Final Report: Building an Educational Assistant LLM

Ali Essonni | 316383 | ali.essonni@epfl.ch
Aymeric de chillaz | 326617 | aymeric.dechillaz@epfl.ch
Leila Aissa | 325708 | leila.aissa@epfl.ch
NaturalLanguagePioneers

Abstract

In recent years, large language models (LLMs) have demonstrated remarkable capabilities in terms of natural language processing tasks, yet they often struggle with STEM-related questions, particularly those requiring complex mathematical reasoning, chained reasoning abilities and ability to distinguish nuances. This paper presents an approach to fine-tuning LLMs that enhances their performance in answering multiple-choice questions across STEM fields.

Moreover, we integrate Retrieval-Augmented Generation (RAG) strategies to augment the model's ability to reference relevant information dynamically.

Through extensive experimentation and evaluation, our fine-tuned model demonstrates significant improvements in answering STEM MCQs, offering a promising solution to the limitations of existing LLMs, it also provides a valuable resource for educational purposes supporting both educators and learners achieving better outcome in STEM education.

1 Introduction

The advancements in large language models have revolutionized various aspects of natural language processing, enabling significant progress in tasks such as translation, text generation, and comprehension.
Despite these improvements LLMs continue to face considerable challenges in understanding specific text, for instance mathematical equations, and also in performing tasks that require complex reasoning such as answering STEM-related MCQs that often hide very tricky nuances difficult to notice even for a human.
Addressing these shortcomings is not only a technical challenge but also an educational imperative because students can greatly benefit from automated tutoring systems.

In this paper, we explore different fine-tuning methods for LLMs to improve their ability to solve complex problems related to STEM. Our method builds on the MTSFT and LESFT supervised fine-tuning methods with an extra DPO fine-tuning step (See Related Work for an explanation of these methods).
Furthermore, we incorporate Retrieval-Augmented Generation to enhance the model's performance by giving it access to relevant information as context before answering the MCQ.
This integration allows the model to reference external knowledge bases effectively, thereby improving its ability to resolve nuanced and complex queries that go beyond its pre-existing knowledge.

2 Related Work

Our research builds on the ideas presented in two papers (Mehta and Seetharaman) and (Liu et al., 2023), each addressing key challenges in fine-tuning large language models (LLMs) for complex problem-solving tasks. By extending the methodologies introduced in these works, we aim to enhance the performance of LLMs in answering STEM-related multiple-choice questions (MCQs) through Direct Preference Optimization (DPO) fine-tuning and Retrieval-Augmented Generation (RAG).

Mathematical Reasoning Through LLM Fine-tuning. This paper (Mehta and Seetharaman), explores fine-tuning techniques to improve LLMs' logical reasoning capabilities in solving mathematical problems. They implemented Multi-Task Sequential Fine-Tuning (MTSFT), which involves sequentially training models as solution evaluators and generators, and Logic-Enhanced Sequential Fine-Tuning (LESFT), which includes additional fine-tuning on the LogiQA dataset to enhance logical reasoning. Their approach significantly improved performance on complex math problems, highlighting the importance of structured

fine-tuning and logical reasoning datasets.

Improving Large Language Model Fine-Tuning for Solving Math Problems. This paper by (Liu et al., 2023), investigates fine-tuning strategies such as supervised step-by-step solution fine-tuning (SSFT), solution-cluster re-ranking (SCR), and MTSFT. They found that combining solution re-ranking with majority voting and sequential fine-tuning of solution generation and evaluation tasks significantly enhances performance. This study emphasized the importance of high-quality, well-structured fine-tuning datasets and multi-task learning.

The two papers are closely related in their exploration of fine-tuning techniques to improve LLM performance on mathematical tasks. Both papers highlight the need for sequential fine-tuning approaches that combine solution generation and evaluation. Mehta and Seetharaman's work on LESFT builds on the ideas of sequential training by incorporating logical reasoning datasets, while Liu et al. focus on multi-task learning and the integration of solution-cluster re-ranking to enhance model accuracy. Together, these papers provide a comprehensive framework for improving the logical and mathematical reasoning capabilities of LLMs through structured fine-tuning.

Building on these methodologies, we extend the fine-tuning techniques by incorporating Direct Preference Optimization (DPO) and Retrieval-Augmented Generation (RAG). DPO fine-tuning optimizes the model based on direct feedback and preferences, allowing for nuanced improvements. By integrating preference-based annotations, the model learns from detailed human feedback, refining its ability to produce accurate and relevant answers.

We also implement RAG strategies to enhance the model's capabilities dynamically. Classical RAG retrieves relevant information from a knowledge base during question answering, while RAG fusion integrates multiple retrieval sources and fuses the information into the model's generation process. These enhancements address LLMs' weaknesses in handling complex mathematical reasoning and logical deductions, crucial for accurately answering STEM-related MCQs.

