EPFLLaMA - A Lightweight LLM Finetuned on EPFL Curriculum

Ali Bakly | 383057 | ali.bakly@epfl.ch
Elias Ulf Hörnberg | 384928 | elias.hornberg@epfl.ch
Othmane Sqalli Houssaini | 246132 | othmane.sqallihoussaini@epfl.ch
AB-EH-ME

Abstract

Large language models (LLMs) excel in generating human-like responses but face challenges in specialized tasks like answering university-level course questions. This project enhances the TinyLlama model through Supervised Fine-Tuning (SFT) and Direct Preference Optimization (DPO), utilizing datasets from student annotations and Stack Exchange. A key innovation of our approach is the specialization for Multiple-Choice Question Answering (MCQA) related to EPFL course material, optimizing the model to predict single-letter answers accurately. Additionally, we employed quantization techniques to reduce the model's memory footprint without sacrificing performance. Our results demonstrate that the fine-tuned TinyLlama model surpasses established baselines in MCQA tasks, showing robust performance across various technical subjects. This project demonstrates how open-source LLMs can be effectively adapted for educational purposes, providing accurate and context-aware responses to complex academic queries.

1 Introduction

Large language models (LLMs) have made impressive advancements in generating human-like responses and tackling diverse topics. However, fine-tuning these models for specific applications, such as answering university-level course questions, remains challenging. This difficulty stems from the need for deep domain-specific knowledge, the complexity of academic content, and the requirement for accurate, contextually relevant answers (AI4Science and Quantum, 2023).

Existing methods like supervised fine-tuning (SFT) and reinforcement learning often struggle to balance general language understanding with specialized knowledge. Additionally, techniques such as Chain-of-Thought prompting have shown that generating intermediate reasoning steps can significantly improve the ability of LLMs to perform complex reasoning tasks, highlighting the need for sophisticated approaches in educational contexts (Wei et al., 2023). This is where our project comes in. We address these challenges by enhancing the TinyLlama model through a dual-phase training process: SFT and Direct Preference Optimization (DPO) (Rafailov et al., 2023) while incorporating Chain-of-Thought prompting. We leverage diverse datasets, including student-annotated preferences and Stack Exchange data, to fine-tune the model effectively.

A key innovation of our approach is the specialization for Multiple-Choice Question Answering (MCQA) related to EPFL course material, optimizing the model to accurately predict single-letter answers. We also employ quantization techniques to reduce the model's memory footprint without sacrificing performance (Dettmers et al., 2022).

Our results demonstrate that our model surpasses baselines in MCQA tasks and shows robust performance across various technical subjects.

2 Related Work

Our work builds upon the foundational efforts of the TinyLlama team, leveraging the final checkpoint of their pretrained TinyLlama model, TinyLlama-1.1B-intermediate-step-1431k-3T. The architecture of TinyLlama itself is based on Meta's LLaMA, underscoring the critical contributions of both the TinyLlama and LLaMA teams to our research. The pretrained TinyLlama model will serve as our main baseline, since one should expect that our finetuned model will perform better.

Our training pipeline incorporates both a Supervised Fine-Tuning (SFT) phase and a Direct Preference Optimization (DPO) phase. This methodology is heavily inspired by the original proponents of DPO, as documented in their seminal paper. Additionally, we drew significant insights from "The Alignment Handbook" and the training recipe for Zephyr-7B-β (Tunstall et al., 2023), which sim-

ilarly employs both SFT and DPO phases. The Zephyr-7B-β recipe was particularly instrumental in guiding our choice of hyperparameters and the adoption of a specific chat template, contributing substantially to the refinement of our model.

It's important to note that TinyLlama also provides a finetuned version of their base model, TinyLlama-1.1B-Chat-v1.0, which uses the Zephyr-7B-β recipe for both SFT and DPO phases, aligning closely with our approach. Given the larger and higher-quality dataset used in their finetuning, their chat model likely outperforms ours. However, our goal is to establish a proof of concept within our resource constraints, making TinyLlama-1.1B-Chat-v1.0 an interesting, more challenging baseline.

Our data collection mirrors the approach in "Instruction Tuning with GPT-4" (Peng et al., 2023), which demonstrated the effectiveness of using GPT-4 to generate instruction-following data for finetuning LLMs. They showed that a LLaMA-7B model finetuned on GPT-4 generated data outperformed previous state-of-the-art models in zero-shot tasks. Motivated by these findings, we used the ChatGPT API (GPT-3.5) to collect preference data, essential for our finetuning process.

