AI tutor for EPFL students

Nikolay Mikhalev | 314355 | nikolay.mikhalev@epfl.ch
Nicolas Vuillod | 313321 | nicolas.vuillod@epfl.ch
Niko Pindao | 314958 | niko.pindao@epfl.ch
Anders Hominal | 297073 | anders.hominal@epfl.ch
NNNandAnders

Abstract

The challenge of providing effective study aids for students is critical, particularly in preparing for rigorous academic examinations such as those at École Polytechnique Fédérale de Lausanne (EPFL). Our research addresses this problem by developing a large language model (LLM) specifically designed to assist students in comprehending and answering multiple-choice questions from past EPFL exams. Leveraging LLama 3 as the foundation, we fine-tuned this model using a STEM dataset with Direct Performance Optimization (DPO) formatted questions. The fine-tuned model exhibits high accuracy in answering multiple-choice questions, surpassing baseline models. Thus, the model enhances students' understanding by explaining the rationale behind correct answers, improving their learning experience and conceptual clarity.

1 Introduction

The ability to effectively study and prepare for exams is crucial for students, especially in demanding STEM fields. At EPFL, students face particularly challenging multiple-choice questions (MCQ) in their exams, which test not only their knowledge but also their understanding of complex concepts. These questions often require deep comprehension and the ability to apply theoretical principles to specific problems, which can be tricky for models that are simply pre-trained, without a fine-tuning step with STEM data.

To accomplish such a complex task, we used a performant pre-trained model, Llama-3-8B-Instruct, the last LLM released by Meta in April 2024. It is an auto-regressive language model that uses an optimized standard decoder-only transformer architecture. The fine-tuning step has been performed using Direct Performance Optimization (DPO) on datasets containing STEM samples (Rafailov et al., 2023). The goal of such a training method is to teach the model the proper way to answer, with a detailed development. Then, the model's evaluation is performed on MCQA data and we want to get the key of the correct answer to compute the accuracy. To do so, we add a post-processing step which analyses the generated answer and extracts only the corresponding key.

Moreover, we improved the model's performance using two updates: Retrieval Augmented Generation (RAG) and Quantization of the model. The first method incorporates relevant external knowledge from databases in the prompt, enabling the model to provide well-informed and up-to-date responses to user queries (Gao et al., 2024). The second one is the process of reducing the precision of numerical values, to enable efficient inference and reduce memory requirements without compromising performance (Dettmers et al., 2022).

2 Related Work

2.1 Direct Performance Optimization

Motivated by the challenges of applying reinforcement learning algorithms on large-scale problems such as fine-tuning language models, our goal is to derive a simple approach for policy optimization using preferences directly.

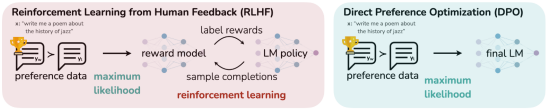

Figure 1: Direct Preference Optimization (DPO) optimizes language models for human preferences using a simple classification objective, thus avoiding the need to use a separate reward model and reinforcement learning to maximise it.

Unlike prior reinforcement learning with human feedback (RLHF) methods (Azar et al., 2023), which learn a reward and then optimize it via reinforcement learning (RL), our approach leverages a particular choice of reward model parameterization

that enables the extraction of its optimal policy in closed form, without an RL training loop (Rafailov et al., 2023). Indeed, DPO is able to bypass both fitting an explicit reward and performing RL to learn the policy using a single maximum likelihood objective (see Figure 1).

2.2 Low-Ranked Adaptation (LoRA)

A neural network contains many dense layers which perform matrix multiplication. The weight matrices in these layers typically have full rank. When adapting to a specific task, research has shown that the pre-trained language models have a low "intrinsic dimension" and can still learn efficiently despite a random projection to a smaller subspace (Aghajanyan et al., 2020).

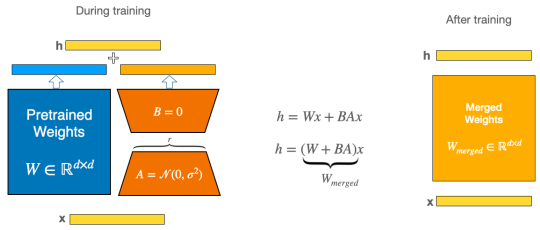

Figure 2: Diagram representing the weight matrix decomposition during training.

In the Transformer architecture, there are four weight matrices in the self-attention module (W_q, W_k, W_v, W_o) and two in the MLP module. We treat W_q (or W_k, W_v) as a single matrix of dimension $d_{model} \times d_{model}$, even though the output dimension is usually sliced into attention heads. We decided to apply LoRA only on the attention weights for downstream tasks and freeze the MLP modules (so they are not trained in downstream tasks) both for simplicity and parameter efficiency (Hu et al., 2021).

