Perplexed Tutor: An Efficient and Aligned AI for STEM MCQA

Alexander Wohlfahrt | 375419 | alexander.wohlfahrt@epfl.ch
Anasse El Boudiri | 374212 | anasse.elboudiri@epfl.ch
Jan Steiner | 374940 | jan.steiner@epfl.ch
Nathan Felber | 326410 | nathan.felber@epfl.ch
Perplexed Prompters

Abstract

As large language models (LLMs) become common tools for student learning, we assess the potential of small-scale LLMs specialized in answering multiple choice questions by applying several state-of-the-art techniques to the Qwen3-0.6B-Base model. We successfully improve the base model's performance across several metrics and discuss our findings in this report.

1 Introduction

Large Language Models (LLMs) like ChatGPT and DeepSeek are increasingly used by students for concept understanding and problem-solving. While instructors and TAs remain key, AI tools offer flexible, always-available support.

However, paywalled premium versions create inequality, especially in STEM, by offering better tools and longer context. Open-source models exist but are often too large for local deployment. Our EPFL tutor addresses this with a lightweight, open, and locally deployable design.

We develop a compact STEM-focused tutor by fine-tuning Qwen3-0.6B-Base (Team, 2025) on multiple-choice questions, aligning its behavior via Direct Preference Optimization (DPO) (Rafailov et al., 2024) using student preferences, and integrating retrieval-augmented generation (RAG) (Lewis et al., 2020). Quantization ensures the model remains efficient and deployable.

Given limited and unstable training resources, we used small batch sizes and frequent checkpoints for efficient, resilient training.

2 Approach

2.1 Reward Model

To align the tutor's behavior with EPFL student preferences, we implemented a reward modeling pipeline using Qwen3-0.6B-Base as the underlying policy model. The training dataset 3.1.1, collected during Milestone 1, consisted of preference triplets $\mathcal{D} = (x_i, y_i^+, y_i^-)_{i=1}^N$, where each triplet includes a student-style prompt x_i, a preferred response y_i^+, and a less preferred alternative y_i^-.

All the trained models use the following DPO objective :

$$\mathcal{L}_{\text{DPO}} = -\log \sigma \left(\beta \log \frac{P_{\text{curr}}(y^+ \mid x)}{P_{\text{base}}(y^+ \mid x)} - \beta \log \frac{P_{\text{curr}}(y^- \mid x)}{P_{\text{base}}(y^- \mid x)} \right)$$

2.2 MCQA

We fine-tune the Qwen3-0.6B-Base model to answer multiple-choice questions related to STEM using full supervised fine-tuning (SFT) over a large training dataset of MCQAs aggregated from several sources and post-processed using ChatGPT. The method consists of resuming the training process of a pre-trained model to acquire specific capabilities or knowledge on a typically much smaller set of examples than was used during the initial training phase.

We consider two different approaches. In the first, we train a model to generate comprehensive answers with question breakdown, correct option, and rationale. In the second approach, we train the model to only generate the correct option. The former attempts to create a more useful learning assistant, which is more aligned with the objectives of the project and our ethical considerations. However, this is a very ineffective approach because our models are evaluated via single token prediction accuracy and the model is incentivized to generate reasoning instead. The latter approach, while providing less value to the user, has the potential to achieve significantly higher performance on the evaluation benchmark. Both are implemented in 3.2 and compared in 4.2.

2.3 Quantization

2.3.1 Quantization and Adapter

We quantized the MCQA model, a fine-tuned version of Qwen3-0.6B-Base. We followed the

QLoRA approach (Dettmers et al., 2023), combining 4-bit quantization using the BitsAndBytes library with LoRA adapters (Hu et al., 2021) to fine-tune a small number of parameters on top of a frozen base model. The goal was to significantly reduce memory usage while keeping, or even improving, model performance.

To train the quantized model, we used a subset of 1,024 examples from the original MCQA training dataset. This reduced dataset size proved effective even when applying LoRA adapters to all linear layers in the model, allowing us to fine-tune a moderately larger set of parameters while still avoiding overfitting and keeping training efficient.

We then applied LoRA adapters to all linear layers in the transformer architecture by setting target_modules="all-linear". This allowed us to inject trainable low-rank matrices throughout the model while keeping the original weights frozen, enabling efficient fine-tuning without full retraining.

2.3.2 Training Setup and Evaluation

We trained for two epochs on 1,024 samples using the Trainer API with adamw_bnb_8bit, a 3e-4 learning rate, and LoRA settings of r=16, alpha=32, and dropout=0.08. This setup led to excellent results in terms of both accuracy and VRAM efficiency.