To generate quality preference data, we adopted the "chain of thought" approach (Wei et al., 2023), along with prompting techniques from "Large Language Models as Optimizers" (Yang et al., 2024) and "Language Models are Few-Shot Learners" (Brown et al., 2020). These strategies are further elaborated in the Approach section.

3 Approach

This section details the methodologies and strategies employed in our study. We begin with our Data Collection process, leveraging advanced prompting techniques and external datasets. We then describe the core components of our model architecture, including our approaches to Supervised Fine-Tuning, Direct Preference Optimization, Parameter-Efficient Fine-Tuning (PEFT) and Low-Rank Adaptation (LoRA), as well as our adaptations for Multiple-Choice Question Answering (MCQA) tasks and model Quantization.

3.1 Data Collection

For our fine-tuning pipeline, we primarily require two types of data. For the DPO phase, we need preference data that includes a question along with a chosen and a rejected answer. For the SFT phase, we require gold label data consisting only of a question and the desired answer. This data will be collected from two sources: student-annotated data and the H4 Stack Exchange Preferences Dataset (Lambert et al., 2023).

3.1.1 EPFL Data

A significant portion of our preference data came from a database of questions used by EPFL staff across various technical curricula. Students enrolled in CS-552 generated responses using ChatGPT, creating datasets of 100 preference pairs each. Students were responsible for fact-checking and ranking these responses.

For our own set of 300 preference pairs, we developed two prompting strategies, A and B, to generate high-quality answers. Strategy A was designed to be superior and especially handle questions outside our domain of knowledge and was preferred unless manual inspection suggested otherwise.

Prompting strategy A integrates ideas from three key papers. Research shows that very large language models excel at few-shot learning, often outperforming previous state-of-the-art fine-tuning methods (Brown et al., 2020). By providing examples, LLMs contextualize questions better and produce more accurate answers, similar to human reasoning. Doing so manually would be cumbersome so we automated this using ChatGPT with an introductory prompt like, "[...] identify the main subject area it pertains to within these fields of science. Also, outline the key points or concepts that are necessary for solving this question." This helps ChatGPT contextualize the question and identify the necessary knowledge, mimicking a human's approach. In a follow-up prompt within the same chat instance, we ask ChatGPT to solve the problem using this context.

We also incorporate "chain of thought" prompting by adding "work on this problem step-by-step" (Wei et al., 2023). Inspired by Google DeepMind's research, we extend this to, "Take a deep breath and work on this problem step-by-step," which has shown to enhance performance on grade-school math word problems (Yang et al., 2024). The full prompting strategy is detailed in the Appendix A.1.1.

Prompting Strategy B did not adopt the multi-prompt technique used in Strategy A. Instead, it leveraged the capability of Large Language Models to function as Zero-Shot Reasoners (Brown

et al., 2020). In this approach, we directly request the solution, while still incorporating the chain-of-thought phrase "work on this problem step-by-step" to guide the model's reasoning process. You find the full prompting strategy in in the Appendix A.1.2.

More exact data handling is discussed in the Data section.

3.1.2 Stack Exchange Data

Another source of data for this work is the H4 Stack Exchange Preferences Dataset (Lambert et al., 2023), which contains questions and answers from discussions across all Stack Exchange websites. We specifically focused on the Data Science, Computer Science, Physics Stack Exchange and Mathematics forums. These datasets were available directly with a downloaded.

In the Stack Exchange dataset, each answer is scored using the following formula:

$$\text{score} = \begin{cases} \text{round}\left(\log_2\left(1+u\right)\right)+a & \text{if } u \geq 0, \\ -1 & \text{otherwise,} \end{cases} \tag{1}$$

where u is the number of upvotes for a specific answer, and $a = 1$ if the answer is accepted by the user, or $a = 0$ if it is not. The fact that each answer is ranked like this lets us regulate the quality and amount of answers for our training data. More exact data handling is discussed in the Data section.

3.2 Model Architecture

TinyLlama, a compact and efficient language model, is based on a Transformer architecture influenced by the Llama 2 model. It employs a 1.1 billion parameter, decoder-only configuration that includes 22 layers, each with 32 attention heads and a grouped-query attention mechanism which segments heads into four key-value query groups to optimize computational efficiency. This structure supports an embedding size of 2048. TinyLlama incorporates advanced features such as RoPE (Rotary Positional Embedding) for injecting positional context and RMSNorm for input normalization, which stabilizes training. The use of SwiGLU, a combination of Swish and Gated Linear Unit, serves as the activation function, enhancing the model's performance while maintaining computational efficiency. These architectural choices contribute to TinyLlama's ability to achieve remarkable training throughput and competitive performance on numerous NLP tasks, despite its relatively small size compared to other large language models.