3 Approach

The model we use is the Phi-3-Mini-128K-Instruct, a 3.8 billion-parameter decoder-only transformer, state-of-the-art (fewer than 8 billion parameters) open model trained on 3.3 trillion tokens. We chose this model for its performance and compatibility with a single V100 GPU without excessive quantization. It serves as the foundation for our fine-tuning efforts due to its optimal balance between computational efficiency and performance, it is designed to deliver robust NLP capabilities while maintaining a manageable size. This model is very relevant to educational applications since its small size means it can be deployed on various platforms, including low-powered devices, and it has a strong reasoning capability which is needed in our specific use-case.
We explore two versions of this model to enable various training techniques and enable more thorough research. The standard 3.8B parameters phi-3 model is trained using QLoRA and a variant pruned down to 2.1B parameters for LoftQ.
Our fine-tuning method is two-folds, we first fine-tune the model using supervised solution fine-tuning, then we perform DPO fine-tuning on the same model.

3.1 Supervised Solution Fine-Tuning (SSFT)

SSFT is about fine-tuning a large language model M to generate a step by step solution S and an answer A given a problem P. In order to perform SSFT, we typically have a dataset of instruction prompts and desired output that the model should learn from. In practice, the probability of the model outputting the i^{th} token of the concatenation: $X = S||A$ is given by:

$$p_M(X|P) = \prod_i p_M(x_i|X_{0,\ldots,i-1}, P) \quad (1)$$

With this fine-tuning layer, we aim at improving the model's reasoning capabilities by specializing it using a dataset of STEM questions and answers, this is done in order to have a strong basis on which we can then perform DPO. The idea of performing supervised fine-tuning before DPO is backed by the results of the following paper: (Saeidi et al., 2024)

3.2 Direct Preference Optimization (DPO)

DPO is a very competitive fine-tuning method that is easier to set up than PPO (Proximal Policy Optimization). The paper (Rafailov et al., 2023) argued

that we can use the same model and different completions to achieve the same results.

$$\mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}\left[\log \sigma\left(\beta\left(\log \frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \log \frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)}\right)\right)\right] \tag{2}$$

- π_θ: The policy being learned, parameterized by θ.
- π_{ref}: The reference policy against which the new policy is compared.
- β: A scaling factor that adjusts the sensitivity of the comparison between the log ratios.

We aim at optimizing the above loss function which compares the policy probabilities of the old policy and the new policy across a winning completion and a losing completion, we are then naturally optimizing so that the probabilities of the winning completions are the highest.

3.3 RAG

RAG is a way of improving the accuracy and reliability of a model's output by relying on external sources (Lewis et al., 2021).
We implemented two variants of Retrieval-Augmented Generation (RAG) to enhance the model's capabilities in answering STEM-related multiple-choice questions (MCQs): classical RAG and RAG fusion.

3.3.1 Classical RAG Implementation

Classical RAG is designed to augment the model's response generation by retrieving relevant information from external knowledge bases during the question-answering process. The core idea is to enhance the model's ability to generate accurate answers by providing it with additional context that it may not inherently possess.
We utilized two primary knowledge sources: Wikipedia and Arxiv Math datasets (see section 4.1.3), chosen for their comprehensive coverage of general and specialized knowledge relevant to STEM fields.
To ensure the context provided was relevant, we split the data into chunks, we then embedded these chunks of text in a vector space using a specialized language model, finally we used cosine similarity between embeddings to determine the relevance of each chunk to the question and only keep the most relevant chunks as context.
For each dataset, we tuned the number of samples to provide to our model, ensuring a balance between providing enough context and maintaining prompt coherence.
The retrieved documents were then incorporated into the prompt (See A.4) provided to the model. The prompt template included sections for context, related questions, and the main question, structured to help the model utilize the retrieved information effectively. The model then generated an answer, considering the additional information provided.

3.3.2 RAG Fusion Implementation

RAG Fusion is an advanced RAG technique (Rackauckas, 2024) that leverages language models' ability to reason in order to ask meaningful questions that could guide the retrieval part. It extends the classical RAG approach by integrating multiple retrieval sources and ranking the related context.
Initially, multiple questions related to the main question are generated using a potentially smaller model. In our case, we keep our model. because we are satisfied with its generations. This approach generates diverse questions that might be good to answer before tackling the main one, ensuring a broad coverage of the relevant topic areas.
Once these related questions are generated, we used cosine similarity to find the most similar data samples from our knowledge bases. The samples were ranked by cumulative relevance for each question, ensuring the highest quality and most pertinent information was included.
The reranking algorithm used is *Reciprocal Rank Fusion* where each document is assigned:

$$\text{rrf} = (\text{rank} + k)^{-1} \tag{3}$$

where the rank is the current rank of the documents sorted by distance relative to a question, and k is a constant smoothing factor that determines the weight given to the existing ranks. The rrf scores are then accumulated across all questions. The documents with the highest scores are then inserted in the prompt.

3.3.3 Embeddings and retrieval method

To encode our context, we selected the open-source model with fewer than 1 billion parameters that performs best on the Massive Text Embedding Benchmark (Muennighoff et al., 2023). Consequently, we chose the Alibaba-NLP gte-large-en-v1.5 model (Li et al., 2023b).
We noticed that our text chunks were semantically very similar, which is expected since all the documents from which they originate are related to STEM. Consequently, the chunks are clustered in a very restricted area of the vector space.