2.4 RAG

Our RAG pipeline is composed of two main components: a generator and a retriever.

For the generator, we experimented with several strategies. The best-performing approach turned out to be reusing the same model and training procedure as our MCQA model, which had already been fine-tuned for multiple-choice reasoning.

For the retriever, we fine-tuned the GTE-ModernBERT (Zhang et al., 2024b; Li et al., 2023) model on synthetic data that we generated using the GPT wrapper and our custom document corpus.

One other important aspect is our document corpus. We manually curated a STEM-focused document collection from Wikipedia and arXiv, yielding an initial corpus of 97.2k documents. Each document was chunked into 512-token segments. Some were summarized using GPT when too long or too noisy. However, due to difficulties with chunking scientific papers and poor retrieval performance, arXiv documents were ultimately removed. The final corpus comprises 62.2k cleaned Wikipedia STEM articles, which stay under the 51.2M tokens limit.

3 Experiments

3.1 Reward Model

In this section, we detail the experimental setup used to align the reward model based on preferred outputs from annotated data.

3.1.1 Reward Model dataset

The training data was collected during Milestone 1 and consists of preference pairs generated by GPT-4o-mini. Each example includes a prompt, a preferred response, and a less preferred alternative. Student annotations were based on four criteria: correctness, relevance, clarity, and completeness. The dataset was randomly split into 90% for training and 10% for evaluation.

3.1.2 Reward Model evaluation method

We use accuracy as the primary evaluation metric: the percentage of examples for which the model assigns a higher score to the preferred response y^+ than to the rejected response y^-. This metric directly reflects the alignment of the model's output preferences with human annotations.

The evaluation is conducted on a held-out test set comprising 10% of the annotated preference data.

The score for each response is defined as the folowing:

$$\text{chosen\_reward} = \log P_{\text{policy}}(y^+ \mid x) - \log P_{\text{ref}}(y^+ \mid x)$$

$$\text{rejected\_reward} = \log P_{\text{policy}}(y^- \mid x) - \log P_{\text{ref}}(y^- \mid x)$$

The model is considered accurate on a given example if:

$$\text{chosen\_reward} > \text{rejected\_reward}$$

During training, we tracked accuracy and reward margin over time with Weights and Biases (see appendix B) to assess convergence, stability, and the model's ability to distinguish preferred from rejected responses.

3.1.3 Reward Model experimental details

We tested three reward modeling strategies using Qwen3-0.6B-Base: **Direct Preference Optimization (DPO)** applied to the base model, a **LoRA-based reward head** with the base model kept frozen, and a **two-stage approach** consisting of supervised fine-tuning on (x, y^+) pairs followed by DPO.

Additionally, we experimented with Regularized Preference Optimization (RPO) and Weighted Preference Optimization (WPO) as alternative alignment methods. However, all attempts using either loss function resulted in out-of-memory (OOM) errors, preventing successful training.

For the standard DPO configuration, we used a maximum sequence length of 4096 and a maximum prompt length of 512. The temperature parameter was set to $\beta = 0.1$. A batch size of 32 was used. The learning rate was set to 7×10^{-6}. Training was carried out over 3 epochs.

3.1.4 Reward Model results

We report the reward model's accuracy on the held-out test set for each training configuration in Table 1.

Reward Model	Reward Accuracy
DPO	**0.731** ± 0.009
SFT + DPO	0.572 ± 0.010
LoRA + DPO	0.480 ± 0.010

Table 1: Reward model accuracy for each training configuration.

Applying DPO directly to the base model yielded the best performance. Additional experiments and results are provided in the appendix C.

3.2 MCQA

This section describes the experiments we conducted to fine-tune our MCQA model.

3.2.1 MCQA dataset

Perhaps the most important step in supervised fine-tuning is gathering a sufficiently large dataset of quality examples relevant to the task. To achieve this, we first collected 24,039 STEM-related questions-answer pairs from *stemQ* (Drori et al., 2024), *MATH* (Hendrycks et al., 2021b), *QASC* (Khot et al., 2020), the *ScienceQA* (Lu et al., 2022) train split, and the preference data collected for the project.

We then post-processed the question-answer pairs using the ChatGPT API to standardize all answers and reformulate non-multiple-choice questions as MCQAs. Our dataset contains the following fields:

- **dataset**: the dataset the question originates from.
- **id**: a unique id associated with the entry.
- **question**: the question. If the original question is not an MCQ, it is reformulated as an MCQ during post-processing. However, it is not formatted to maintain the robustness of the model to a variety of question formats.
- **answer**: the expected answer. It follows a specific format composed of question breakdown, correct answer and rationale to guide our model towards useful responses and enhanced reasoning capabilities.
- **single_token_answer**: the correct answer option (e.g. A, B, C or D). This field is useful for fine-tuning our single token model.

