate is fluent and conveys the meaning of the reference. It is comparable to BLEU, BERTScore, and COMET from Assignment 3. 1000 sentences are generated and compared to the reference. We are not doing translation but a good answer should still conveys some of the meaning of the reference.

BLEURT	**base_GPT2**	**trained GPT2**
students	-0.9893	-0.7856
hermes	-0.9615	-0.8444
argilla_math	-0.8227	-0.7681

Table 5: Scores for Base Model and Trained Model

4.5.2 MCQA results

See the table 6 below. We used 356 pairs from mcqa_example.jsonl provided by TA as test set. There were 3 different subjects in this dataset, and since accuracy among subjects were significantly different, we report them seperately. We report untrained model as base model, and model just trained on DPO, model trained on DPO and using RAG, and model trained on DPO but quantied.

	base	DPO	DPO+RAG	DPO+Q
ML	0.37	0.40	0.42	0.40
Bio	0.62	0.63	0.63	0.62
Chem	0.53	0.56	0.58	0.55

Table 6: MCQ Accuracy for different models for different subjects. Here base model referes to untrained zephyr-seft.Q refers to 4-bit quantized model.

4.5.3 Size change due to Quantization

We report model size in memory before and after quantization. See Table 7 below.

	Before Q.	After Q.
size	29200MB	4948MB

Table 7: Model size in memory of before/after Quantization.

5 Analysis

5.1 Open questions

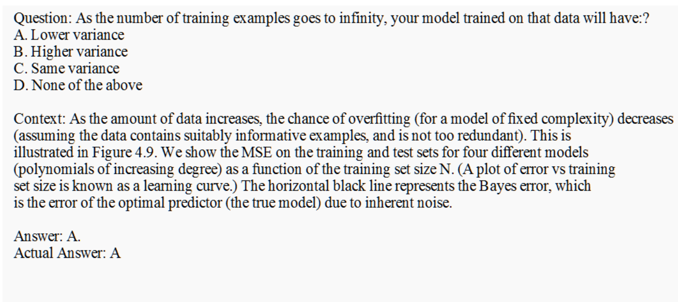
- **Effect of overfitting:** Note that accuracy of choosing prefered pair increased from untrained to epoch 1 significantly. Then for epoch 2 accuracy increase slightly, and then for epoch 3 it decrased slightly. This could be think of overfitting, where our dataset had only 24k diffent pairs, and LoRa finetuning is also vulnerable in overfitting. If we look at generation, we can see that for epoch 3 it starts to repeat unreasonalbe formulas and equations which it saw during traiing. We used epoch 2 model as our best model.

- **Choice of evaluation metric:** OUr quality of generation was not satisfactory, as you can see from GEval score. Correctness was always below 0.05(1 scaled to be ground truth and 0 to be worse). This could be due to fundamental limit of LoRA finetuning using small dataset by DPO, and also we are using 7B model, which is capable of everyday conversation, but not capable of reasoining with complicated logic and requiring mutliple steps. Thus in this case we believe most critical evaluation metric is choosing prefered answer along pair, which we reported in "Accuracy of choosing prefered pair"

5.2 MCQA questions

- **Performance difference among subjects:** There was significant difference of mcqa accuracy among subjects. For subject requiring more logic and complex reasoning, performance was worse. We can see that for subject like machine learnining, accruacy is relatively low with/without DPO training/RAG. And also accuracy was relatively high with/without DPO training/RAG. For instnace, for biology question, untrained model was already getting high performance and finetuning and adding database didn't changed performance much.

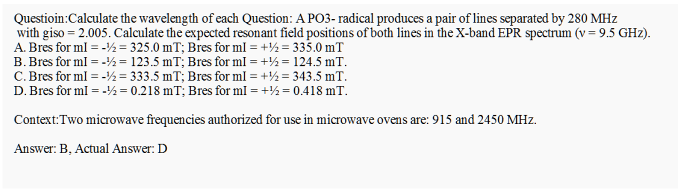
- **RAG being detrimental on subjects outside database:** Although we tried to add as many resources in RAG database it was still 0.12GB, and there were questions which are irrevalent of any of data in database. In this case, since RAG is retrieving wrong context, it could actually lead to wrong answer, which it got it right without RAG.