3.3 Supervised Fine-Tuning

Supervised fine-tuning, is a prevalent strategy for refining the performance of large language models (LLMs) to generate desired outcomes by aligning their responses more closely with human-like understanding and reasoning. Formally, let us consider a dataset containing I prompt-response pairs, where each prompt $\mathbf{x}_i$ is a sequence of tokens $[x_{i,1}, x_{i,2}, \ldots]$, and each associated response $\mathbf{y}_i$ is also a sequence of tokens $[y_{i,1}, y_{i,2}, \ldots, y_{i,T_i}]$. The objective of supervised fine-tuning is to adjust the model parameters $\boldsymbol{\theta}$ such that the likelihood of the actual response given the prompt is maximized. This can be expressed using the loss function:

$$\mathscr{L}_{\text{SFT}}(\boldsymbol{\theta}) = -\sum_{i=1}^{I}\sum_{t=1}^{T_i} \log \mathbf{P}(y_{i,t+1} \mid \mathbf{x}_i, \mathbf{y}_{i,1:t}; \boldsymbol{\theta}),$$

where $\mathbf{y}_{i,1:t} = [y_{i,1}, y_{i,2}, \ldots, y_{i,t}]$ (Prince, 2023). For the SFT phase of the training, we utilized the prebuilt `SFTTrainer` from the `trl` library. In the SFT training we also allow for instruction tuning, since we will use a chat template that will allow for a system instruction. This is crucial for our MCQA specialization.

3.4 Direct Preference Optimization

In the Direct Preference Optimization (DPO) phase (Rafailov et al., 2023), a preference-based loss function was used to fine-tune the model based on human preferences. This method eliminates the need for traditional reinforcement learning by directly optimizing the model's policy to align with human preferences. The DPO loss function models the probability of a human preferring one response over another. We use a variant of the DPO loss function called cDPO (Mitchell, 2023), which adjusts for noisy preference labels by assuming a probability of noise ϵ such that $0 < \epsilon < 0.5$:

$$\begin{aligned}\mathscr{L}^{\epsilon}_{\text{DPO}}(\boldsymbol{\theta}, y_w, y_l) = (1-\epsilon)\cdot \mathscr{L}_{\text{DPO}}(\boldsymbol{\theta}, y_w, y_l) \\ + \epsilon \cdot \mathscr{L}_{\text{DPO}}(\boldsymbol{\theta}, y_l, y_w)\end{aligned} \tag{2}$$

In this equation, $\boldsymbol{\theta}$ are the weights of the policy model being optimized, and variables y_w and y_l denote the preferred and less preferred responses, respectively. The parameter β controls the strength of the KL-divergence constraint, and σ denotes the

logistic sigmoid function. Further the vanilla DPO loss function is defined as

$$\mathscr{L}_{\text{DPO}}(\boldsymbol{\theta}, y_w, y_l) = -\log \sigma\left(\beta\, h_{\pi_{\boldsymbol{\theta}}}^{y_w, y_l}\right)$$
$$h_{\pi_{\boldsymbol{\theta}}}^{y_w, y_l} = \log \frac{\pi_{\boldsymbol{\theta}}(y_w)}{\pi_{\text{ref}}(y_w)} - \log \frac{\pi_{\boldsymbol{\theta}}(y_l)}{\pi_{\text{ref}}(y_l)}$$

In this equations, $\pi_{\boldsymbol{\theta}}$ is the policy model being optimized, and π_{ref} is the reference model, which is often the same as the initial model before DPO.

For the DPO phase of the training, we utilized the prebuilt `DPOTrainer` from the `trl` library. We did not need to create a custom loss function because `DPOTrainer` already includes the cDPO loss as an alternative.

3.5 PEFT and LoRA

Parameter-Efficient Fine-Tuning (PEFT) and Low-Rank Adaptation (LoRA) are techniques designed to enhance the efficiency of fine-tuning large pre-trained models. Traditional fine-tuning involves updating all parameters of a model, which can be computationally expensive and memory-intensive. PEFT addresses this by focusing on adjusting only a small subset of parameters, thereby reducing the computational and memory overhead. LoRA, a specific method within the PEFT framework, further improves efficiency by decomposing the weight updates into low-rank matrices (Hu et al., 2021). We use a LoRA adapter to be able to adhere to our time constraints and computational resources.