This clustering results in poor RAG performance because its retrieval method relies on cosine similarity between documents, and the high similarity between clustered vectors diminishes its effectiveness.

To address the sparsely distributed embeddings, we standardized the samples: centering our space around the origin to enable a broader range of high and low similarities.

$$z_{ji} = \frac{x_{ji} - \mu_i}{\sigma_i} \tag{4}$$

For efficient cosine similarity throughout the database, we used instances of the ScaNN library (Guo et al., 2020) for each dataset.

3.4 MCQ Specialization

Once the prompt is ready (See Appendix Model Prompt With & Without RAG) the model is asked to give its prediction. We found that the model extracts more value by having a template where we concatenate samples with clear delimiters (Context, and Question/Answer). Then to get an answer to our question we test two methods. The first approach is to ask the model to respond first with its prediction, so that we may parse for A,B,C or D using a simple regex. In practice we find that this method works all the time on our evaluation data: however, this method can fail if the model decides to output anything else. We need a method that guarantees that we find the model's prediction. Second, instead of generating a response, we can simply have a single forward pass over our prompt to do a next token prediction, then check the probability of the token of each letter. Finally, we choose the letter with the highest probability of appearing. We prefer the latter strategy as it guarantees a better generalization to other datasets and is also more efficient since it only requires a single forward pass to generate a single token rather than a complete answer.

3.5 Global Overview

Our method consists of testing different combinations of fine-tunings and RAG in order to find the best-performing combination, but also the less costly combination in terms of resources and generation time.

We hypothesize that fine-tuning our base model using SSFT and DPO will yield the most significant performance improvements, with the addition of RAG further enhancing the capabilities of this trained math specialist.

In the following section, we describe the experiments conducted to test out this hypothesis.

4 Experiments

4.1 Data

4.1.1 DPO data for QLORA model

Preference pairs collection We gathered a comprehensive dataset based on exercise sheets and exams from EPFL. The process involved 300 students, each assigned a set of 100 questions. The student's task was to determine relevant prompting strategies to generate answers for these questions using GPT-3.5. Each student was then asked to manually annotate the chosen and rejected answers based on several criteria: factual correctness, relevance, clarity, and completeness.

This resulted in a substantial dataset of approximately 30,000 samples, providing a rich source of diverse annotated responses for fine-tuning. We used diverse prompting strategies as explained in A.2.

4.1.2 Finetuning and DPO data for LOFTQ model

Supervised Solution Fine-tuning For the supervised fine-tuning phase, we utilized the GarageBAInd/Open-Platypus dataset (Lee et al., 2024), which gathers high-quality instructions and outputs from several well-known STEM datasets. These include PRM800K, MATH, ScienceQA, ReClor, TheoremQA, and others. The dataset comprises multiple questions and answers, each answer is very detailed and presents all the steps of the thought process used to answer the question.

To ensure efficiency and manageability during training, we filtered out responses and questions exceeding 1024 tokens. This preprocessing step ensured that the model could process the data within its token limits without truncation. The final dataset consisted of a substantial number of samples from each source as shown in Table 3. In total having 24,902 high quality question answer pairs.

DPO We use the Argilla ultrafeedback-binarized-preferences-cleaned dataset (Bartolome et al., 2023), which is widely used and robustly labeled. Indeed, this dataset with 60K pairs results from thorough work such as the decontamination of the TruthfulQA dataset. We pruned the non-scientific samples with labels we generated with

the Phi-3 model, yielding 13K samples. We use this dataset for the DPO training.
This diverse and comprehensive dataset provided a robust foundation for training the model, ensuring exposure to a wide range of STEM problems and solutions.

4.1.3 Dataset for RAG

For both models, we implement RAG similarly using 2 datasets:
Wikipedia Dataset (Foundation) Wikipedia is the most comprehensive knowledge base available, making it an ideal source for extracting pages related to our scientific topics of interest. First, we reviewed all Wikipedia categories and selected the relevant subset. Using the PetScan tool, we extracted the IDs of all Wikipedia pages associated with these categories up to a depth of 2: where depth 1 includes direct links, and depth 2 includes indirect links. This process resulted in approximately 150,000 pages, which we then extracted from the HuggingFace Wikipedia dataset. These pages were then split into chunks. We found 300 words to be a good balance to avoid having to store too many embeddings, while not including too much context.
Arxiv MATH (Kenney, 2023) This dataset consists of question-answer pairs in STEM-related fields: Math, computer science and Physics derived from ArXiv abstracts. We preprocess this dataset based on the samples' length, to avoid keeping overly long samples as they might cause memory issues at inference time.
Overall we have a total of 0.914 GB with 0.81 GB for Wikipedia data and 0.104 GB for Arxiv data.

