# EPFL TA Meister

Malak Lahlou Nabil | 329571 | malak.lahlounabil@epfl.ch
Sara Anejjar | 329905 | sara.anejjar@epfl.ch
Frederic Khayat | 326634 | frederic.khayat@epfl.ch
Peter Abdel Massih | 325664 | peter.abdelmassih@epfl.ch
4-pack

## Abstract

**Generative language models like ChatGPT have become popular online helpers. Building on these advancements, we present the development and adaptation of an educational chatbot named EPFL TA Meister, designed specifically for students in science and engineering fields. Our approach involves a three-stage pipeline: pre-training with language modeling, supervised fine-tuning (SFT), and Direct Preference Optimization (DPO). These latters improve the model's performance going for an alignment of 0.692 to 0.848 on course datasets. We also explore various quantization techniques, including 8-bit and 4-bit quantization, GPTQ, and Adaptive Weight Quantization (AWQ), to optimize the model for deployment. Our methodology includes Retrieval-Augmented Generation (RAG) to improve question-answering performance. This latter still needs to be improved by adding pre and post-processing for retrieval documents. Our main findings show that the 8-bit quantized model performs almost as well as the base model with much better efficiency, while 4-bit quantization methods like GPTQ maintain higher accuracy compared to others.**

## 1 Introduction

The latest generative Large Language Models (LLMs) have shown impressive capabilities in engaging users through human-like conversation, demonstrating vast knowledge across numerous topics. These chatbots hold great potential in education by providing quick, personalized answers to student queries, easing the burden on teaching staff. However, many current chatbots are proprietary, limiting transparency regarding their internal workings, training methods, and data privacy practices.

In this project, we are developing an educational chatbot, EPFL TA Meister, using open-source resources and limited computational power. We employ open-source datasets and models, using training techniques like supervised fine-tuning (SFT) and a new method called Direct Preference Optimization (DPO).

Reinforcement learning by human feedback (RLHF) was crucial for the success of chatbots like ChatGPT, but it's a complex process involving reward model fitting and reinforcement learning. DPO, introduced by Stanford University, simplifies this by bypassing explicit reward estimation and directly optimizing the language model using preference data through a single maximum likelihood objective. This approach combines the language model and reward model into a single policy network, making the training process more straightforward and efficient.

## 2 Related Work

### 2.1 Direct Preference Optimization (DPO)

Prior to the development of Direct Preference Optimization (DPO), language model fine-tuning commonly employed a separate model known as the reward model or Reinforcement Learning from Human Feedback (RLHF) model (Ouyang et al., 2022). This method, based on Proximal Policy Optimization (PPO) (Schulman et al., 2017), involved sampling completions from large language models (LLM) and having the reward model evaluate each completion. Given the high costs and logistical complexities of human evaluation, the reward model was designed to approximate human feedback economically and effectively (Schulman et al., 2017). Building on this foundational approach, DPO enhances the fine-tuning process by directly optimizing model parameters based on human preferences derived from pairwise comparisons of model outputs. Research by Rafailov et al. (Rafailov et al., 2023) has demonstrated how this methodology significantly improves response relevance and quality. Furthermore, the integration of the Iterative Reasoning Preference Optimization (IRPO) algorithm, which combines DPO loss with negative likelihood loss (NLL), offers further improvements in model accuracy (Pang et al., 2024).

### 2.2 Retrieval-Augmented Generation (RAG)

Innovative approaches in this field include the Corrective RAG (CRAG) approach, which rectifies inconsistencies in retrieved content (Smith et al., 2024), and Retrieval-Augmented Fine-Tuning (RAFT), which enhances model adaptability during the training phase (Jones et al., 2024). Additional strategies such as Reason and Action (ReAct), integrate causality and actionable insights into the response generation process (Lee et al., 2024b), while Self-Reflective RAG evaluates its performance and adapts strategies autonomously (Kim et al., 2024). RAG Fusion combines multiple RAG models to synthesize a broader range of data and perspectives (Garcia et al., 2024), and function or tool calling during inference dynamically utilizes external computational resources or databases (Martinez et al., 2024). Temporal Augmented Retrieval (TAR) incorporates time-aware dynamics to enhance the relevance of retrieved information based on temporal contexts (Nguyen et al., 2024). These methodologies collectively enhance our ability to decompose complex queries into simpler subqueries, dynamically reformulate queries based on initial retrieval results.