Question: As the number of training examples goes to infinity, your model trained on that data will have:?
A. Lower variance
B. Higher variance
C. Same variance
D. None of the above

Context: As the amount of data increases, the chance of overfitting (for a model of fixed complexity) decreases (assuming the data contains suitably informative examples, and is not too redundant). This is illustrated in Figure 4.9. We show the MSE on the training and test sets for four different models (polynomials of increasing degree) as a function of the training set size N. (A plot of error vs training set size is known as a learning curve.) The horizontal black line represents the Bayes error, which is the error of the optimal predictor (the true model) due to inherent noise.

Answer: A.
Actual Answer: A

Figure 3: Example of RAG being beneficial


Questioin:Calculate the wavelength of each Question: A PO3- radical produces a pair of lines separated by 280 MHz with giso = 2.005. Calculate the expected resonant field positions of both lines in the X-band EPR spectrum (ν = 9.5 GHz).
A. Bres for mI = -½ = 325.0 mT; Bres for mI = +½ = 335.0 mT
B. Bres for mI = -½ = 123.5 mT; Bres for mI = +½ = 124.5 mT.
C. Bres for mI = -½ = 333.5 mT; Bres for mI = +½ = 343.5 mT.
D. Bres for mI = -½ = 0.218 mT; Bres for mI = +½ = 0.418 mT.

Context:Two microwave frequencies authorized for use in microwave ovens are: 915 and 2450 MHz.

Answer: B, Actual Answer: D

Figure 4: Example of RAG being detrimental

- **Quantization leading similair performance:** We can see that quantzation made model size much smaller, yet mcqa accuracy was similar to before.

6 Ethical considerations

6.1 Adapting to different languages

To adapt to different languages, we can use Sparse Fine-Tuning method (SFT) for each language. This method only trains small subset of whole parameter in model for different languages. Since we usedLoRA finetuning for task, finetuning to other language could be done following this guideline (Chen et al., 2024). We will compose task sub-network and language specific sub-network. For instance, if we want to adapt to Urdu, we could finetune our urdu specific sub-network using Wikipedia text on Urdu using MLM, then compose this subnetwork with task sub-network.

6.2 Adapting to signed languages:

We could combine our model with signed language (video) to English text translator (Yin et al., 2021). From signed language (video) to text we could use (Selvaraj et al., 2022), and from text to signed language we could use (Baltatzis et al., 2024) (data is open-sourced, but we need to train diffusion model).

7 Conclusion

In this project, our objective was to train an AI tutor specialized in the course content at EPFL, aiming to assist students in their learning by providing a language model capable of generating answers to questions in science, technology, computer sciences, engineering, and mathematics. Our approach involved three key steps: data collection, model training using Direct Preference Optimization (DPO), and enhancing the model's ability to retrieve, as well as still being accurate yet being smaller in size.

Our main findings reveal that the model, after being trained with DPO, showed improvement picking out prefered answer from pairs for open questions, and getting accurate answer for MCQA. Furthermore, adding information retrieval from database slightly imporve performance. In addition, by quantizing we had model 6 times smaller yet achieveing similar performance.

There was still limitations to our problem. First is that performance imporvement wasn't significant. Also despite accuray of picking better answer improved, quality of generated answer itself wasn't satisfactory. These could be due to many reasons, our model was 7B in the first place and dataset was only 24k, 23k for open and MCQ questions. We didn't finetuned the whole model but only finetuned adapter tensors using LoRa.

Future work could explore scaling up the model and datasets, incorporating more advanced reasoning and logic capabilities, and adapting the model to different languages and signed languages. These steps could provide deeper insights and further validate our findings, significantly advancing our understanding and contributing to the practical applications of AI in education. By addressing these areas, future studies could enhance the educational experience and outcomes for students, ensuring a more effective and accessible AI tutor.