4.2 Evaluation method

DPO Evaluation. Following DPO training, we evaluated our two policy models, QLoRA and LoftQ, in comparison to our reference model, phi-3-mini-128K. To gauge the improvements, we utilized 10% of the data outlined in Section Finetuning and DPO data for LOFTQ model and calculated the accuracy with which each policy model ranks preferences relative to the reference model.

Seeing QLoRA's poor DPO results in Table 1 we wondered if our training had severely hindered its performance. So, we chose to assess the model's performance on a comprehensive set of math and science benchmarks (See Table 4).
We test our models on various benchmarks. For this, we utilize a GitHub repository (Gao et al., 2023) containing many of the benchmarks of interest, ensuring identical benchmark settings. This enabled us to compare our model's performance with the sub-10-billion parameter math-state-of-the-art (April 2024) model Rho-Math (Lin et al., 2024) and LLama2 (Touvron et al., 2023). (See Results in 4.5)

RAG Evaluation. To evaluate our RAG models, we required a dataset of scientific MCQs with highly reliable answers. We selected relevant topics from the widely used MMLU benchmark (Hendrycks et al., 2021), specifically focusing on mathematics, physics, and electrical and computer sciences.

Using the 245-sample evaluation dataset, we conducted a grid-search hyperparameter tuning to determine the optimal number of Wikipedia chunks and arxiv Q&As to include in the context for maximizing performance. Varying both parameters from 0 to 7 produced a comprehensive heatmap overview of our model performance. We explore these two ranges because they won't be longer than our max prompt length, which we found experimentally to be 7808. Note that setting both parameters to 0 provides the non-RAG-enhanced performance, allowing us to observe the improvements introduced by RAG. This evaluation was conducted on both our trained models and the baseline for comparison purposes. The RAG scores are displayed in Figure 1.

4.3 Baselines

Throughout our research, we consistently used the 4-bit quantization of the Phi-3-mini-128K as our baseline model. This choice was made because the Phi-3-mini-128K is not only the foundational model we fine-tuned, but also a robust model known for its reliability. The 4-bit quantization was essential for running evaluations and aligning with the quantization of our trained models. We also conducted comparisons with other state-of-the-art (SOTA) models on general math benchmarks (see Table 4). Additionally, we compared the performance of our two trained models: LoftQ and QLorRA.

4.4 Experimental details

4.4.1 QLORA model

DPO training For the DPO task, we were cautious about using the collectively collected data because of the mislabeled samples we found, as well as the

absence of validation in previous work. Despite these issues, the provided questions were closely aligned with our downstream task of providing answers to EPFL MCQs. As such, we decided that extracting only the MCQs from the preference pairs was a common ground which we were comfortable with. Along with those MCQs we used the dataset discussed in section 4.1.2.

Due to the high memory requirements of the default Adam optimizer, we found an alternative: initially AdamW bnb 8bit. Unlike AdamW, which uses more than 8 bytes per parameter, this version uses only 2 bytes, allowing us to train on the V100 GPU at the cost of training efficiency. We did not want to pay this cost so we used adamw_torch_fused as it leads to better performance than AdamW, and used LoRA (Hu et al., 2021) on a 4-bit quantized model to fit in memory.

4.4.2 LOFTQ model

Model Specifications: For this fine-tuning task, we employed the Phi-3 mini model from LoftQ (LoRA-Fine-Tuning-Aware Quantization). LoftQ is a library that offers fine-tuning-aware quantized models (Li et al., 2023a), enhancing training efficiency compared to standard QLoRa models as explained in the paper. The specific base model used was LoftQ/Phi-3-mini-128k-instruct-4bit-64rank, derived from the Phi-3-mini-128k-instruct model that has been pruned down to 2.1B parameters. This model integrates a quantized backbone with LoRA adapters, facilitating efficient training and performance.

This approach is particularly suitable for our objectives of optimizing computational efficiency without sacrificing performance mainly under the limiting resources we have.

Training Process: The training process was managed using the SFTTrainer module from the `trl` library (von Werra et al., 2020), which streamlined the training operations. Key training parameters included:

- **Epochs and Batch Size**: The model was trained for three epochs with a batch size of one per device for both training and evaluation.
- **Gradient Accumulation**: Steps were accumulated to ensure effective learning despite the small batch size.
- **Optimizer and Learning Rate**: The AdamW optimizer with torch fusion was used, and the learning rate was set to 5e-6.
- **Gradient Clipping and Warmup**: Gradient clipping was set at a max norm of 0.3, and a warmup ratio of 0.1 was employed to stabilize training.
- **Learning Rate Scheduler**: A cosine learning rate scheduler was used to adjust the learning rate dynamically.
- **Precision Settings**: Bfloat16 and TF32 precision were enabled to reduce memory usage and accelerate training.

These parameters were carefully chosen to optimize the training process, ensuring efficient use of computational resources while achieving high performance. The use of advanced precision settings and optimized learning strategies contributed to the model's enhanced capability in handling complex STEM-related MCQs.

4.5 Results

DPO. The results of DPO evaluation, presented in Table 1, indicate that LoftQ predicts more accurately in 54.9% of the cases, demonstrating a slight improvement during training. Conversely, QLoRA shows a performance decrease, accurately predicting only 39.9% of the time.