The most significant benefit comes from the reduction in memory and storage usage. This allows us to train with significantly fewer GPUs and avoid I/O bottlenecks. Another benefit is that we can switch between tasks while deployed at a much lower cost by only swapping the LoRA weights as opposed to all the parameters. This allows for the creation of many customized models that can be swapped in and out on the fly on machines that store the pre-trained weights in VRAM.

2.3 Prompting strategies and Retrieval Augmented Generation

To get more accurate answers from our LLM we implemented two techniques.

The first one is a specific "prompting" method for conditioning the model's answer. Large language models (LLMs) have demonstrated considerable performance on complex reasoning tasks by breaking them down into intermediate steps before providing an answer. This process, known as chain-of-thought (CoT) prompting, involves two major paradigms: Zero-Shot-CoT and Manual-CoT. Zero-Shot-CoT uses a simple prompt to facilitate reasoning chains without task-specific demonstrations ("Let's think step by step", (Kojima et al., 2022)), while Manual-CoT relies on manually crafted demonstrations, where we give along with the query and the reasoning chain prompt a similar question and it's reasoning, to get superior performance. Both prompting methods demonstrated higher performance in answering complex tasks. Amongst all the strategies, the one displaying the highest accuracy in answers is Few-Shots-CoT, which is the same as Manual-CoT but with several similar questions and their reasoning.

The second strategy to enhance the model's answers is by integrating a Retrieval Augmented Generation model (RAG) to our pipeline. RAG improves LLMs by incorporating external knowledge from databases, enabling the model to provide well-informed and up-to-date responses to user queries (Gao et al., 2024). RAG can be added during training or during the inference. We decided to implement RAG exclusively during inference, as it was originally developed to enhance LLMs' ability to handle complex and knowledge-intensive tasks at this stage ((Gao et al., 2024)). Furthermore, by using RAG only during inference, we avoid the prolonged training times caused by additional input tokens and eliminate the need to modify the model's loss function (retrieval accuracy).

2.4 Quantization

Quantization reduces the precision of numerical values to enable efficient inference and decrease memory requirements without compromising performance (Dettmers et al., 2022). During implementation, we made minor adjustments to ensure the model's compatibility with the quantization process. These adjustments included optimizing certain layers and operations that are more sensitive

to quantization, ensuring they function effectively and efficiently within the quantized framework.

During the training process, we loaded the model in 8-bit precision using bits and bytes python module to reduce the memory demands and make training possible. The trained LoRA adapters were trained and saved in float 16-bit precision.

3 Approach

We introduce the fundamental architecture of our base model, Llama 3 8B and provide a detailed description of the DPO LoRA training objective utilized in this work. Additionally, we outline further modifications to the pipeline, including the use of RAG for augmenting prompts and quantization techniques for loading the model with reduced precision. These modifications enhance memory efficiency and ensure compatibility with lighter hardware.

3.1 Llama 3 architecture

Llama 3 8B Instruct is an auto-regressive language model that uses an optimized standard decoder-only transformer architecture and a tokenizer with a vocabulary of 128K tokens that encodes language much more efficiently.

Furthermore, it adopted a grouped query attention (GQA) approach to improve the model's inference efficiency. Finally, the model has been pretrained on sequences of 8,192 tokens, using a mask to ensure self-attention does not cross document boundaries.

To this date, it is the most performant model in the < 10B tier, having been recently released by Meta, Llama scores high on multiple benchmarks, thus proving its utility for our purposes.

3.2 LoRA training with a DPO training objective

Instead of doing full fine-tuning, which would be impossible with our resources, we trained adapters using the LoRA training technique, meaning that only linear layers of the transformer blocks were tuned, thus drastically reducing the memory demands. This way of training is widely used because of its performance and productivity, which it has proved to be once again in our case.

Furthermore, our base model (Llama 3B) has been trained with the DPO loss objective, using the Transformer Reinforcement Learning (TRL) library from HuggingFace together with PEFT, which implements LoRA training and is compatible with loading the model in a quantized version for more memory efficient training.

The dataset we used for training this time consisted of 150k samples after it had been reduced from over 1M by our data preprocessing pipeline. Indeed, we decided to train the model using another dataset, as the DPO training accuracy of Milestone 2 was not satisfying enough due to the poor quality of the crowdsourced dataset. Given the amount of data, we only trained our model for one epoch, which already took around 3 days and yielded plausible results. We have saved the LoRA adapter weights every 1'000 iterations (with gradient accumulation of 6 and batch size of 2 which means every 12'000 samples) in order to compare them later.