### 2.3 Model quantization

Quantization techniques have significantly advanced, reducing computational demands by transforming high-precision data into lower-bit representations. Notably, GPTQ by Frantar et al. (Frantar et al., 2023) introduced post-training quantization tailored for generative pre-trained transformers, significantly reducing bit-width to 3-4 bits and experimenting down to 2-bit and ternary quantization. SmoothQuant by Xiao et al. (Xiao et al., 2023a) furthered this by proposing a post-training quantization framework that migrates difficulty from activations to weights, optimizing the quantization process for W8A8 configurations.

Further innovations include Activation-aware Weight Quantization (AWQ) by Lin et al. (Lin et al., 2024), targeting edge devices with a low-bit weight-only quantization method and focusing on protecting salient weights through activation monitoring to retain performance. Additionally, Outlier-Aware Weight Quantization (OWQ) by Lee et al. (Lee et al., 2024a) minimizes the memory footprint through structured weight prioritization and applies high precision to critical subsets.

Lastly, BitDistiller by Du et al. (Du et al., 2024) leverages self-distillation to enhance performance in sub-4-bit configurations, utilizing a tailored asymmetric quantization and clipping technique combined with a Confidence-Aware Kullback-Leibler Divergence as the distillation loss. Finally, Quantized Low-Rank Adaptation (QLORA) (Dettmers et al., 2023) merges low-rank adaptation with quantization to efficiently reduce model size and computational overhead while maintaining performance.

## 3 Approach

### 3.1 Data preprocessing

We have used multiple datasets to train and evaluate our model, as described in Table 1. Most of the datasets are from Hugging Face. To add to our database, we generated two datasets in the following ways:

- **M1 - EPFL preference dataset (25.8K)**
  This dataset contains preference pairs collected using prompted ChatGPT 3.5 to integrate human feedback into the model. The goal is to compare outputs generated from two different prompts: one emphasizing chain-of-thought and the other one with minimal instruction as described below.

  Prompt A - Chain-of-Thought

  {question}

  —

  [if Options available] Provide a step by step solution to the problem.

  —

  Let's think step by step. First work out your own solution to the problem. Then compare your solution to the given options and choose the correct one. Don't choose an option until you have done the problem yourself.

  Prompt B - Minimal Instruction

  {question}

  —

  [If Options available] Answer the question.

  —

  Answer the question, then choose the correct option

  We evaluate these outputs across multiple categories such as Correctness and Clarity and use