	Policy Reward Accuracy (%)
LOFTQ Model	54.9
QLORA Model	39.9

Table 1: Policy Reward Accuracy Comparison with Phi-3-mini-128K as Reference Model

The Benchmark results upon DPO training with QLoRA(See Table 4) reveal that on average our model is better than all baselines (we didn't consider Llama-3-8B because of memory constraints). It performs significantly better than our base model in general math like Minerva Math (Lewkowycz et al., 2022) and TABMWP (Lu et al., 2023). Also, for MMLU STEM (Hendrycks et al., 2021) which is the benchmark of MCQS that resembles the most our downstream task we get a significant improvement over our baseline. However, there are benchmarks such as MATHQA (Amini et al., 2019) where our model seems to have gotten worse, which is a shame as it is another instance of an MCQ benchmark.

Results RAG. Table 2 reveals significant performance gains with RAG whether it be for our base model or our QLoRA and LoftQ models. Interestingly, we can see that QLoRA performs the best.

	QLORA	LOFTQ
Policy accuracy (%)	42.8	XXX
RAG policy accuracy (%)	51.4	XXX

Table 2: Policy RAG Accuracy Comparison

It is evident that RAG yields significant improvements. However, we are also interested in understanding whether our trained models outperform the base model. In Figure 1 we see that the LoftQ scores are significantly lower, with the best score achieved being 114/245, compared to QLoRA's 120/245 and Phi-3's 119/245.

5 Analysis

DPO Analysis: All-in-all, it seems that the QLoRA model didn't improve at ranking preference pairs with an accuracy of 39.9%(See Table 1), yet it got better on the general math benchmarks (See Table 4).
We suspect that this drop in DPO performance is due to the quality of the preference pairs gathered from the class. In fact, (Morimura et al., 2024) highlights the impact of data quality. This is why we decided in the subsequent fine-tuning of the LoftQ Phi-3 to omit the class's preference pairs and only perform DPO using the Argilla dataset. This change resulted in a DPO accuracy of 54.9% revealing a slight improvement.

RAG Analysis: The heatmaps in Figure 1 yield many findings. Firstly, the Phi-3 and QLoRA models perform very similarly across various hyperparameters, indicating that our training was not very effective. Secondly, a trend emerges from these results: the scores are most correlated with the number of Wikipedia chunks, indicating that this is the most crucial component of our RAG system. Conversely, the Arxiv data does not seem to add much value. This raises the question of whether the data quality is lacking or if inserting shots into our prompt is not beneficial for this task. Thirdly, a performance peak around 5 context chunks underlines the importance of relevance when choosing samples. The LOFTQ model shows less sensitivity to hyperparameter changes indicating less effective utilization of additional context. Overall, we observe that RAG yields performance improvements, which is also backed up by Table 2. However, the value of fine-tuning the model is not clear.
By integrating multiple sources and dynamically selecting relevant information, we hoped that RAG Fusion would provide a richer and more informative context for the model, significantly enhancing its response accuracy and relevance. However, that was far from the truth. Indeed, we tried multiple hyperparameters yet non yielded significant improvements over the no RAG baseline. This could be because we didn't find the right hyperparameter; however, we performed extensive hyperparameter-tuning. Alternatively, it would be interesting to try with smaller chunks to see whether RAG Fusion functions best with fine-grain context.

Understanding the Insignificant Finetuning Improvements: We hypothesize that the poor finetuning results are driven by the extensive training of our base model, Phi-3-mini-128K. This is evidenced by its deviation from the Chinchilla Scaling Laws (Hoffmann et al., 2022). Using the common approximation $C = 6ND$ (validated in the Chinchilla paper), where C is the amount of training FLOPs, D is the number of training tokens, and N is the number of parameters, we can compute and plot the FLOPs for Phi-3. With $D = 3.3 \times 10^{12}$ and $N = 3.8 \times 10^{9}$, we get $C = 7.5 \times 10^{22}$.
As shown in Figure 2 our base model is indeed very much to the right of the Chinchilla scaling law. The scaling law tells us that for a model with 3.8B parameters, the optimal amount of FLOPs is about 10 times less than the training that Phi-3 has undergone. In fact, for $C = 7.5 \times 10^{22}$ FLOPs, the optimal model size would be about 25B. The more you train a model, the lower the performance gains. We made sure to have qualitative data; however, the training we underwent with a single V100 GPU over a few days is quasi-insignificant.

6 Ethical considerations

Our educational LLM is strictly fine-tuned on English data which could cause significant ethical considerations regarding linguistic inclusivity. Adapting our model to high-resource languages like French or German is indeed straightforward and could be considered for future work. However, extending the model to low-resource languages can be very challenging and it would be necessary first to invest in collecting meaningful datasets.
One could also argue the importance of enabling interaction with users in signed language. This is indeed crucial for giving equal opportunities to users mainly in education context. In future work, we could consider integrating computer vision techniques and real-time video processing to interpret

signed language input. The output could be kept as text or we may even consider speech synthesis.