The loss function used during the training is the following :

$$\mathrm{L}_{DPO}(\pi_\theta; \pi_{ref}) = -E_{(x,y_w,y_l)\sim D}\left[\log \sigma\left(\beta \log \frac{\pi_\theta(y_w|x)}{\pi_{ref}(y_w|x)} - \beta \log \frac{\pi_\theta(y_l|x)}{\pi_{ref}(y_l|x)}\right)\right].$$

where π_θ is the optimal policy, π_{ref} is the reference policy, y_w stands for the preferred response, y_l stands for the rejected one and β is a parameter controlling the deviation of the optimal policy from the reference one (fixed to 0.1).

By minimizing this loss function, we are increasing the likelihood of the preferred completions and decreasing the likelihood of rejected completions.

We have considered other training objectives such as IPO and KTO to improve the models' performance, described in detail in our progression report, however, changing the training dataset to a more liable and versatile one, has shown to resolve our problems as shown on the training graphs comparison in the appendix. Considering that training seemed to converge correctly this time and that it took around 80 hours of training on an RTX309 (24G VRAM) GPU card to get the presented results, we found it unnecessary and too costly to do yet another training but with a different objective.

3.3 RAG implementation

The Retrieval-Augmented Generation (RAG) workflow consists of several key steps: Indexing, Retrieval, Generation and Augmentation ((Lewis et al., 2020)).

During Indexing, documents are collected and pre-processed into a uniform format. The document's text is then segmented into smaller chunks (context limitations of language models) which are encoded into vector representations using an em-

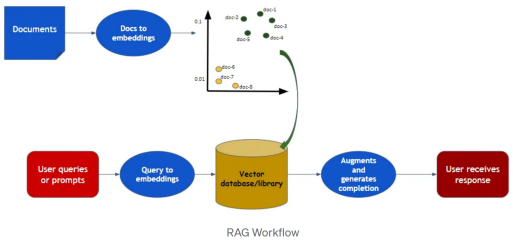

Figure 3: Illustration of the Retrieval-Augmented Generation (RAG) process. Documents are embedded into vectors, relevant documents are retrieved based on the query, and these documents are added to the initial query to generate a more accurate and context-rich response.

bedding model. These vectors are stored in a vector database to enable efficient similarity searches and accelerate the retrieval process (by avoiding embedding the whole database each time the user inputs a query). To pre-process the PDF files to plain text we harnessed PYPDF2 as it displays higher speed in processing the files and higher accuracy when testing the model's retrieval capabilities compared to pdfplumber. Furthermore, an advantage of this approach (PYPDF2) is that documents can be retrieved with page numbers allowing us to further manually verify the retrieved information. We further processed the file by embedding and indexing the collection of text documents for efficient retrieval.

Initially, we load the text files, split them into manageable chunks, and create embeddings using a pre-trained Hugging Face model (the sentence-transformer "MiniLM-L6-v2"). These embeddings are then indexed and saved using FAISS (Facebook AI Similarity Search), enabling quick similarity searches and avoiding increased running time due to document splitting and embedding. Finally, we augment the input file. The system reads each question from the input file and computes similarity scores between the query vector and the vectors of chunks in the database. The best chunk with the highest similarity to the query is retrieved and added as context to the questions, ultimately augmenting the input file with pertinent contextual information from the corpus, improving the quality and accuracy of the responses. The sentence-transformer "MiniLM-L6-v2" and FAISS were chosen based on several tests and combinations of different models and comparing their accuracy.

To further improve RAG we can perform augmentation. This step involves enhancing the LLM's performance through iterative refinement of the retrieval and generation processes. Techniques include iterative retrieval, recursive retrieval, and adaptive retrieval to improve the robustness and relevance of the generated content. We could not perform this step due to time and resource limitations, but implementing it would result in a higher answer accuracy (as explained by (Lewis et al., 2020)).

3.4 Quantization implementation

We opted to apply post-training quantization (PTQ) for inference. PTQ involves reducing precision after training our model, specifically focusing on compressing the trained model. To implement this quantization, we initially loaded our trained model and then proceeded to quantize this wrapped model in 4-bit before evaluating its performance during inference.

4 Experiments

4.1 Data collection

The selection of the datasets used for the training is a crucial step to fine-tune the model qualitatively.

1. **Crowdsourced dataset**: This dataset was created through the collective efforts of all our classmates during Milestone 1. We collected preference pairs by interacting with ChatGPT4, using questions from past EPFL exams across various subjects and several prompting strategies (zero-shot prompting, few-shot-prompting, ...). For each pair, we determined the best (chosen) and worst (rejected) answers for several criteria. This process yielded 21,596 samples of EPFL-focused STEM questions.

2. **STEM-DPO**: This dataset has been found on HuggingFace [*STEM_DPO* from user elfonthefly, accessed June 4, 2024] and regroups more than 1.2M samples. The level of the questions is similar to the ones from EPFL's old exams. Small preprocessing steps had to be performed to reduce the size of the dataset. We removed questions from the dataset that included references to internet sources (e.g., containing "https://...") and other websites and we also removed duplicated questions, given the fact that there was no rating between them, so we were not in a position to use it during

the DPO training (Pattnaik et al., 2024). We finally got 150k samples for the training.