The exact ChatGPT prompts used are shown in Appendix D, and the complete dataset is available on Hugging Face (Felber, 2025). We fine-tuned two separate MCQA models discussed in sections 3.2.2 and 3.2.3 on this dataset, starting from Qwen3-0.6B-Base.

3.2.2 Reasoning model

Our first model is fine-tuned on the entire train split of our MCQA dataset for 3 epochs with a batch size of 64 and a learning rate of 5×10^{-5}. These settings have been experimentally determined to yield a good compromise between sample efficiency and overfitting. The actual training examples were obtained by concatenating the *question* and *answer* fields of each dataset entry. An example is shown in Listing 3, and the training loss of the model is shown in Fig. 3.

3.2.3 Single token answer model

Our second model is fine-tuned using the same dataset and parameters. However, this time the training examples were created by concatenating the *question* and *single_token_answer* fields of each dataset entry in order to match the format expected by the evaluation suite. An example is shown in Listing 4, and the training loss of the model is shown in Fig. 4.

3.2.4 MCQA evaluation

We evaluated our two models on the provided Lighteval benchmark using the zechen-nlp/MNLP_STEM_mcqa_evals dataset available on Hugging Face. To assess the quality of our dataset, we also compare against a version of the single token model fine-tuned on the MMLU train split (Hendrycks et al., 2021a), which includes

$\sim$100,000 MCQAs from a diversity of scientific datasets. We include the model we fine-tuned on a small subset of the dataset for milestone 2 of the project in our comparison, as well as two baselines: Qwen3-0.6B-Base and Qwen3-0.6B (already fine-tuned as a chat model). The accuracy of each model is shown in Table 2.

Model	Accuracy
Qwen3-0.6B-Base	0.425 ± 0.018
Qwen3-0.6B	0.313 ± 0.017
MCQA_M2	0.423 ± 0.018
MCQA_M3_reasoning	0.335 ± 0.017
MCQA_M3_st	**0.442 ± 0.018**
MCQA_M3_st_MMLU	0.352 ± 0.017

Table 2: Accuracy of MCQA models. Mx denotes the model at milestone x; *st* indicates a single-token answer variant; *MMLU* refers to models fine-tuned on the MMLU training split.

The best accuracy is achieved by our single token model as expected, with a 4% improvement over the Qwen3-0.6B-Base baseline. Interestingly, all other models perform worse than the baseline, with Qwen3-0.6B performing the worst. This may be due to the fine-tuned models becoming more specialized and therefore less accurate for the specific task of predicting single-token MCQ answers. In particular, our reasoning model performs significantly worse despite being trained on the exact same questions, which emphasizes the importance of the answer format. We also observe that dataset quality is very important, as highlighted by the poor performance of the model fine-tuned on the MMLU train split, which might be contaminated with less domain-specific questions than our own.

3.3 Quantization

3.3.1 Exploration and Toolkit Evaluation

We explored quantization using GPTQ (Frantar et al., 2023) and SmoothQuant (Xiao et al., 2024) via LLM Compressor on a subset of GSM8K (Cobbe et al., 2021). Although tuning schemes like W8A8, W4A8, W4A16, and smoothing strenghts reduced model size, performance gains were limited and inference latency increased.

3.3.2 QLoRA on MCQA

We based all final results on a snapshot of the MCQA model to ensure reproducibility. QLoRA, combined with BitsAndBytes in 4-bit mode, outperformed previous methods in both accuracy and efficiency, and was adopted for the final model. Alternatives like OpenVINO, ORT Quantizer, and Unsloth were briefly explored but found less suitable.

3.3.3 Evaluation Strategy and Results

For evaluation, we used a modified version of Lighteval from EPFL staff. We added peak VRAM tracking and excluded model loading time for more stable duration evaluation time. These changes allowed for fairer and more consistent comparisons between models. All models were evaluated using the zechen-nlp/MNLP_STEM_mcqa_evals dataset. To better capture both performance and efficiency, we used a custom metric: $\text{Score} = \frac{\text{Accuracy}}{\text{Average Peak VRAM}}$. We eventually adopted QLoRA for efficient fine-tuning without full retraining.