Mathematically, given a pre-trained weight matrix $W \in \mathbb{R}^{d\times k}$, LoRA represents the update ΔW as the product of two lower-rank matrices A and B, such that $\Delta W = AB$, where $A \in \mathbb{R}^{d\times r}$ and $B \in \mathbb{R}^{r\times k}$. The adapted weight matrix is then $W' = W + \Delta W = W + AB$. Furthermore in the paper they scaleΔW by $\frac{\alpha}{r}$, where α is hyperparameter, similar to a learning rate. During fine-tuning, only the matrices A and B are learned, while the original weight matrix W remains fixed. This reduces the number of parameters to be learned from $d \times k$ to $d \times r + r \times k = r(d + k)$, where $r << \min\{d, k\}$, leading to significant computational and memory savings.

Finally, we merge the the trained LoRA adapter into the base model, to utilize the full capabilities of the base model.

3.6 MCQA specialization

We specialized our model for Multiple-Choice Question Answering (MCQA) using the same SFT and DPO pipeline but focused on MCQA data where a single letter is the gold/preferred output. To avoid overwriting our previous fine-tuned model, which might affect non-MCQA responses, we used a chat template with specific system instructions. This allows our model to retain full reasoning capabilities for non-MCQA questions.

During inference, we predict one of the alternatives: A, B, C, or D, by running `model.forward()` on the question (plus instruction) and examining the logits for the subsequent token corresponding to each option. The token with the highest logit becomes our MCQA answer prediction. This approach usually aligns with the output of `model.generate()`. In rare cases where the model does not predict only one letter (a situation we have not encountered), we still obtain the letter with the highest probability.

3.7 Quantization

The final step in our model development was quantization, converting the model from `BF16` to `int8` precision to reduce its memory footprint. This was done using Huggingface's `BitsAndBytesConfig`, and enables efficient 8-bit model operations.

We employed the LLM.int8() technique (Dettmers et al., 2022), which optimizes transformer matrix multiplications through vector-wise quantization and mixed-precision for outliers. This method uses distinct normalization constants for each matrix vector, ensuring precise low-bit computation for most data while applying 16-bit precision to significant outliers. This hybrid approach maintains over 99.9% of data points in 8-bit, preserving model accuracy and significantly reducing memory use.

We quantized only the final MCQA specialized model, focusing on its ability to provide single-letter answers. Since we do not overwrite our model's ability for long answers, we will also evaluate its performance on those type of answers using all our evaluation metrics.

4 Experiments

4.1 Data

4.1.1 SFT

Our SFT data consists purely of the collected Stack Exchange dataset. To ensure high-quality answers, we selected questions with the best answer as the gold output if the score in equation 1 was greater

than or equal to 4, which roughly equates to 10 upvotes.

Furthermore, we cleaned the data using Python's BeautifulSoup to remove HTML code. For the SFT training, we used the chat template provided by Zephyr-7B-β (Tunstall et al., 2023), as seen below.

Chat Template

<|system|>
You are an experienced teacher who answers the STEM-related question asked by a student below.</s>
<|user|>
What is the time complexity of merge sort?</s>
<|assistant|>
...

Our Instruction for the non MCQA specialized model is "You are an experienced teacher who answers the STEM-related question asked by a student below."

Since our model has a max sequence length of 2048 we remove data points (tokenized) that exceed this threshold. In fact, for the sft training we concatenate the prompt with gold output, together with the chat template, so we actually remve datapoints whoose sum of tokenized prompt + answer lengths is larger than 2000 to also have margin for the template. This yields roughly 11k datapoints, where we used 90% for training and 10% testing. We did not find the need for a validation set since we did not do any hyperparameter tuning based on the test data.

4.1.2 DPO

For the DPO training dataset, we used both student-annotated preference data and the Stack Exchange dataset. We cleaned the student data by removing non-alphanumeric answers as some data points had chosen answers labeled as "..." and we removed also questions related to MCQA, which were reserved for specialization.

For Stack Exchange, preferences were based on the best and worst answers, accepting any best answer with a score of ≥ 2. Data points that were too long were removed. Specifically, if the combined (tokenized) length of the prompt and chosen answer, or prompt and rejected answer, exceeded 2000 tokens, we discarded the data point. Here we also included the chat template for the prompt.

This process yielded approximately 30,000 Stack Exchange and 11,000 student-annotated preferences. We sampled 10,000 data points from each to create a dataset of 20,000, with 18,000 for training and 2,000 for testing.