3. **Inference MCQA**: This dataset is a subset of a STEM-oriented dataset found on HuggingFace [*stem_mcqa_questions* from mvujas user, accessed June 10, 2024], to stay consistent with the training process. It regroups 80 samples (20 samples per category, respectively math, physics, computer science and technical engineering) of the same level of difficulty than EPFL exam questions. It will be used to evaluate our models during inference. Each question has a total of four choices with only one correct answer.

4. **RAG dataset**: The dataset used to implement RAG is composed, for the most part, of free coursebooks found on Google or on University websites (as the MIT OpenCourseWare: https://ocw.mit.edu/). This database regroups subjects like physics (mechanics, thermodynamics, quantum), maths (linear algebra, calculus), electrical engineering, and other STEM related subjects. Additionally, we uploaded field science-related Wikipedia dataset from HuggingFace [*wikipedia_field_of_science* from user millawell, accessed June 3, 2024].

4.2 Evaluation methods

Rather than just looking at the accuracy of chosen/rejected choices produced by our model or other similarity metrics with the dataset, we also wanted to directly test the productivity of our model in solving its main task: answering questions correctly.

4.2.1 DPO evaluation

Basic proof of training results was however needed. Thus, we tried several metrics to evaluate if the generated output was closer to the chosen answer than the rejected one:

BertScore and **Comet**: These two metrics have been used to determine the proximity of the generated output with the chosen answer and with the rejected one. However, both of them struggled to get properly the semantics of the generated answer. The results were too random even if the outcome was closer to one of the two possibilities, that's why we avoided using them.

Logprob difference: This is the method we used to evaluate the model's performance. After generating the outcome with the fine-tuned model (policy), we computed the log probability of each token and we compared them with the ones of the chosen and the rejected answer and summed them all to have the overall probability (score). We did the same with the reference model to determine from which one we were closer. The rewards of the chosen answer and the one rejected are computed by making the difference between the scores of the policy model and the ones of the reference model.

Then we compared the two rewards, if the chosen reward is bigger than the rejected one, count it as a positive result. The accuracy is computed by calculating the ratio of the samples that are considered positive over the total number of samples.

4.2.2 MCQA evaluation

The main objective of our model is to be able to respond to MCQAs from EPFL exams. However, during training, we teach the model to answer DPO format questions enhancing developed responses and pushing the model to imitate COT produced by GPT4, not just outputting the correct key. To resolve this issue, we used a prompting strategy that tells the model to generate the answer under a certain format, to be able to extract the key answer easily with a RegEx rule. Thus, we ran an evaluation with the inference MCQA dataset on both models to see how well they performed compared to each other. The overall metric we use for comparison is the accuracy of answers produced by the model (number of correct answers divided by the overall number of questions).

4.3 Baselines

Considering a random choice of an answer out of 4 possible choices would have an accuracy of 0.25, this could be considered a bare minimum baseline. However, more realistically, we tend to compete with the performance of the base Llama3 model, proving the efficiency of applied techniques (fine-tuning and RAG) by providing better accuracy. Another objective is not losing much accuracy with quantization, and maybe even getting the same or better results on a quantized fine-tuned model than on a full base model. The **base model scored a mean accuracy of 0.865** in 10 different runs, thus providing us with a benchmark for further experimentation.

4.4 Experimental details

Concerning the different hyperparameters in such intensive tasks as LLM training, it is troublesome to provide a detailed search through experimentation, especially considering the number of other details such as the training data itself, the quantization during training, and further inference techniques. As described in the advancements report, we took hyperparameters from reports presented online. Mainly different experiments used similar hyperparameters, thus we were not eager to change them without any particular reason. We set our LoRA adapters to be of rank 32 with a LoRA betta of 64. We use gradient accumulation of 6 in order to make training compatible with our hardware resources, together with setting 1024 as the maximum amount of generated tokens for the same reason.

When running the evaluation on the MCQ, we prompted the model to output the answer in a specific manner, so that we could later identify the final result without having to fine-tune the model or add any additional linear layers. We also tried prompting the model to first develop a reasoning for why a specific answer was correct or not and only then give the final result. However, this approach proved less productive than simply asking for the final answer: not only did the inference take much longer because of the quantity of tokens to generate, but the accuracy was actually less by 0.05. Thus, we proceeded to do all the experiments prompting a strict simple answer to each MCQ.

Finally, our training gradually produced 12 different adapters, that we tested individually to pick the best one and further run experiments on it. After running the MCQ evaluation on each checkpoint 5 times, we have concluded that the adapter saved at 10k (10th out of 12 saved) iterations performed the best and was thus our choice.