7 Conclusion

In this project, we developed an educational LLM to answer multiple-choice questions related to university-level scientific courses. We used the Phi-3-mini as our base model, employing multiple layers of fine-tuning, DPO, and RAG techniques. Our goal was to achieve optimal performance within the constraints of our limited resources by employing diverse optimization strategies. Our two trained models yielded intriguing results, suggesting that further in-depth experiments could enhance performance.

We were unable to achieve significant performance gains through fine-tuning, likely due to our base model being trained 10 times more than the Chinchilla Scaling Laws recommend, coupled with the limited fine-tuning we could perform using a single GPU and constrained time. However, our base RAG implementation did yield substantial performance improvements, despite the RAG Fusion not retrieving highly relevant documents. Our findings highlighted that in LLM training, data quality is far more critical than quantity. During DPO training, eliminating the lower-quality half of the dataset resulted in better training performance. With RAG, we found that carefully selecting relevant chunks, rather than overwhelming the model with somewhat relevant data, was the most effective approach.

For future work, we plan to use more pertinent data for RAG. Students typically study with textbooks and exercise sets rather than Wikipedia, and through our university, we have access to a wealth of such resources. It would be interesting to evaluate the performance gains using these tools.

References

Aida Amini, Saadia Gabriel, Peter Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. 2019. Mathqa: Towards interpretable math word problem solving with operation-based formalisms.

Alvaro Bartolome, Gabriel Martin, and Daniel Vila. 2023. Notus. https://github.com/argilla-io/notus.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems.

Wikimedia Foundation. Wikimedia downloads.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2023. A framework for few-shot language model evaluation.

Ruiqi Guo, Philip Sun, Erik Lindgren, Quan Geng, David Simcha, Felix Chern, and Sanjiv Kumar. 2020. Accelerating large-scale inference with anisotropic vector quantization. In *International Conference on Machine Learning*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. 2022. Training compute-optimal large language models.

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.

Matthew Kenney. 2023. arxiv-math-instruct-50.

Rik Koncel-Kedziorski, Subhro Roy, Aida Amini, Nate Kushman, and Hannaneh Hajishirzi. 2016. MAWPS: A math word problem repository. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1152–1157, San Diego, California. Association for Computational Linguistics.

Ariel N. Lee, Cole J. Hunter, and Nataniel Ruiz. 2024. Platypus: Quick, cheap, and powerful refinement of llms.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2021. Retrieval-augmented generation for knowledge-intensive nlp tasks.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo

Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. 2022. Solving quantitative reasoning problems with language models.

Yixiao Li, Yifan Yu, Chen Liang, Pengcheng He, Nikos Karampatziakis, Weizhu Chen, and Tuo Zhao. 2023a. Loftq: Lora-fine-tuning-aware quantization for large language models.

Zehan Li, Xin Zhang, Yanzhao Zhang, Dingkun Long, Pengjun Xie, and Meishan Zhang. 2023b. Towards general text embeddings with multi-stage contrastive learning.

Zhenghao Lin, Zhibin Gou, Yeyun Gong, Xiao Liu, Yelong Shen, Ruochen Xu, Chen Lin, Yujiu Yang, Jian Jiao, Nan Duan, and Weizhu Chen. 2024. Rho-1: Not all tokens are what you need.

Yixin Liu, Avi Singh, C. Daniel Freeman, John D. Co-Reyes, and Peter J. Liu. 2023. Improving large language model fine-tuning for solving math problems.

Pan Lu, Liang Qiu, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, Tanmay Rajpurohit, Peter Clark, and Ashwin Kalyan. 2023. Dynamic prompt learning via policy gradient for semi-structured mathematical reasoning.

Yash Mehta and Karthik Seetharaman. Mathematical reasoning through llm finetuning.

Shen-Yun Miao, Chao-Chun Liang, and Keh-Yih Su. 2021. A diverse corpus for evaluating and developing english math word problem solvers.

Tetsuro Morimura, Mitsuki Sakamoto, Yuu Jinnai, Kenshi Abe, and Kaito Ariu. 2024. Filtered direct preference optimization.

Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. 2023. Mteb: Massive text embedding benchmark.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are nlp models really able to solve simple math word problems?

Zackary Rackauckas. 2024. Rag-fusion: A new take on retrieval augmented generation. *International Journal on Natural Language Computing*, 13(1):37–47.

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.

Amir Saeidi, Shivanshu Verma, and Chitta Baral. 2024. Insights into alignment: Evaluating dpo and its variants across multiple tasks.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models.

Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, and Shengyi Huang. 2020. Trl: Transformer reinforcement learning. `https://github.com/huggingface/trl`.

A Appendix (optional)

A.1 Model Specifications

- **LoRA (Low-Rank Adaptation)**: A technique to efficiently fine-tune large language models by adding low-rank matrices to the original model weights, significantly reducing the number of trainable parameters.

- **QLoRa (Quantized LoRA)**: An extension of LoRA that involves quantizing the model's weights to reduce memory usage and computational requirements further, making the fine-tuning process more efficient.