References

Vasileios Baltatzis, Rolandos Alexandros Potamias, Evangelos Ververas, Guanxiong Sun, Jiankang Deng, and Stefanos Zafeiriou. 2024. Neural sign actors: A diffusion model for 3d sign language production from text.

T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

B. Busa-Fekete, B. Szörényi, P. Weng, W. Cheng, and E. Hüllermeier. 2014. Preference-based reinforcement learning: Evolutionary direct policy search using a preference-based racing algorithm. *Machine Learning*, 97(3):327–351.

Arnav Chavan, Raghav Magazine, Shubham Kushwaha, Mérouane Debbah, and Deepak Gupta. 2024. Faster and lighter llms: A survey on current challenges and way forward. *arXiv preprint arXiv:2402.01799*.

Pinzhen Chen, Shaoxiong Ji, Nikolay Bogoychev, Andrey Kutuzov, Barry Haddow, and Kenneth Heafield. 2024. Monolingual or multilingual instruction tuning: Which makes a better Alpaca. In *Findings of the Association for Computational Linguistics: EACL 2024*.

H. W. Chung, L. Hou, S. Longpre, B. Zoph, Y. Tay, W. Fedus, Y. Li, X. Wang, M. Dehghani, S. Brahma, A. Webson, S. S. Gu, Z. Dai, M. Suzgun, X. Chen, A. Chowdhery, A. Castro-Ros, M. Pellat, K. Robinson, D. Valter, S. Narang, G. Mishra, A. Yu, V. Zhao, Y. Huang, A. Dai, H. Yu, S. Petrov, E. H. Chi, J. Dean, J. Devlin, A. Roberts, D. Zhou, Q. V. Le, and J. Wei. 2022. Scaling instruction-finetuned language models. Manuscript.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. 2022. Llm.int8(): 8-bit matrix multiplication for transformers at scale. *arXiv preprint arXiv:2208.07339*. Published at NeurIPS 2022. Camera-ready version.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2022. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*. ICLR 2023.

Phillip J. Grimaldi, Debshila Basu Mallick, Andrew E. Waters, and Richard G. Baraniuk. 2019. Do open educational resources improve student learning? implications of the access hypothesis. *PLOS ONE*.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models.

J. Jeon and S. Lee. 2023. Large language models in education: A focus on the complementary relationship between human teachers and chatgpt. *Education and Information Technologies*.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*. Submitted on 10 Oct 2023.

E. Kasneci, K. Seßler, S. Küchemann, M. Bannert, D. Dementieva, F. Fischer, U. Gasser, G. Groh, S. Günnemann, E. Hüllermeier, and et al. 2023. Chatgpt for good? on opportunities and challenges of large language models for education.

J. Kreutzer, J. Uyheng, and S. Riezler. 2018. Reliability and learnability of human bandit feedback for sequence-to-sequence reinforcement learning. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1777–1788. Association for Computational Linguistics.

Shuming Ma, Hongyu Wang, Shaohan Huang, Li Dong, Ruiping Wang, Jilong Xue, and Furu Wei. 2024. The era of 1-bit llms: All large language models are in 1.58 bits. *arXiv preprint arXiv:2402.17764*. Published on Feb 27, Submitted by akhaliq on Feb 28.

S. Mishra, D. Khashabi, C. Baral, and H. Hajishirzi. 2022. Cross-task generalization via natural language crowdsourcing instructions. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3470–3487. Association for Computational Linguistics.

L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. F. Christiano, J. Leike, and R. 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.

A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. 2019. Language models are unsupervised multitask learners. Manuscript, OpenAI.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *arXiv preprint arXiv:2305.18290*. Submitted on 29 May 2023 (v1), last revised 13 Dec 2023 (this version, v2).

Prem Selvaraj, Gokul Nc, Pratyush Kumar, and Mitesh Khapra. 2022. OpenHands: Making sign language recognition accessible with pose-based pretrained models across languages. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2114–2133, Dublin, Ireland. Association for Computational Linguistics.