4.5 Results

4.5.1 DPO training evaluation

The TensorBoard visualizations for the two models reveal a stark contrast in their stability (see Figures 5 and 6 in the Appendix section). In particular, the M3 model demonstrates more stable results, as evidenced by the significant difference between the log probabilities (log probs) of the chosen and rejected outcomes after half of the training time. This larger gap indicates that the model consistently assigns higher probabilities to the correct choices while confidently rejecting the incorrect ones, thereby reflecting robust decision-making and clearer separability in its predictions. In contrast, the M2 model exhibits a smaller difference between these log probs, suggesting less certainty and greater variability in distinguishing between the chosen and rejected outcomes. This comparative analysis underscores the superior stability and reliability of the model trained on the STEM_DPO dataset in its predictive performance.

4.5.2 MCQA accuracy

One common problem we found during evaluation, was the varying accuracy rate of answers. When prompted to solely give the correct answer to the question, we achieved **0.95 CIs of around 0.05** (illustrated in Figure 4), which meant multiple inferences were needed to get a meaningful understanding of the results. Otherwise, another possibility was to have a larger dataset, but considering our resources, this approach was too time-consuming.

Model	Description	Accuracy
M2 model	Model fine-tuned on crowd-sourced dataset	0.714
M2 model quantize	Quantize model fine-tunned on crowdsourced dataset	0.671
M2 model + RAG	Model fine-tuned on crowd-sourced dataset with RAG	0.75

Table 1: Inference results of the model fine-tuned on the crowd-sourced dataset and with the improvements. All inferences have been performed using the same evaluation dataset.

Model	Description	Accuracy
M3 model	Model fine-tuned on STEM-DPO	0.865
M3 model quantize	Quantize model fine-tuned on STEM-DPO	0.763
M3 model + RAG	Model fine-tuned on STEM-DPO with RAG	0.856

Table 2: Inference results of the model fine-tuned on the STEM-DPO dataset from HuggingFace and with the improvements. All inferences have been performed using the same evaluation dataset.

Fine Tuned version: Analysing the results of model fine-tuning performance, we can observe

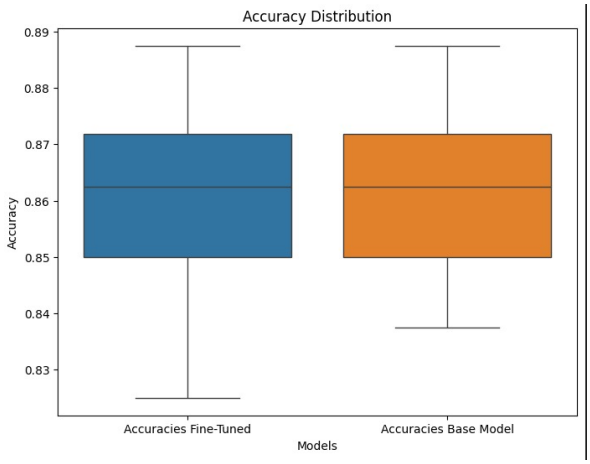

Figure 4: Bar plot of accuracy performance from the M3 fine-tuned model (left) and base model (right) collected from 10 different inference runs. We see that performance is nearly the same if not a bit worse from the m3 model, caused by a larger CI.

that our first try has made the model less performant, while the M3 results yield pretty much the same as the base model performance. This is due to the nature of training and data in use, which we further discuss in the analysis section of the report.

Model quantization: Reducing the model to 4-byte precision does reduce the performance. However, considering the substantial reduction in memory usage and computational load, the models maintained satisfactory accuracy and remained effective in predicting answers to EPFL MCQA questions. This demonstrates that quantization if correctly implemented, can be a viable technique for optimizing resource efficiency without compromising the model's predictive capabilities.

Retrieval-Augmented Generation: Additionally, the integration of RAG contributed to the model's performance enhancement. RAG allows the model to retrieve relevant information dynamically, which is particularly beneficial in complex question-answering scenarios. By accessing and utilizing a broader knowledge base, the model can generate more accurate and contextually appropriate responses. This retrieval mechanism complements the existing strengths of the M2 model, resulting in a noticeable boost in overall accuracy and reliability (see Table 1). For the M3 model, the accuracy remains roughly the same with or without RAG.

5 Analysis

First of all, two main points determine the performance of training: data and the training objective itself. Needless to say, fine-tuning an LLM to answer open questions and MCQ are different tasks, requiring different data and training objectives. Thus, even though our training seems to converge in both situations, we can't be confident about how the model will perform on a downstream task such as MCQ answering. In both, M2 and M3 we can observe the memory loss of the model (see Figures 5 and 6 in the appendix), luckily gravely reduced with the second iteration of training, leaving a similar performance to the base model. In reality, our training was mostly orientated to developing a good explanation of correct answers, which could be evaluated with a more sophisticated pipeline, probably involving direct interaction with humans for reliable feedback. However, we didn't have the time for such experiments in the course of this project.