We experimented with various dataset sizes (from 256 to 8,192 examples), training hyperparameters, and LoRA configurations. Our best setup used 1,024 training examples with r=16, lora_alpha=32, lora_dropout=0.08, trained over 2 epochs with a learning rate of 3e-4 and the adamw_bnb_8bit optimizer. Below, we compare our best quantized model with the original baseline.

Model	VRAM	Accuracy	Score
Baseline	3.42 GB	0.42 ± 0.018	0.124
QLoRA	**2.81 GB**	**0.43** ± 0.018	**0.152**
W8A8+0.1	3.44 GB	**0.43** ± 0.018	0.125
BnB+8bit	3.01 GB	0.42 ± 0.018	0.138
BnB+4bit	**2.81 GB**	0.42 ± 0.018	0.149

Table 3: Model average peak VRAM, accuracy and score (acc / avg peak VRAM) for the best tested models

Model	Size	Duration
Baseline	1.11 GB	**52.463**
QLoRA	**506.88 MB**	85.823
W8A8+0.1	717.53 MB	167.033
BnB+8bit	716.88 MB	148.391
BnB+4bit	**506.88 MB**	103.400

Table 4: Model size and evaluation duration for the best tested models

While early experiments with GPTQ and SmoothQuant (via LLM Compressor) allowed us to explore various quantization levels such as W8A8, W4A16, and W4A8, they allowed to maintained high accuracy, but only modest memory savings.

In contrast, QLoRA maintained the highest accuracy (0.43 ± 0.018) despite aggressive 4-bit quantization, matched only by W8A8+0.1, while using significantly less VRAM (2.81 GB vs. 3.44 GB) and achieving the highest overall Score (0.152). It also reduced the model size by around half (506.88 MB vs. 1.11 GB). Other configurations like BnB+8bit and especially BnB+4bit performed close to QLoRA, with BnB+4bit achieving nearly identical VRAM usage (2.81 GB) and a competitive Score (0.149).

3.4 RAG

3.4.1 Generation Model

For the generation model, as the MCQA model, we tested two variants: one generating answers with reasoning (inspired by ChatGPT) and another predicting only the correct option. We also applied RAFT fine-tuning (Zhang et al., 2024a) to help the model better use retrieved documents.

To support this, we built a 10,000 example dataset from our STEM corpus using the GPT wrapper, prompting ChatGPT with one or two documents to generate questions and answers. The dataset follows this structure:

- **question**: a question generated from one or more documents in the corpus.
- **choices**: 4 answer options.
- **answer**: the correct answer (e.g., A, B, C, or D).
- **doc1**: the primary document used to generate the question.
- **doc2**: a secondary supporting document (optional).

Using this dataset, we fine-tuned our MCQA model by including the relevant documents directly in the prompt. This adaptation enables the generator to condition its answers on external knowledge retrieved at inference time. Additional training details are provided in Appendix J.

3.4.2 Retriever Model

For the retriever, we used the publicly available Alibaba-NLP/gte-modernbert-base model (Zhang et al., 2024b; Li et al., 2023), a BERT-based encoder optimized for dense retrieval tasks. To adapt it to our domain, we fine-tuned it using a *Triplet Loss* on synthetic triplets derived from our STEM corpus.

We constructed a training set of triplets $\mathcal{T} = \{(q_i, d_i^+, d_i^-)\}_{i=1}^N$, where each q_i is a query comprising a multiple-choice question with four answer choices, d_i^+ is the positive document used to generate the question, and d_i^- is a randomly sampled unrelated negative document.

The model learns an embedding function $f(\cdot)$ such that the cosine similarity between $f(q_i)$ and $f(d_i^+)$ is higher than that between $f(q_i)$ and $f(d_i^-)$. We finetune using the Triplet Loss (Appendix K).

To evaluate retrieval quality, we use the **Recall@k** metric. For each query q_i, the retriever returns the top-k most similar documents. (Appendix K).

3.4.3 Evaluation

Model	Accuracy
Qwen3-0.6B-Base + retriever	0.429 ± 0.018
MCQA + retriever	0.433 ± 0.018
MCQA_reasoning + retriever	0.412 ± 0.018
MCQA_RAFT + retriever	0.419 ± 0.017
MCQA + retriever_ft	**0.440 ± 0.018**

Table 5: RAG models accuracy on zechen-nlp/MNLP_STEM_mcqa_evals dataset. MCQA_RAFT is the MCQA model + the RAFT finetune, retriever is gte-modernbert-base and retriever_ft is the finetune gte-modernbert-base model.

Our RAG setup yielded minimal improvement: the best configuration (MCQA + fine-tuned retriever) reached 0.440 accuracy, slightly below the MCQA-only baseline (0.442). Several factors likely explain this:

Model size. Qwen3-0.6B-Base may be too small to effectively leverage retrieved context.