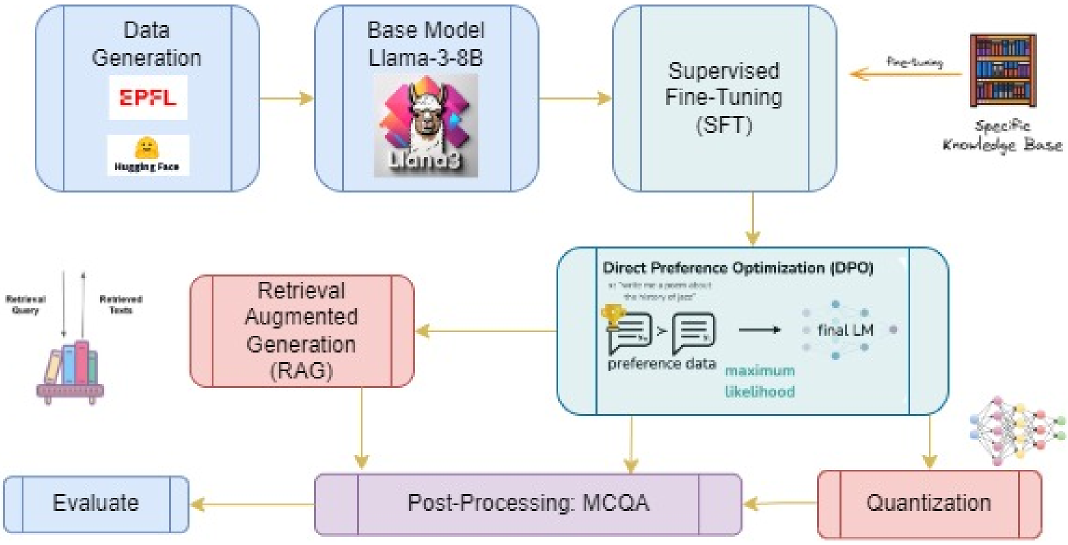

Figure 1: Model pipeline

this dataset for fine-tuning our model, along with training using DPO.

- **Stack Exchange (43K)**: We created a DPO dataset from the StackExchange database (Internet Archive, 2024), which includes forum posts on STEM topics like computer science, physics or mathematics. The dataset's "prompt" part combines the title and content of each post. For "chosen" and "rejected" answers, we selected questions with at least two differently scored answers, based on user "likes".

### 3.2 Supervised Fine-Tuning (SFT) and Direct Preference Optimization (DPO)

Based on insights from Rafailov et al. (2023) (Rafailov et al., 2023), integrating Supervised Fine-Tuning (SFT) prior to employing Direct Preference Optimization (DPO) is critical to enhancing system performance. We used the ‘SFTTrainer‘ from the ‘trl‘ package (von Werra et al.), selecting parameters that optimize computational efficiency and prevent out-of-memory (OOM) errors. Here, we provide a detailed justification for these choices:

- Based on insights from Rafailov et al. (2023) (Rafailov et al., 2023), integrating Supervised Fine-Tuning (SFT) prior to employing Direct Preference Optimization (DPO) is critical to enhancing system performance.
- We used the ‘SFTTrainer‘ from the ‘trl‘ package (von Werra et al.).
- For DPO, consistent general parameters with SFT were maintained using the ‘DPOTrainer‘ from trl, aligned with preference data.
- Following the Hugging Face alignment handbook, the learning rate for DPO was set significantly lower than SFT.
- The SFT checkpoint served as the reference model, supplemented by newly configured LoRA layers.
- A sigmoid loss function, as discussed in (Rafailov et al., 2023), was employed.

The DPO loss function used to optimize the policy is given by:

$$\mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = -E_{(x,y_w,y_l)\sim\mathcal{D}}\Bigg[ \log\sigma\left(\beta\log\frac{\pi_\theta(y_w\mid x)}{\pi_{\text{ref}}(y_w\mid x)} - \beta\log\frac{\pi_\theta(y_l\mid x)}{\pi_{\text{ref}}(y_l\mid x)}\right)\Bigg] \quad (1)$$

This equation compares the policy probabilities of the old policy ($\pi_{\text{ref}}$) and the new policy ($\pi_\theta$) for a winning completion ($y_w$) and a losing completion ($y_l$). Optimizing this function encourages the new policy to favor winning responses over losing ones.

### 3.3 Quantization Approaches

To optimize our model for deployment, we explored four different quantization methods: 8-bit quantization, 4-bit quantization using BitsAndBytes, GPTQ, and AWQ. Each method was evaluated for its effectiveness in maintaining model performance while reducing computational requirements.

**8-bit Quantization:** We began with 8-bit quantization, utilizing the BitsAndBytesConfig from the ‘trl‘ package (Hu et al., 2021). An outlier threshold of 6.0 was applied to handle high precision layers, while other layers were quantized to 8-bit integers. This approach was chosen to balance the need for precision with the benefits of reduced memory usage. The high precision layers that exceeds the threshold remained in FP16 to ensure that the model's critical operations maintained their accuracy. Moreover, important scaling layers are kept in fp32 to ensure numerical stability.