The training objective and data could have been more effective if the model were prompted to generate longer chains of thought, utilizing information acquired during fine-tuning as well as the knowledge already embedded in the base model. Sadly, as stated earlier, we observe that the model actually underperforms when prompted to answer in such a manner. A possible reason we found was simply the size of the model: even though 8B parameters is a lot, it's still far from models with tens or even hundreds of billions of parameters that are the ones showing SOTA results in the field of reasoning.

We employed prompting strategies and regular expression to identify the key of the "predicted" answer in multiple-choice question answering. While this approach can yield accurate results, it is prone to errors and mispredictions due to the inherent ambiguities and variability in the generated output. These limitations suggest that an alternative method, such as using a classification model, could enhance our system's reliability to answer MCQ. A classification model would offer a more structured and systematic understanding of the generated outputs, enabling us to determine the correct key with greater precision. By leveraging classification techniques or adding a classification header, we can mitigate the risks associated with prompting strategies and achieve more accurate and consistent predictions in MCQA tasks. However, this would come at an additional cost of training and require data,

possibly from the same distribution as the MCQ datasets we use to test performance. Finally such an approach would surly not improve the models capabilities for natural language explanations.

With higher computational resources, we could extensively tune the hyperparameters and conduct inference on a larger test set, greatly enhancing our model's performance. Superior computational power would facilitate more rigorous experimentation, allowing us to explore a wider range of hyperparameter configurations and identify the most effective settings for our model. Additionally, applying the model to a more extensive test set would provide a more comprehensive assessment of its generalization capabilities, ensuring that our findings are robust across diverse data. By leveraging these advanced resources, we can significantly improve the accuracy, stability, and overall efficacy of our model.

Finally, during inference with RAG, we observed that it didn't outperform the trained m3 model. To understand why, we plotted the accuracy of answers across different fields (Figures 8 and 9 in the appendix). We found that RAG underperforms in physics but shows better results in computer science. This discrepancy can be explained by four observations. Firstly, we incorporated more additional data for computer science than for physics when using RAG, enhancing the retrieval process for computer science. Secondly, during text extraction from PDFs to text files, we lost more information from physics books compared to computer science books because physics books contain many graphical explanations that are difficult to (can't be) extract with PYPDF2. Thirdly, answering physics questions may be more challenging due to their format, which often includes a lot of LaTeX and Markdown. Lastly, we did not perform the final step of RAG, which is augmentation. This step is crucial for significantly improving the model using RAG, as it allows for the verification and tuning of the retrieval method. Due to time constraints, we were unable to complete the augmentation.

6 Ethical considerations

One of the primary ethical considerations when developing a large language model (LLM) is addressing bias and discrimination arising from language barriers. Specifically, it is essential to explore how the model can be adapted for both high-resource languages, such as French and German, and low-resource languages like Urdu and Swahili. For high-resource languages, the ample availability of diverse datasets enables effective fine-tuning, ensuring the model captures linguistic nuances. The Llama3 model we use supports several languages, including English, Spanish, French, and more. In contrast, the main challenge with low-resource languages is data scarcity, which can be mitigated through data augmentation techniques, synthetic data generation, or by translating high-resource language data into Swahili or Urdu for exemple.

To interact with users in signed languages, the model must process and generate multimodal data, including video and possibly 3D motion data, by implementing advanced gesture recognition technologies (Gong et al., 2024). Additionally, the model should ensure accessibility for users with hearing impairments by providing both text and sign language outputs.

If our model functions as intended, students would benefit significantly by having a new, highly knowledgeable teaching assistant (TA) always available. The model would offer high accuracy in answering questions, provide additional context and explanations, and simplify students' research on theorems and explanations. Moreover, EPFL would benefit financially by replacing human TAs with the LLM, saving costs.

However, the main losers would be the TAs, who might be replaced by the model, potentially losing a critical source of income that helps them cover essential expenses such as rent and food. Additionally, coursebook writers could be negatively impacted. If our model is trained and fine-tuned on enough coursebook data, it could render new books obsolete by rephrasing and recreating similar problems using synthetic data generation. Furthermore, while the model is designed to assist students in their studies, a future enhancement of the LLM could lead to it replacing teachers by generating voice explanations and teaching theory based on its training.

Several harms associated with our model are more likely to affect individuals who are already members of minority or vulnerable groups. For instance, TAs who rely on their income to meet basic needs might find themselves in a precarious financial situation if replaced by the model. To mitigate this, EPFL could provide alternative employment opportunities or retraining programs for displaced TAs. Additionally, ensuring that the model does

not perpetuate or exacerbate biases present in the training data is crucial. This can be addressed by actively working to identify and mitigate biases during the model development and training process.