4.1.3 MCQA

As previously mentioned, we removed the MCQA data from the original student-annotated dataset to create a specialized model that responds with a single letter corresponding to the correct option. The preference dataset had the correct one-letter option as the chosen output and the original long-form response (which could be incorrect) as the rejected output.

Creating this dataset posed two challenges: first, the chosen answers were long-form responses including reasoning, and second, multiple student annotations led to varying correct options for the same question. We addressed the first problem by an automated script using ChatGPT to map each chosen answer to one letter, and to solve the second problem we selected the most frequently occurring option as the correct answer for each question. This resulted in approximately 14,000 data points. Additionally, we created a non-duplicate dataset with only the correct answers, yielding around 800 data points for the SFT regime, as our specialized model will go through the same SFT+DPO pipeline.

To ensure consistent data splits for SFT and DPO, we performed a 90%-10% split on both datasets. We trained the model on top of our previously trained model, using a different instruction in the chat template to avoid overwriting the performance on long-form answers. For the MCQA task, the instruction was: "You are an experienced teacher who answers the STEM-related question asked by a student below. This is a multiple-choice question, thus you must answer with only one letter corresponding to the correct answer."

4.2 Evaluation Method

To evaluate the performance of our models, we employed a combination of automatic metrics and task-specific accuracy measures. The primary metrics used for evaluation include BLEU, BERTScore, ROUGE, METEOR, and MCQA accuracy.

- **BLEU and METEOR:** Measure the precision and relevance of generated responses by

comparing n-gram overlap and alignment with reference answers, providing insights into text quality.

- **BERTScore:** Evaluates semantic similarity using pre-trained BERT embeddings to compute precision, recall, and F1 score, so to capture contextual nuances.
- **ROUGE:** Assesses coherence by measuring the overlap of n-grams, word sequences, and word pairs between generated and reference texts. We use ROUGE-1 (unigrams), ROUGE-2 (bigrams), and ROUGE-L (longest common subsequence) to evaluate different aspects of text similarity.
- **MCQA Accuracy:** Measure the accuracy in predicting the correct option (A, B, C, or D) for MCQ answering task, to reflect the performance of the model in our task. Additionally, we evaluate examples of correct and incorrect MCQ predictions to highlight the model's strengths and areas of difficulty.

4.3 Baselines

We employ two baselines. The primary baseline is the pretrained model `TinyLlama-1.1B-intermediate-step-1431k-3T`, we expect improvements post-fine-tuning, and this serves as our main success criterion.

Our more stringent baseline is `TinyLlama-1.1B-Chat-v1.0`, fine-tuned on the UltraChat dataset with 200k synthetic dialogues from ChatGPT, and further aligned using the `DPOTrainer` on the UltraFeedback dataset, which includes 64k prompts and GPT-4 ranked completions. This model uses a higher LoRA rank $r = 128$, resulting in more trainable parameters. Given its extensive training, we hypothesize it will outperform our model, hence it serves as a tougher baseline.

We exclude comparisons to larger state-of-the-art models like GPT-4 or Claude, considering them unfair due to differences in scale. Despite being more stringent, `TinyLlama-1.1B-Chat-v1.0` remains a fair baseline, sharing the same pretrained model and SFT+DPO pipeline.

4.4 Experimental Details

We used the Unsloth ecosystem and LoRA, targeting all projection modules. While Unsloth supports 4-bit precision, we used TinyLlama in its original BF16 format, planning to quantize later. Training was done on Google Colab, funded by our own budget.

Supervised Fine-Tuning (SFT): We used the `SFTTrainer` class from the `trl` library with the base model, tokenizer, and dataset, employing a cross-entropy loss function. For the non-MCQA model, LoRA dimensions were $r = 32$ and $\alpha = 32$. For the MCQA model, we used $r = 8$ and $\alpha = 8$ to reduce trainable parameters and mitigate overfitting due to the smaller dataset.

Direct Preference Optimization (DPO): We used the cDPO method with label smoothing parameter $\epsilon = 0.2$ to handle noisy preferences. Training had a learning rate of 5×10^{-7}, a cosine learning rate scheduler, and a warmup ratio of 0.1, with a batch size of 4 over 2 epochs. For the non-MCQA model, we used $r = 32$ and $\alpha = 32$, while for the MCQA model, we used $r = 8$ and $\alpha = 8$. We also experimented with β, using $\beta = 0.1$ and $\beta = 0.4$, finding performance largely insensitive to this change.