Corpus coverage. While STEM-focused, our 62k-document Wikipedia corpus remains broad and could benefit from more targeted sources such as textbooks or lecture notes.

RAFT limitations. A richer and more comprehensive training set could potentially lead to improved performance.

Despite this, the fine-tuned retriever **still improved results** over the base one (0.440 vs. 0.433), suggesting that with a richer corpus and better training data, RAG could yield stronger gains.

4 Analysis

4.1 Reward Model

The reward model was trained on student preference data collected at EPFL, aligning its outputs

with the expectations and learning styles specific to that academic context. This focus supports the goal of building an AI tutor specialized in EPFL course content. However, as student expectations can vary across institutions and cultures, the model's alignment may not transfer directly to other educational settings. Future work could explore incorporating preference data from a broader student population to improve adaptability and extend the tutor's relevance beyond EPFL.

4.2 MCQA

A qualitative comparison of our models with the baseline reveals that single-token accuracy may be a poor indicator of a model's value as a learning assistant. Across multiple test prompts, we found that while the single-token model was somewhat accurate, the lack of explanations limits its educational usefulness and verifiability. The Qwen3-0.6B-Base model, meanwhile, often gave inconsistent or repetitive answers, sometimes generating follow-up questions instead of responding properly. In contrast, the reasoning model produced concise, well-structured explanations and correctly answered some questions the others missed, thanks to its enhanced reasoning capabilities. Representative outputs are shown in Appendix G.

4.3 Quantization

In addition to the quantitative results, we observed that QLoRA's outputs remained qualitatively consistent with the baseline. In a few selected cases, it even chose more correct answers than the original model, likely due to a regularization effect from the LoRA fine-tuning.

On the other hand, static quantized models like BnB+4bit occasionally produced less confident or inconsistent outputs, especially on multi-step reasoning questions, suggesting that fine-tuning plays a key role in preserving robustness after aggressive quantization.

4.4 RAG

We make similar observations as in 4.2. We also notice that the fine-tuned RAFT model tends to rely on documents even when they are irrelevant, which could be addressed this by including non-relevant documents during training.

5 Ethical considerations

We discuss here some relevant ethical concerns raised by our work.

5.1 Model accessibility

As the model was fine-tuned solely on English data, it is inaccessible to non-English speakers, particularly those from underserved educational backgrounds. Expanding our training dataset to multiple languages could improve inclusivity, though it may require a larger model. For low-resource languages like Urdu or Swahili, a supplementary translation model could help address data scarcity.

5.2 Intended and unintended use

Though designed for education, the model could be misused, for instance, to cheat on exams or rely on answers without verification, leading to potential misinformation due to accuracy limits. However, even in its intended use, the model may impact human teaching assistants by introducing unfair competition, despite not matching the depth of human support.

5.3 Training data

All data we used for training originates from public datasets purposely crafted for LLM training. Therefore, we did not build our model on top of the work of nonconsenting people and see no major ethical implications in our choice of training data.

6 Conclusion

We built Perplexed Tutor, a compact and aligned AI assistant tailored for STEM multiple-choice questions. By combining supervised fine-tuning, preference alignment through Direct Preference Optimization (DPO), model quantization, and retrieval-augmented generation (RAG), we significantly improved performance and efficiency.

Fine-tuning on MCQA data led to a 4% accuracy gain over the base model, while DPO brought the model's outputs closer to student preferences with a reward accuracy of 73%. Using QLoRA, we reduced the model size by half and even slightly improved accuracy, making it the most effective compression method tested. Although RAG provided only modest gains, the fine-tuned retriever outperformed the baseline, suggesting further potential with a richer document corpus.

Despite constrained training resources, our models proved effective and deployable. Future work should focus on expanding training data, improving corpus quality, and evaluating the system in real-world educational settings.

References

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.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms.

Iddo Drori, Sarah Zhang, Zad Chin, Reece Shuttleworth, Albert Lu, Linda Chen, Bereket Birbo, Michele He, Pedro Lantigua, Sunny Tran, Gregory Hunter, Bo Feng, Newman Cheng, Roman Wang, Yann Hicke, Saisamrit Surbehera, Arvind Raghavan, Alexander Siemenn, Nikhil Singh, Jayson Lynch, Avi Shporer, Nakul Verma, Tonio Buonassisi, and Armando Solar-Lezama. 2024. A dataset for learning university stem courses at scale and generating questions at a human level. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(13):15921–15929.