- **Rank**: In the context of LoRA, the rank refers to the size of the low-rank matrices added to the model. A higher rank allows for capturing more information but increases computational load.

- **Quantization**: A process that reduces the precision of the model's weights, which decreases memory usage and speeds up computation while aiming to maintain model performance.

A.2 Preference pairs collection: prompting strategies

The prompting strategies in our group were varied, we explain some of them in the following:

- **Strategy 1**: We first prompt the Model to give an answer to the question nudging it to give all steps of its thought process by adding `let's think step by step` to the prompt. We then prompt it to summarize the answer keeping only key points.
- **Strategy 2**: We provide the model with instructions to guide its behavior when answering the query. Moreover, depending on the course we indicate the field of that question. An example of instruction is `You are an instructor for university students with strong expertise in analysis. You will be asked to answer questions related to your field of expertise.` Also, we use diverse prompts in order to get high-quality data.

A.3 Dataset for SSFT

Source	Samples
TheoremQA	564
ReClor	4530
Airoboros	2589
Guanaco	790
SciBench	616
Tigerbot-Kaggle	386
ARB	713
LeetCode_NE	1100
MATH/PRM-800K	12,297
ScienceQA	1317

Table 3: Distribution of samples across sources

A.4 Model Prompt With & Without RAG

Prompt

<s><|user|>
You are being provided with context and a question. Try to find out the answer to the question using the context information and what you know.

{related_questions_part}
{examples_part}
{context_part}

Question: {question}
Use the examples and context that we provided to answer the question.
Answer a single letter from [A, B, C, D], then you may explain your reasoning.
Answer:<|end|>
<|assistant|>

A.5 Benchmark Datasets

See Table 4

A.6 RAG Performance

See Figure 1

A.7 Chinchilla Scaling Laws

See Figure 2

Model Performance on Math Tasks

Model	Size	GSM8K	Minerva Math	SVAMP	ASDiv	MAWPS	TabMWP	MATHQA	MMLU STEM	AVG
LLaMA-2	7B	14.0	3.6	39.5	51.7	63.5	30.9	12.4	32.7	31.0
Mistral	7B	41.2	11.6	64.7	68.5	87.5	52.9	33.0	49.5	51.1
DSM*	7B	64.1	**34.2**	74.0	83.9	92.4	**63.4**	62.4	56.4	66.4
Rho-Math	7B	66.9	31.0	77.8	79.0	93.9	49.9	58.7	54.6	64.0
Phi-3 128K*	3.8B	81.7	29.6	89.9	**91.0**	**97.1**	60.7	**69.3**	64.4	73.0
Our QLoRA	3.8B	**82.1**	33.2	**90.6**	**91.0**	97.0	62.8	68.3	**66.3**	**73.9**

Table 4: GSM8K: (Cobbe et al., 2021), Minerva Math: (Lewkowycz et al., 2022), SVAMP: (Patel et al., 2021), ASDiv: (Miao et al., 2021), MAWPS: (Koncel-Kedziorski et al., 2016), TabMWP: (Lu et al., 2023), MATHQA: (Amini et al., 2019), MMLU STEM: (Hendrycks et al., 2021)

Note 1: DSM*: DeepSeekMath
Note 2: Phi-3 128K was tested with its 4-Bit quantization.
Note 3: All these benchmarks were run with the same prompts & code.