**4-bit Quantization:** Next, we implemented 4-bit quantization using the BitsAndBytes library (Kalajdzievski et al., 2023). This method involved double quantization and the use of NF4 quantization type, which leverages weights from a normal distribution. We utilized bfloat16 for the computation of the quantized weights to enhance computational efficiency while preserving model accuracy. This method allowed us to significantly reduce the memory footprint without compromising performance.

**GPTQ Quantization:** For the GPTQ approach, we defined a configuration with 4-bit quantization and a group size of 128 to optimize memory usage and computational efficiency (Frantar et al., 2023). We enabled descent activation and set a damping percentage of 0.1. The model was loaded onto the CPU to maximize GPU availability for the quantization process. The main equation for GPTQ involves minimizing the quantization error, typically represented as:

$$\min_{Q} \|W - Q(W)\|_F^2 \tag{2}$$

where $W$ is the weight matrix, $Q(W)$ is the quantized weight matrix, and $\| \cdot \|_F$ denotes the Frobenius norm. This method was particularly chosen to handle large-scale models with minimal performance degradation.

**AWQ Quantization:** Lastly, we utilized AWQ (Adaptive Weight Quantization) with a configuration that included zero-point quantization, a group size of 128, and a 4-bit weight quantization (Lin et al., 2024). This method involved calibrating the model with a subset of training data to fine-tune the quantization parameters. The main equation for AWQ, focusing on minimizing the reconstruction error, is given by:

$$\min_{Q} \sum_{i=1}^{n} \|W_i - Q(W_i)\|_2^2 \tag{3}$$

where $W_i$ represents the weights of the $i$-th layer, and $Q(W_i)$ represents the quantized weights of the $i$-th layer. This approach allowed us to adaptively quantize the model weights while maintaining high accuracy and efficiency.

**Data Preparation:** For both GPTQ and AWQ methods, we used the same data as for Supervised Fine-Tuning (SFT). The datasets included a combination of various sources, such as competition math datasets, StackExchange data, physics datasets, code feedback datasets, and subsets of the Wikitext (Merity et al., 2016) dataset. This diverse data set was essential for ensuring the robustness and generalizability of the quantized models. The transformation functions used to prepare the datasets included converting the examples into a text format suitable for training and quantization. These functions ensured that the model could learn effectively from the provided data, optimizing its performance across different tasks.

By experimenting with these four quantization methods, we were able to identify the approach that best balanced computational efficiency and model performance. The implementation of these methods allowed us to deploy a highly efficient model capable of handling large-scale data with reduced memory requirements.

## 3.4 Retrieval-Augmented Generation (RAG)

Retrieval-Augmented Generation (RAG) leverages external knowledge sources to enhance the model's ability to handle complex queries. This technique, significantly advanced by contributions from Lewis et al. (Lewis et al., 2021) facilitates iterative reasoning across multiple documents.

In order to implement RAG, multiple choices had to be made. Below are the justification of our choices:

1. **Library:** The most common libraries used to implement RAG are HuggingFace (Hugging Face, 2024), LangChain (LangChain, 2024), Llama-Index (Llama-Index, 2024), and Haystack (deepset GmbH, 2024). We chose to use LangChain because of its integration and flexibility.

2. **Embedding model:** We chose to use the "BAAI/bge-base-en-v1.5" (Xiao et al., 2023b)

embedding model to transform text into embeddings. After extensive research, we concluded that this model had a very good performance while being fairly small (109M of parameters). Furthermore, it ranks as one of the best on the MTEB leaderboard (Muennighoff et al., 2022) on retrieval tasks.

3. **Database:** We chose to store our chunk embeddings using FAISS, a database specifically designed for high-performance similarity search and retrieval. FAISS excels in handling large-scale vector data, making it ideal for this type of application.

4. **Data collection:** We populated our database using a combination of PDF STEM manuals and datasets from Hugging Face. (Table 1)

   For PDF files, we utilized PDFMiner (Shinyama) to extract text, which we then segmented into chunks of up to 900 characters. This segmentation was done while considering sections and subsections divisions in the PDF to avoid splitting important information between chunks. We also added an 80-character overlap between chunks to make the division flexible when necessary. We chose this chunk size because we believe that 900 characters are enough to deliver the necessary context while avoiding unnecessary information.