Nathan Felber. 2025. Mnlp_m3_mcqa_dataset (revision 1a934d6).

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

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

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021b. Measuring mathematical problem solving with the math dataset.

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.

Tushar Khot, Peter Clark, Michal Guerquin, Peter Jansen, and Ashish Sabharwal. 2020. Qasc: A dataset for question answering via sentence composition.

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. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.

Zehan Li, Xin Zhang, Yanzhao Zhang, Dingkun Long, Pengjun Xie, and Meishan Zhang. 2023. Towards general text embeddings with multi-stage contrastive learning. *arXiv preprint arXiv:2308.03281*.

Pan Lu, Swaroop Mishra, Tony Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. 2022. Learn to explain: Multimodal reasoning via thought chains for science question answering.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model.

Qwen Team. 2025. Qwen3 technical report.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. 2024. Smoothquant: Accurate and efficient post-training quantization for large language models.

Tianjun Zhang, Shishir G. Patil, Naman Jain, Sheng Shen, Matei Zaharia, Ion Stoica, and Joseph E. Gonzalez. 2024a. Raft: Adapting language model to domain specific rag.

Xin Zhang, Yanzhao Zhang, Dingkun Long, Wen Xie, Ziqi Dai, Jialong Tang, Huan Lin, Baosong Yang, Pengjun Xie, Fei Huang, et al. 2024b. mgte: Generalized long-context text representation and reranking models for multilingual text retrieval. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 1393–1412.

A AI Usage

We used ChatGPT throughout the project as a tool to help us understand concepts, debug Python code and reformulate sentences. However, all key contributions have primarily been made by the team, and all information has been verified and traced back to the original sources. We also used ChatGPT as a post-processing tool for our MCQA dataset as described in section 3.2.1.

B DPO training visualization

Figs. 1 and 2 show respectively the accuracy and margins during the training of the reward model.

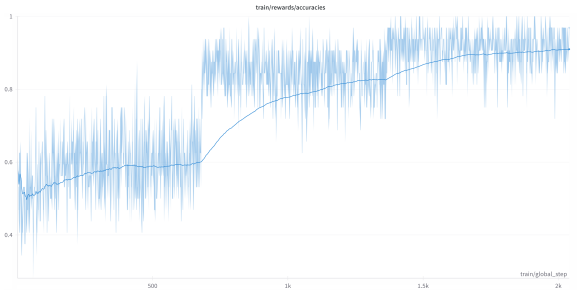

Figure 1: The training accuracies of the DPO model over 3 epochs

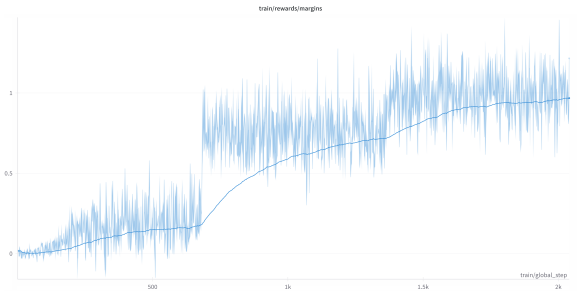

Figure 2: The training margins of the DPO model over 3 epochs

C Additional DPO Experiments and Results

Additional DPO training was conducted on the same dataset, Titantek/MNLP_M3_dpo_dataset, using 25k samples from the argilla/ultrafeedback-binarized-preferences-cleaned dataset (available on Hugging Face as Titantek/MNLP_M3_dpo_dataset_ultrafeedback).

We also performed DPO training on Titantek/MNLP_M3_dpo_dataset using the cDPO method.

The results of these experiments are presented in Table 6.

Reward Model	Reward Accuracy
DPO_ULTRA	0.6379 ± 0.0098
cDPO	0.6391 ± 0.0098

Table 6: Reward model accuracy DPO_ULTRA and cDPO

D ChatGPT prompts for dataset generation

The following instruction (1) and prompt (2) were used to post-process our initial aggregated dataset with the ChatGPT API.