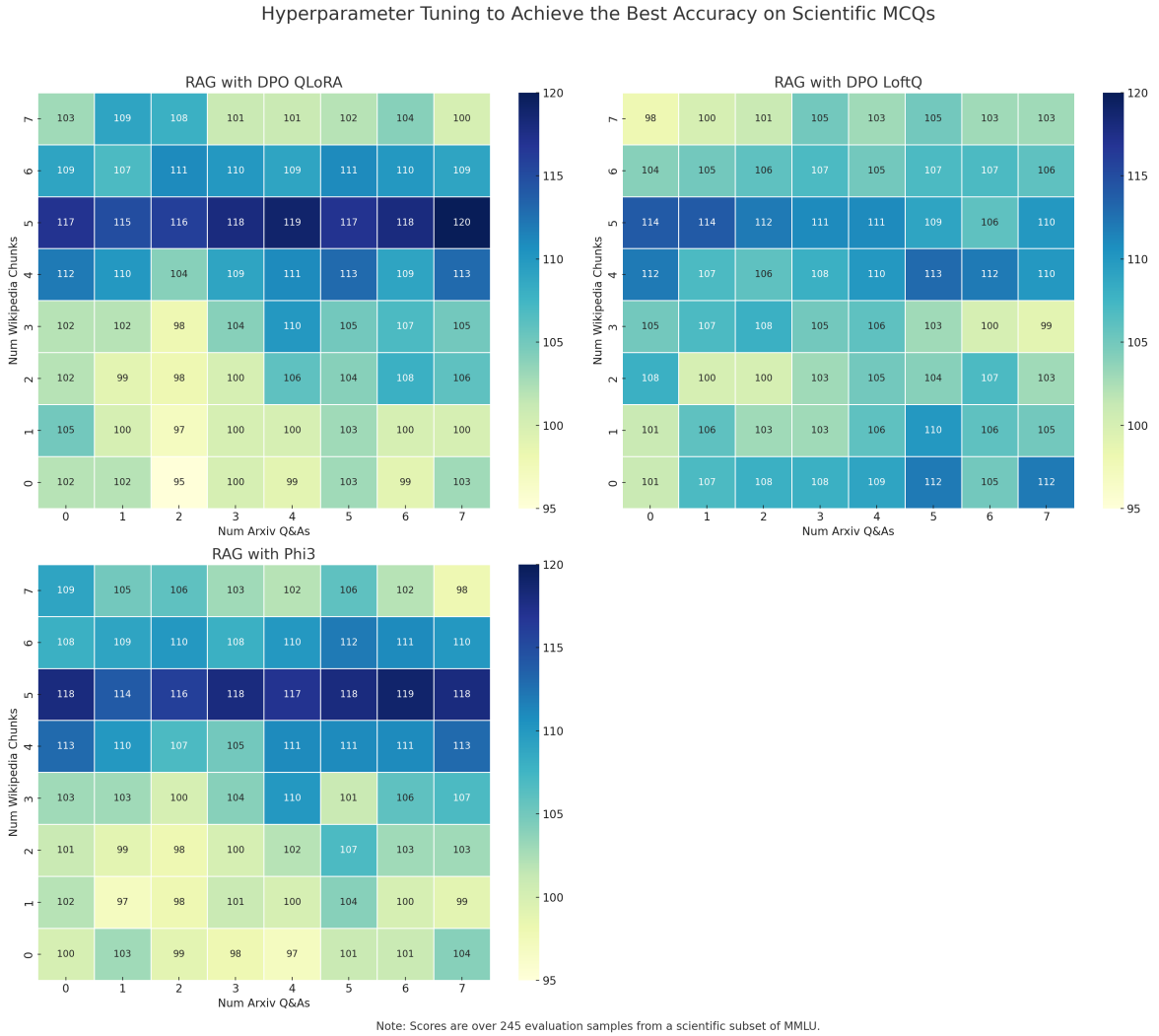

Figure 1: RAG performance comparison with our two trained models, QLoRA & LoftQ, and our base model Phi-3-mini-128K. Each row corresponds to the number of Wikipedia chunks provided as context, and the columns to the number of Arxiv Questions&Answers inserted in the prompt. Note that the (0,0) corresponds to running without RAG. Scores are over 245 evaluation samples from a scientific subset of MMLU.

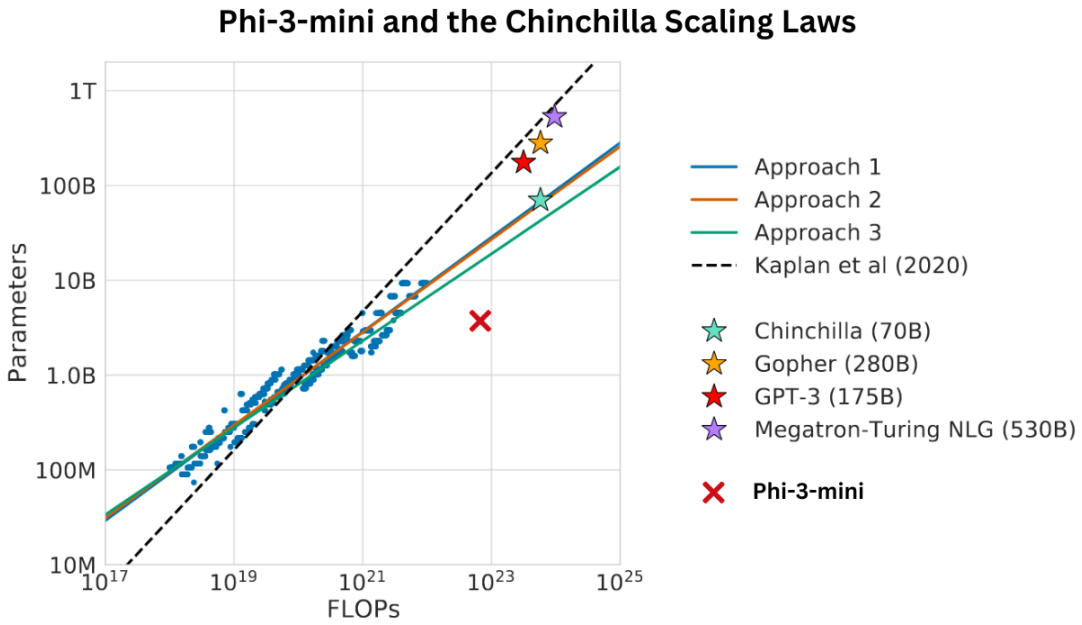

Figure 2: Plotting the phi-3-mini-128K on the Chinchilla Scaling Laws reveals how overtrained the model is, giving a potential explanation for the little improvements achieved through finetuning.