Finally, considering the potential misuse of the model, strict ethical guidelines and oversight should be implemented to prevent the model from being used in ways that could harm users, such as spreading misinformation or replacing human educators without sufficient justification and support systems in place.

7 Conclusion

In conclusion, the chosen model, Llama 3 8B Instruct, proved effective in predicting answers to EPFL MCQA questions, with its performance further improving through fine-tuning, specifically with a well-curated dataset.

Integrating Retrieval-Augmented Generation (RAG) enhances model performance, providing more accurate and contextually relevant responses to multiple-choice questions from EPFL exams, being specifically effective in mitigating the effects of model memory loss after training. We also found that quantizing the model to 4-byte precision during training did not substantially impact performance, allowing us to manage computational resources more efficiently. These achievements underscore the potential of RAG and quantization in optimizing large language models for specific tasks.

However, our work faced notable limitations, primarily due to constrained computational resources, which limited the available models and the amount of data we were able to use for training and inference. This restriction may have impacted the overall robustness of our findings. Despite these challenges, the results are promising and provide a solid foundation for future research.

Another fundamental part of this work, that doesn't match the objective, is the data quality. While SOTA LLMs are efficient on multiple benchmarks, STEM seems to be one where they still under-perform. This is explainable by the importance of precision, yet variability of data in this field. Without a further development in LLM architecture, the only way to mitigate this and get an LLM that generalizes well in the STEM field, is by collecting carefully curated, large and high quality datasets: something we didn't have access to during this project.

Exploring different model architectures and more advanced fine-tuning techniques could further optimize performance. Additionally, integrating approaches like active learning or semi-supervised learning could improve efficiency and accuracy in the prediction task. Investigating more sophisticated quantization methods may also offer better resource management without compromising performance. These directions could significantly advance the effectiveness of large language models in handling complex tasks like multiple-choice question answering.

References

Armen Aghajanyan, Luke Zettlemoyer, and Sonal Gupta. 2020. Intrinsic Dimensionality Explains the Effectiveness of Language Model Fine-Tuning. ArXiv:2012.13255 [cs].

Mohammad Gheshlaghi Azar, Mark Rowland, Bilal Piot, Daniel Guo, Daniele Calandriello, Michal Valko, and Rémi Munos. 2023. A General Theoretical Paradigm to Understand Learning from Human Preferences. ArXiv:2310.12036 [cs, stat].

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. 2022. Llm.int8(): 8-bit matrix multiplication for transformers at scale.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. 2024. Retrieval-augmented generation for large language models: A survey.

Jia Gong, Lin Geng Foo, Yixuan He, Hossein Rahmani, and Jun Liu. 2024. LLMs are Good Sign Language Translators. ArXiv:2404.00925 [cs].

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. ArXiv:2106.09685 [cs].

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.

Pulkit Pattnaik, Rishabh Maheshwary, Kelechi Ogueji, Vikas Yadav, and Sathwik Tejaswi Madhusudhan. 2024. Curry-DPO: Enhancing Alignment using Curriculum Learning & Ranked Preferences. ArXiv:2403.07230 [cs].

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:2305.18290 [cs].

A Appendix

A.1 AI policy

We decided against using AI for the main task due to its complexity and the need for precise, nuanced decision-making beyond current AI capabilities. Instead, AI was utilized only for minor data preprocessing tasks and the correctness could be evaluated easily by looking at the output shape.

A.2 Graphs

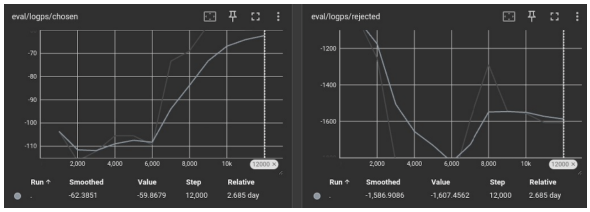

Figure 5: Plots portraying logits given to chosen (left) and rejected (right) samples by the M3 model during training. Chosen samples become more probable starting from 1/2 of the overall training time, while the probabilities of rejected samples go down from the beginning and stay rather low, at least visibly lower than the chosen ones.

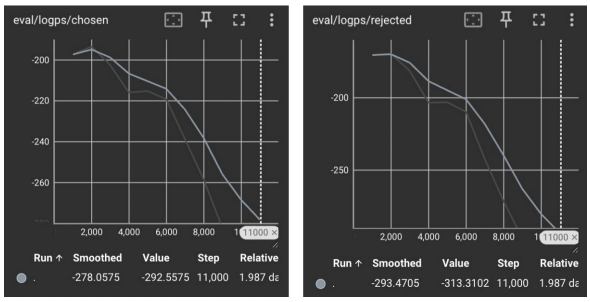

Figure 6: Plots portraying logits given to chosen (left) and rejected (right) samples by the M2 model during training. Chosen and rejected sample's probability stay close to each other during the entire training time and actually go down for both.