Listing 1: Instruction used for MCQ reformulation.

```
You are a very good and helpful tutor
responsible for producing reference solutions
of course-related questions to help students
learn. Do not talk to the user, always get to
the answer directly.
```

Listing 2: and [END_ANSWER] tokens allow us to parse the output and extract the different parts.]Prompt used for MCQ reformulation. The {entry.question} and {entry.answer} placeholders are substituted for the actual question and answer for each dataset entry. The [END_QUESTION] and [END_ANSWER] tokens allow us to parse the output and extract the different parts.

```
Reformulate the following question and solution
to fit the given format. If the question is not
a multiple choice question, reformulate it as
an MCQ and create a set of plausible concise
options including correct option(s). The
reasoning should start by shortly explaining
the question, then breaking down the problem
step by step and be helpful to the student. It
must be correct with respect to the given
solution. The rationale should explain why the
solution is correct and be clear and concise.
Use [END_QUESTION] to separate the question
from the solution. Finish your answer with
[CORRECT_ANSWER] followed by the id of the
correct answer (e.g. A, B, C, or D).

Question:

{entry.question}

Solution:

{entry.answer}

Format:

<multiple choice question>[END_QUESTION]

### Question Breakdown:

<reasoning>
```

```
### Correct Solution:

<solution>

### Rationale:

<rationale>

[CORRECT_ANSWER]<id>
```

E MCQA training examples

Listings 3 and 4 show training examples for supervised fine-tuning of the MCQA models.

Listing 3: Reasoning training example.

```
What is the least positive integer that is
divisible by three distinct prime numbers?

A) 12
B) 18
C) 30
D) 42

### Question Breakdown:

The question asks for the smallest positive
integer that can be divided evenly by three
different prime numbers. To solve this, we need
to identify the three smallest prime numbers
first, then calculate their product.

### Correct Solution:

The three smallest prime numbers are 2, 3, and
5. We multiply these together:
\(2 \times 3 \times 5 = 30\). Thus, the least
positive integer that is divisible by these
three distinct primes is 30.

### Rationale:

The solution is correct because any integer
that is divisible by multiple primes must be at
least the product of those primes, as stated by
the Fundamental Theorem of Arithmetic. Since we
used the three smallest primes, their product
gives us the smallest possible number that
meets the criteria.
```

Listing 4: Single token answer training example.

```
What is the least positive integer that is
divisible by three distinct prime numbers?

A) 12
B) 18
C) 30
D) 42

C
```

F MCQA training losses

Figs. 3 and 4 show the training loss of our MCQA models during supervised fine-tuning.

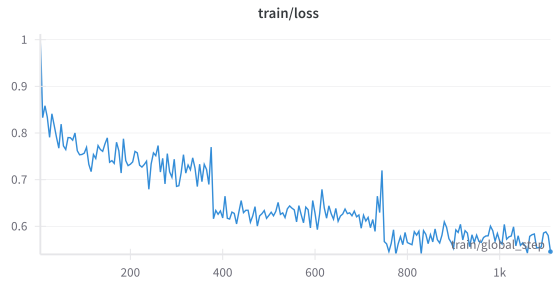

Figure 3: The training loss of the MCQA model fine-tuned on reasoning answers over 3 epochs.

Figure 4: The training loss of the MCQA model fine-tuned on single token answers over 3 epochs.

G Example answers from MCQA models

Three representative example outputs for the same question from our evaluated MCQA models are shown in Listings 6, 5 and 7. The Qwen3-0.6B-Base model is wrong and unhelpful, the single token model is correct but unhelpful, and the reasoning model is both correct and helpful.

Listing 5: Example Qwen3-0.6B-Base model output.

```
Complete the sentence: Deep-frying chicken is a
().

A. chemical change
B. physical change
Answer:

B. physical change

What is the name of the process that occurs
when a substance is heated to a high
temperature and then cooled?
Answer:

sublimation
```

```
What is the name of the process that occurs
when a substance is heated to a high
temperature and then cooled?
Answer:

sublimation

What is the name of the process that occurs
when a substance is heated to a high
temperature and then cooled?
Answer:

sublimation

What is the name of the process that occurs
when a substance is heated to a high
temperature and then cooled?
Answer:

sublimation

What is the name of the process that occurs
when a substance is heated to a high
temperature and then cooled?
Answer:

sublimation

What is the name of the process that occurs
when a substance is heated to a high
temperature and then cooled?
Answer:

sublimation<|endoftext|>
```

Listing 6: Example single token answer model output.

```
Complete the sentence: Deep-frying chicken is a
().

A. chemical change
B. physical change
Answer:

A<|endoftext|>
```

Listing 7: Example reasoning model output.

```
Complete the sentence: Deep-frying chicken is a
().

A. chemical change
B. physical change
Answer:

### Question Breakdown:

This question asks to identify the type of
change that occurs when deep-frying chicken. It
is important to understand the difference
between chemical and physical changes. A
chemical change involves a transformation that
alters the substance's chemical structure,
while a physical change does not.

### Correct Solution:

A. chemical change

### Rationale:

Deep-frying chicken involves a process where
heat causes the proteins and fats in the
chicken to undergo chemical reactions, leading
to changes in texture and flavor. This results
in a new substance that is not the same as the
original chicken, indicating that it is a
chemical change.<|endoftext|>
```