4.5 Results

Our evaluation metrics reveal how our models stack up against the baselines, as seen in Tables 1 and 2. Table 1 contains the comparison with the first baseline (TinyLlama-1.1B-intermediate-step-1431k-3T), while Table 2 shows the comparison with the second baseline (TinyLlama-1.1B-Chat-v1.0). The following points summarize our observations:

- **BLEU:** The EPFLLaMA models had lower BLEU scores compared to the baselines. This was somewhat expected, as our fine-tuning was focused on technical tasks using SFT and DPO rather than on general language generation.
- **BERTScore, ROUGE, and METEOR:** The EPFLLaMA models performed well on BERTScore, ROUGE-1, ROUGE-L, and METEOR, often surpassing the first baseline. These results show that our models can generate contextually relevant, coherent, and high-quality responses, especially in academic and STEM-specific tasks. So despite the lower BLEU scores, these metrics confirm the models' strength in specialized areas.
- **MCQA Accuracy:** The MCQA-specific models (EPFLLaMA-MCQA and EPFLLaMA-MCQA-Quantized) really shined here. The main MCQA model achieved a respectable

Best | Better than Baseline | Worst | Not Investigated | Baseline

Model	BLEU	BERTScore Precision	BERTScore Recall	BERTScore F1	ROUGE-1	ROUGE-2	ROUGE-L	METEOR	MCQA Accuracy
TinyLlama-1.1B-intermediate-step-1431k-3T	0.0640	0.7663	0.7726	0.7687	0.1777	0.0518	0.1374	0.1391	32.5%
EPFLLaMA	0.0471	0.7889	0.7940	0.7909	0.1921	0.0460	0.1496	0.1446	27.0%
EPFLLaMA-MCQA	-	-	-	-	-	-	-	-	45.0%
EPFLLaMA-MCQA-Quantized	0.0415	0.8395	0.7968	0.8166	0.1863	0.0696	0.1408	0.1086	42.5%

Table 1: Evaluation of our models using TinyLlama-1.1B-intermediate-step-1431k-3T as baseline.

Model	BLEU	BERTScore Precision	BERTScore Recall	BERTScore F1	ROUGE-1	ROUGE-2	ROUGE-L	METEOR	MCQA Accuracy
TinyLlama-1.1B-Chat-v1.0	0.1224	0.8347	0.8300	0.8319	0.3241	0.1161	0.2022	0.2232	22.5%
EPFLLaMA	0.0471	0.7889	0.7940	0.7909	0.1921	0.0460	0.1496	0.1446	27.0%
EPFLLaMA-MCQA	-	-	-	-	-	-	-	-	45.0%
EPFLLaMA-MCQA-Quantized	0.0415	0.8395	0.7968	0.8166	0.1863	0.0696	0.1408	0.1086	42.5%

Table 2: Evaluation of our models using TinyLlama-1.1B-Chat-v1.0 as baseline.

45.0% accuracy, while the quantized version had a slight drop to 42.5%, which is still quite good considering it is half the size of the uncompressed model. This highlights the models' strong capabilities in handling MCQA tasks, enhanced by our specialized training and quantization efforts.

5 Analysis

5.1 Qualitative Evaluation

In this section, we perform a qualitative evaluation of our MCQA model's performance, focusing on the areas where it excels and we try to identify patterns in its failures. Our analysis is based on a detailed examination of individual prediction examples. Selected examples where the model correctly and incorrectly predicted the answer are detailed in the Appendix A.2

5.1.1 Correct Predictions

Our MCQA specialized model performed well on various technical questions, especially in machine learning, probability, and optimization. The model was able to accurately answer questions based on solid theoretical knowledge, and shows a grasp of key concepts such as the bias-variance tradeoff, noise impact in regression models, dimensionality reduction in PCA, and GAN mechanisms. You can refer to the Appendix for some examples Table 3.

5.1.2 Incorrect Predictions

Despite the overall good performance, the model exhibited some notable weaknesses, particularly in interpreting complex theoretical nuances and logical reasoning. Selected examples of incorrect predictions are detailed in the Appendix Table 4. Specifically, the model struggles with:

- **Complex Algorithmic Concepts**: Misinterpreting practical limitations versus theoretical capabilities, such as in the regression task with neural networks.
- **Optimization Theory**: Failing to correctly apply conditions for gradient descent convergence, indicating a gap in understanding the nuances of optimization algorithms.
- **Model Architectures**: Incorrectly identifying the architecture of models like FastText, suggesting a need for deeper training on specific model structures.