N. Stiennon, L. Ouyang, J. Wu, D. M. Ziegler, R. Lowe, C. Voss, A. Radford, D. Amodei, and P. Christiano. 2022. Learning to summarize from human feedback. Manuscript.

Hugo Touvron, Thibaut Lavril, Guillaume Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. Preprint.

Lewis Tunstall, Edward Beeching, Nathan Lambert, Nazneen Rajani, Kashif Rasul, Younes Belkada, Shengyi Huang, Leandro von Werra, Clémentine Fourrier, Nathan Habib, Nathan Sarrazin, Omar Sanseviero, Alexander M. Rush, and Thomas Wolf. 2023. Zephyr: Direct distillation of lm alignment. *arXiv preprint arXiv:2310.16944*. Submitted on 25 Oct 2023.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. 2022. Emergent abilities of large language models. *Transactions on Machine Learning Research (TMLR)*.

Y. Wu and B. Hu. 2018. Learning to extract coherent summary via deep reinforcement learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'18/IAAI'18/EAAI'18. AAAI Press.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. 2022. Smoothquant: Accurate and efficient post-training quantization for large language models. *arXiv preprint arXiv:2211.10438*. ICML 2023. First two authors contributed equally to this work.

Kayo Yin, Amit Moryossef, Julie Hochgesang, Yoav Goldberg, and Malihe Alikhani. 2021. Including signed languages in natural language processing.

Y. Yue, J. Broder, R. Kleinberg, and T. Joachims. 2012. The k-armed dueling bandits problem. *Journal of Computer and System Sciences*, 78(5):1538–1556. JCSS Special Issue: Cloud Computing 2011.

A Appendix (optional)

If you wish, you can include an appendix, which should be part of the main PDF, and does not count towards the page limit. Appendices can be useful to supply extra details, examples, figures, results, visualizations, etc., that you couldn't fit into the main paper. However, your grader does not have to read your appendix, and you should assume that you will be graded based on the content of the main part of your paper only.

Here, you can find some generation of the model from milestone 2 :

Question

: Consider the following contains function defined on Iterable (in particular, it accepts both Vector and List). def contains[A](l: Iterable[A], elem: A): Boolean = val n = l.size if n <= 5 then for i <- l do if i == elem then return true false else val (p0, p1) = parallel(contains(l.take(n / 2), elem), contains(l.drop(n / 2), elem)) p0 || p1 Let nn be the size of l. Assume that drop and take run in $\Theta(1)$ on Vector and $\Theta(n)$ on List. What is the asymptotic depth of contains if it is called on a List?

Answer

: The function contains is defined in terms of parallel, which is a higher-order function that takes two arguments and returns a tuple of their results. In this case, contains is splitting the input list into two halves and calling itself recursively on each half. The results of these recursive calls are then combined using the || operator, which is short-circuit or (in this case) parallel evaluation.

If contains is called on a List, the drop operation will run in $\Theta(n)$ time, where n is the size of the list. This is because drop iterates over the list and returns a new list without the first $n/2$ elements. The take operation will also run in $\Theta(n)$ time, as it returns the first $n/2$ elements of the list.

The recursive calls to contains will continue until the size of the list is at most 5. At this point, the function will perform a linear search of the remaining elements in the list. The linear search will run in $\Theta(5)$ time, or $\Theta(1)$ time.

The total time complexity of contains on a List is therefore $\Theta(n)$ for the drop and take operations, plus $\Theta(1)$ for the linear search when the list size is at most 5. Therefore, the asymptotic depth of contains on a List is $\Theta(\log n)$, where n is the size of the list.

In other words, the function will split the list in half until the size of each half is at most 5, at which point it will perform a linear search of the remaining elements. This process will continue until the size of the list is 1 or 0, at which point the function will return true or false accordingly. The maximum depth of this recursion is $\log n$, where n is the size of the list.