H ChatGPT prompts for RAFT dataset generation

Listing 8: Example single token answer model output.

```
You are a STEM exam question generator.
Generate 1 multiple-choice question combining
the knowledge from multiple documents.

Rules:
- STEM only.
- 4 choices labeled A, B, C, D.
- Provide correct answer as single letter.
- Output strictly JSON: question, choices,
answer.

Input documents: <docs>

Output format:
{{
"question": "...",
"choices": {{"A": "...", "B": "...", "C":
"...", "D": "..."}},
"answer": "<A|B|C|D>" (choose one, only one
correct answer, no explanations or reasoning
needed just the letter)
}}
```

I RAFT training examples

Examples of training input for RAFT model.

Listing 9: RAFT training example.

```
Profilin-2 is a protein that in humans is
encoded by the PFN2 gene.

The protein encoded by this gene is a
ubiquitous actin monomer-binding protein
belonging to the profilin family. It is thought
to regulate actin polymerization in response to
extracellular signals. There are two
alternatively spliced transcript variants
encoding different isoforms described for this
gene.

Interactions
PFN2 has been shown to interact with ROCK1,
Vasodilator-stimulated phosphoprotein, CCDC113
and FMNL1.



Stearoyl-CoA is a coenzyme involved in the
metabolism of fatty acids. Stearoyl-CoA is an
18-carbon long fatty acyl-CoA chain that
participates in an unsaturation reaction. The
reaction is catalyzed by the enzyme
```

```
stearoyl-CoA desaturase, which is located in
the endoplasmic reticulum. It forms a
cis-double bond between the ninth and tenth
carbons within the chain to form the product
oleoyl-CoA.

The following are multiple choice questions
(with answers) about knowledge and skills in
advanced master-level STEM courses.


What is the primary function of the protein
encoded by the PFN2 gene?
A. Facilitating lipid metabolism
B. Regulating actin polymerization
C. Encoding DNA repair enzymes
D. Transporting oxygen in blood
```

J RAFT training

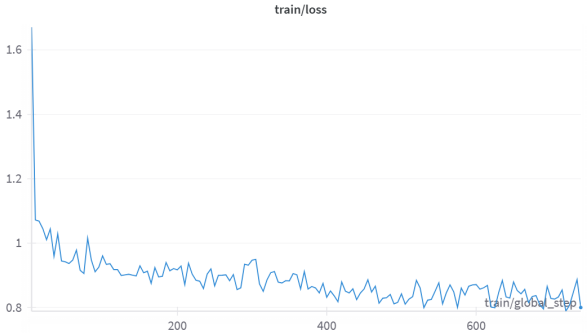

Figure 5: The training loss of the MCQA model fine-tuned with RAFT over 2 epochs.

K Retriever training

Triplet Loss:

$$\mathcal{L}_{\text{triplet}} = \sum_{i=1}^{N} \max\left(0,\ \text{sim}(f(q_i), f(d_i^-)) - \text{sim}(f(q_i), f(d_i^+)) + \alpha\right)$$

where $\text{sim}(x, y) = \frac{x \cdot y}{\|x\|\|y\|}$ denotes cosine similarity, and $\alpha > 0$ is a margin that enforces a minimum separation between positive and negative pairs.
Recall@k is defined as:

$$\text{Recall@}k = \frac{1}{N} \sum_{i=1}^{N} 1\left[d_i^+ \in \text{Top-}k(q_i)\right]$$

It measures the proportion of queries where the correct document appears among the top-k retrieved. Higher recall@k indicates better retrieval accuracy. It allows us to choose the best value for the top-k parameter for the inference.

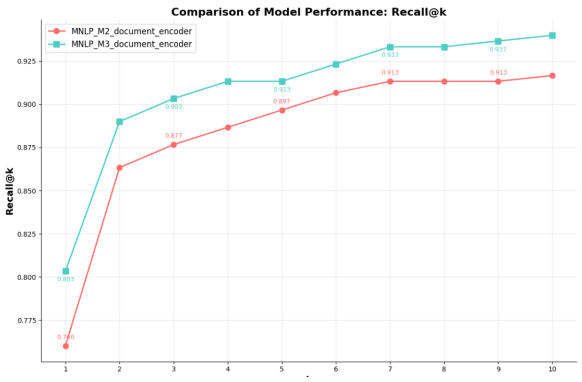

Figure 6: Comparison of recall@k between retriever before and after finetuning.