Overall, the model demonstrates strong performance in specific technical domains but shows weaknesses in more complex reasoning and broader language generation tasks. These insights will inform future iterations and improvements, focusing on enhancing the model's understanding of nuanced theoretical concepts and its general language generation capabilities.

6 Ethical considerations

6.1 Adaption for multilinguality

For high-resource languages like French and German, the first step is to gather a substantial amount of data in the target language and ensure proper tokenization. The model can then be fine-tuned on this new dataset, starting from the existing fine-tuned English model. Although TinyLLaMA is not inherently multilingual, fine-tuning it on a new language dataset is possible but requires significant computational resources.

Adapting the fine-tuned model for low-resource languages like Urdu and Swahili can be effectively achieved using the technique of back-translation (Li et al., 2024). Back-translation is particularly valuable in scenarios where the availability of large, high-quality datasets is limited. This method involves translating text from the source language (in this case, English) to the target low-resource language and then translating it back to the source language. This process generates new, synthetic data that can significantly augment the training dataset and improve the model's performance.

6.2 Adaption for sign language

To adapt the fine-tuned model for sign language, a video processing module would be integrated to convert sign language videos into text. This module, trained on a dataset specifically for sign language to text conversion, uses computer vision and gesture recognition techniques. The generated text, which may have different sentence structures due to the nature of sign language, would then be processed by the language model. The language model would have to be fine-tuned on text generated from sign language to accurately handle the unique sentence structures and linguistic nuances of sign language.

6.3 Potential Beneficiaries and Risks of Harm

If the fine-tuned model works as intended, students and educators benefit from receiving accurate answers to various educational questions, enhancing both learning and teaching experiences. However, potential harms include reinforcing biases present in the student-generated training data, spreading inaccuracies, and facilitating academic dishonesty if used to cheat. The training data could also pose privacy risks if not properly secured. While the model offers significant educational advantages, it requires careful handling to avoid misuse and data privacy issues.

6.4 Impact on Vulnerable Groups and Mitigations

Since the model is trained on complex, university-level computer science questions, users without sufficient prior knowledge might struggle to use it effectively. This could disproportionately affect students from disadvantaged backgrounds who may not have had access to high-quality educational resources. To mitigate these risks, it is crucial to ensure diverse and representative training data. Additionally, providing clear documentation and supplementary educational resources can help bridge the knowledge gap and make the model more accessible to all users. Training the model on more basic data as well would help support those with varying levels of knowledge, making the tool more inclusive and beneficial.

7 Conclusion

This project aimed to enhance the TinyLlama model for answering EPFL-related university course questions. By implementing a dual-phase training process of Supervised Fine-Tuning (SFT) and Direct Preference Optimization (DPO), we improved the model's ability to provide accurate and relevant answers. Specialization for Multiple-Choice Question Answering (MCQA) and quantization techniques also effectively reduced the model's memory footprint while maintaining performance.

Our results show that our models outperform established baselines in MCQA tasks, demonstrating strong performance across various technical subjects. This highlights the potential of open-source LLMs in educational settings, offering reliable and context-aware responses to complex academic queries.

From this project, we learned that the dual-phase training process balances general language understanding with domain-specific knowledge, and leveraging diverse datasets enhances adaptability. Extending the noisy student preference dataset with Stack Exchange data was crucial for our performance improvement.

However, the project revealed limitations such as dependency on training data quality and struggles with complex theoretical nuances. The computational resources required for fine-tuning and the need for continuous updates are ongoing challenges.

Future work could involve expanding the training dataset, exploring transfer learning for low-resource languages, and integrating real-time feedback mechanisms. On the other hand, ensuring ethical use and addressing data privacy concerns remain critical.

Overall, this project demonstrates the feasibility and benefits of tailoring LLMs for specific educational purposes, and paves the way for more sophisticated and accessible AI-driven learning tools.

References

Microsoft Research AI4Science and Microsoft Azure Quantum. 2023. The impact of large language models on scientific discovery: a preliminary study using gpt-4.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners.

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

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.

Nathan Lambert, Lewis Tunstall, Nazneen Rajani, and Tristan Thrush. 2023. Huggingface h4 stack exchange preference dataset.

Xian Li, Ping Yu, Chunting Zhou, Timo Schick, Omer Levy, Luke Zettlemoyer, Jason Weston, and Mike Lewis. 2024. Self-alignment with instruction backtranslation.

Eric Mitchell. 2023. A note on dpo with noisy preferences & relationship to ipo.