(with correction for latex):

Question

We have a collection of rectangles in a plane, whose sides are aligned with the coordinate axes. Each rectangle is represented by its lower left corner (x_1, y_1) and its upper right corner (x_2, y_2). All coordinates are of type Long. We require $x_1 \leq x_2$ and $y_1 \leq y_2$. Define a case class Rectangle storing two corners.

Write a program that takes a list of rectangles as input and returns the number of pairs of non-overlapping rectangles. Two rectangles overlap if they share any point except for their common boundary.

Answer

```
case class Rectangle(lowerLeft: (Long, Long),
upperRight: (Long, Long)) {
  require(lowerLeft._1 <= upperRight._1
  && lowerLeft._2 <= upperRight._2)
}

object NonOverlappingRectangles {
 def count(rectangles: List[Rectangle]): Int = {
    val rectanglesSet = rectangles.toSet
    rectanglesSet.size
  }
}
```

Explanation

- We define a case class Rectangle to store the lower left and upper right corners of each rectangle. We require that the x-coordinate of the upper right corner is greater than or equal to the x-coordinate of the lower left corner, and the y-coordinate of the upper right corner is greater than or equal to the y-coordinate of the lower left corner.
- We define a function count that takes a list of Rectangle objects as input and returns the number of non-overlapping rectangles.
- We convert the input list of Rectangle objects to a set to remove duplicates and ensure that each rectangle is counted only once.
- We return the size of the set, which is the number of non-overlapping rectangles.

Example Usage

```
val rectangles = List(
  Rectangle((1, 1), (4, 4)),
  Rectangle((2, 2), (5, 5)),
  Rectangle((3, 3), (6, 6))
)

val nonOverlappingRectangles =
 NonOverlappingRectangles.count(rectangles)
println(s"Number of non-overlapping
rectangles: $nonOverlappingRectangles")
```

Output

```
Number of non-overlapping rectangles: 2
```

In this example, the rectangles $(1,1)-(4,4)$ and $(3,3)-(6,6)$ are non-overlapping, while the rectangle $(2,2)-(5,5)$ overlaps with both of them. Therefore, the output is 2.

Question:

Which of the following scheduler policies are preemptive?

Options: A. FIFO (First In, First Out) B. SJF (Shortest Job First) C. STCF (Shortest Time to Completion First) D. RR (Round Robin) Question: Which of the following scheduler policies are preemptive?

Options: A. FIFO (First In, First Out) B. SJF (Shortest Job First) C. STCF (Shortest Time to Completion First) D. RR (Round Robin)

Answer:

A, B, and C are preemptive scheduler policies as they allow a higher priority process to preempt a running process. FIFO is a non-preemptive scheduler policy as it does not allow a process to be preempted by a higher priority process. Round Robin (RR) is a non-preemptive scheduler policy as it does not allow a process to be preempted by a higher priority process.

Explanation: Preemptive scheduler policies allow a higher priority process to preempt a running process, while non-preemptive scheduler policies do not allow a process to be preempted by a higher priority process. FIFO (First In, First Out) is a non-preemptive scheduler policy as it does not allow a process to be preempted by a higher priority process. SJF (Shortest Job First) and STCF (Shortest Time to Completion First) are preemptive scheduler policies as they allow a higher priority process to preempt a running process. Round Robin (RR) is a non-preemptive scheduler policy as it does not allow a process to be preempted by a higher priority process.

Preemptive scheduler policies are useful in real-time systems where processes with higher priority need to be executed immediately. Non-preemptive scheduler policies are useful in general-purpose systems where processes are executed in the order they arrive.

In summary, A, B, and C are preemptive scheduler policies, while D is a non-preemptive scheduler policy.

contributions

- **Research:** Lacour David || Hyunmo Kang
- **Datasets:** Lacour David || Hyunmo Kang
- **Training:** Lacour David || Hyunmo Kang
- **Evaluation:** Lacour David || Hyunmo Kang
- **Debugging:** Lacour David || Hyunmo Kang
- **RAG:** Jiewei Li || Hyunmo Kang
- **Quantization:** Lacour David || Margot Malis