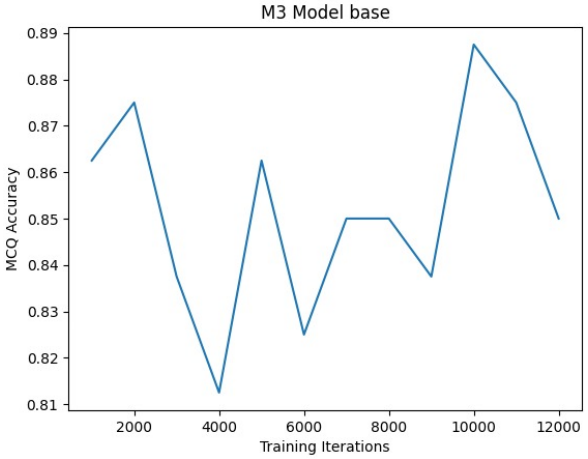

Figure 7: The effect of training iterations on the final result. Out of a few runs, we have remarked that 10k was constantly outperforming the others and thus decided to use it for further experiments

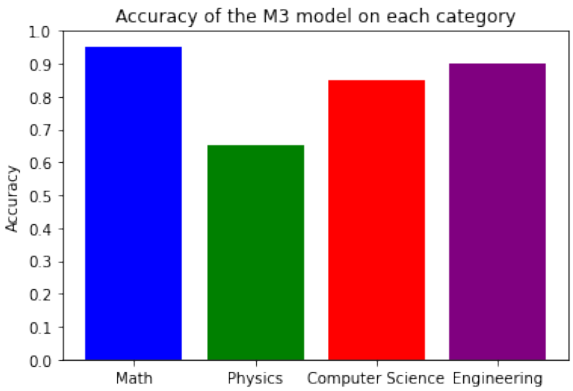

Figure 8: Accuracy of the M3 model for each category of the inference dataset.

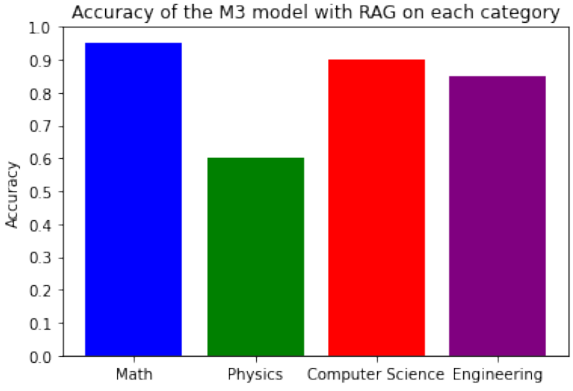

Figure 9: Accuracy of the M3 model with RAG for each category of the inference dataset.

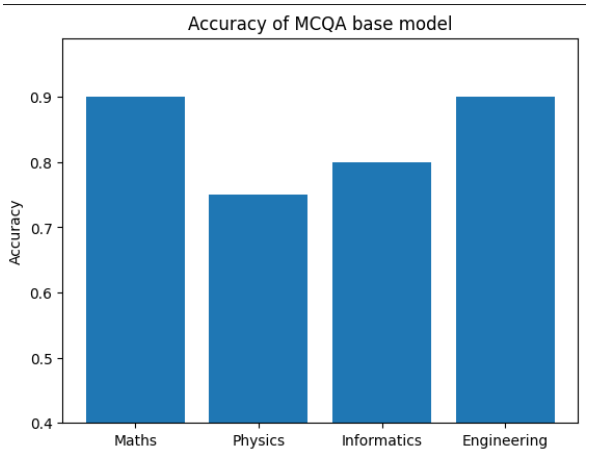

Figure 10: Accuracy of the base model for each category of the inference dataset, based on the results of one training.

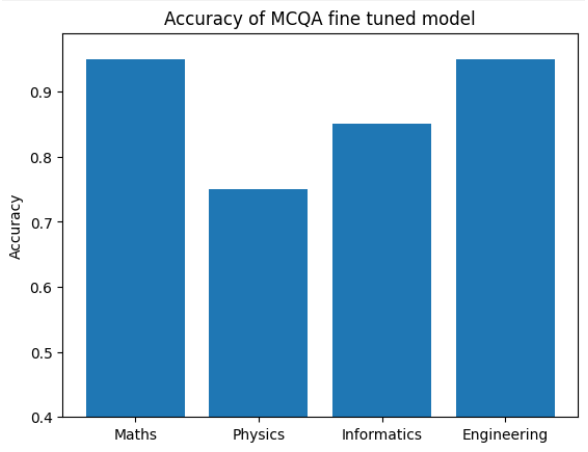

Figure 11: Accuracy of the M3 model for each category of the inference dataset, based on the results of one training.