Baolin Peng, Chunyuan Li, Pengcheng He, Michel Galley, and Jianfeng Gao. 2023. Instruction tuning with gpt-4.

Simon Prince. 2023. Training and fine-tuning large language models. https://www.borealisai.com/research-blogs/training-and-fine-tuning-large-language-models/.

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.

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.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. 2024. Large language models as optimizers.

8 Contributions

In our project, the work was done together as a team, with all members involved in every aspect of the project.

A Appendix

A.1 Prompting Strategies

A.1.1 Prompting Strategy A

Prompting Strategy A

User: Here is a technical question from a STEM course: <question>. Please analyze the question and identify the main subject area it pertains to within these fields of science. Also, outline the key points or concepts that are necessary for solving this question.

ChatGPT: ...

User: Using the subject area and key points you identified from the earlier question, provide a detailed step-by-step solution to the problem: <question>. Consider each step in the process, and explain how you arrive at the solution, including any necessary algorithms, calculations, or theoretical considerations. Take a deep breath and work on this problem step-by-step. Note that any math should be written in LaTeX.

ChatGPT: ...

A.1.2 Prompting Strategy B

Prompting Strategy B

User: Here is a technical question from a STEM course: <question>. Provide a solution to the problem, including an explanation. Work on this problem step-by-step. Note that any math should be written in LaTeX.

ChatGPT: ...

A.2 MCQA Performance Examples

Question	Prediction	Label
A model which has a high bias necessarily has a low variance. Options: A. True B. False	B	B
Consider a regression model where data (x, y) is generated by input x uniformly randomly sampled from $[0, 1]$ and $y(x) = x^2 + \epsilon$, where ϵ is random noise with mean 0 and variance 1. Two models are carried out for regression: model A is a trained quadratic function $g(x; \mathbf{w}) = w_2x^2 + w_1x + w_0$ where $\mathbf{w} = (w_0, w_1, w_2)^\top \in \mathbb{R}^3$, and model B is a constant function $h(x) = 1/2$. Then compared to model B, model A has ? Options: A. higher bias, higher variance. B. higher bias, lower variance. C. lower bias, higher variance. D. lower bias, lower variance.	C	C
In principal component analysis, the left singular vectors $\mathbf{U}$ of a data matrix $\mathbf{X}$ of shape (d features, n datapoints) are used to create a new data matrix $\mathbf{X}' = \mathbf{U}^\top\mathbf{X}$. To achieve dimensionality reduction, we keep only certain rows of the matrix $\mathbf{X}'$. We keep those rows that have: Options: A. the lowest variance. B. the highest variance. C. smallest L2 norm. D. L2 norm closest to 1	B	B
Generative Adversarial Networks use the generator and discriminator models during training but only the discriminator for data synthesis. Options: A. True B. False	B	B

Table 3: Examples of correct predictions by the MCQA model.

Question	Prediction	Label
You are doing your ML project. It is a regression task under a square loss. Your neighbor uses linear regression and least squares. You are smarter. You are using a neural net with 10 layers and activation functions $f(x) = 3x$. What is the reason for the outcome of this bet? Options: A. Because we use exactly the same scheme. B. Because it is almost impossible to train a network with 10 layers without a supercomputer. C. Because I should have used more layers. D. Because I should have used only one layer.	C	A
Consider the loss function $L : \mathbb{R}^d \to \mathbb{R}$, $L(\mathbf{w}) = \frac{\beta}{2}\|\mathbf{w}\|^2$, where $\beta > 0$ is a constant. We run gradient descent on L with a stepsize $\gamma > 0$ starting from some $\mathbf{w}_0 \neq 0$. Which of the statements below is true? Options: A. Gradient descent converges to the global minimum for any stepsize $\gamma > 0$. B. Gradient descent with stepsize $\gamma = \frac{2}{\beta}$ produces iterates that diverge to infinity ($\|\mathbf{w}_t\| \to \infty$ as $t \to \infty$). C. Gradient descent converges in two steps for $\gamma = \frac{1}{\beta}$ (i.e., $\mathbf{w}_2$ is the first iterate attaining the global minimum of L). D. Gradient descent converges to the global minimum for any stepsize in the interval $\gamma \in (0, \frac{2}{\beta})$.	A	D
The FastText supervised classifier can be modeled as a one-hidden-layer neural network. Options: A. True B. False	B	A
The kernel $K(x, x') = \cos(x + x')$ is a valid kernel. Options: A. True B. False	B	A

Table 4: Examples of incorrect predictions by the MCQA model.