   Hugging Face datasets were filtered and processed to ensure each question-answer pair was within that chunk size. Each chunk was then added to the database while preserving the document's source metadata (Douze et al., 2024).

5. **Context selection:** The goal of the RAG system is to enhance prompts with contextual information to help the model generate more accurate and relevant responses. Our model has a context window size of 8192 (AI@Meta, 2024), meaning both the input and output must stay within this limit. To provide a margin of safety, we limit the size of the prompt to one-quarter of the model's limit, which equates to 2048 tokens (About 8000 characters (openAI)).

   To ensure that the added contexts are relevant to the question, we discard any context with an L2 distance higher than 0.40. This threshold was determined experimentally using our database. Keeping the context sizes small and allowing for multiple contexts helps our model gather information from different parts of the same document or from different documents, thereby enhancing the relevance and helpfulness of the added contexts.

### 3.5 MCQA Retrieval

For the final output of our model, we adopted a post-processing approach to extract the letter corresponding to the correct answer for multiple-choice questions. We designed a prompt that encourages a chain-of-thought process before delivering the correct answer.

Basic Prompt Template

{question}
—
Answer the above question. First provide an explanation, then clearly state which letter corresponds to the right answer.
Explanation:

When running our model with RAG, we adapt the prompt to take into account the documents we provided.

RAG Prompt Template

Answer the question. You may utilize the following context:
{context}
—
{question}
—
Answer the above question. First give an explanation to your answer, then clearly state which letter corresponds to the right answer. You may use the above context if you find it helpful.
Explanation:

Given the output of the model for this prompt, we identify answer options and search for specific patterns within the model's response. By leveraging these techniques, we ensure accurate evaluation of the model's performance.

## 4 Experiments

### 4.1 Data:

All the datasets used for this project are described in the Table 1.

Table 1: Summary of SFT, DPO, and RAG datasets

| Dataset | Purpose | SFT Size | DPO Size | RAG Size |
|---|---|---|---|---|
| M1 preference dataset | Questions from various EPFL courses | 25.8K | 25.8K | - |
| Stack Exchange (Internet Archive, 2024) | Dataset from multiple stack exchange forums | 43K | 33.4K | - |
| Mathematical Reasoning (Hendrycks et al., 2021b) | Complex mathematical problems | 10K | 2.4K | - |
| Programming Feedback (M-A-P et al., 2022) | Multiple programming tasks | 10K | - | - |
| Physics (Alves et al., 2021) | Physics questions with generated answers | 10K | - | - |
| Math DPO (argilla, 2024) | Math questions and answers | - | 2.4K | - |
| Python DPO (jondurbin, 2024) | Multiple Python programming tasks | - | 9K | - |
| STEM DPO (thewordsmiths, 2024) | Multiple questions from STEM fields | - | 30K | - |
| Math QA (Yu et al., 2023) | Q&A for mathematical questions | - | - | 275K |
| Python QA (Language, 2023) | Q&A for python questions | - | - | 21K |
| STEM Books | Books taken from the EPFL Drive | - | - | 40 |

We opted for these datasets since they were similar to courses at EPFL in maths, computer science, and physics.

## Data Format Explanation

1. `Supervised Fine-Tuning (SFT)`: We loaded datasets using the Huggingface Datasets library and converted them into the instruction format. We used "prompt": "...", "completion": "..." where `"prompt"` represents the input prompt and `"completion"` represents the corresponding output or completion.

2. `Direct Preference Optimization (DPO)`: For DPO, we reformatted our datasets into triplets: "prompt": "...", "chosen": "...", "rejected": "..." where `"prompt"` is the input prompt, `"chosen"` is the preferred option or completion, and `"rejected"` is the rejected option.

3. `Quantization for GPTQ and AWQ`: We used the format "text": ... where each text field contains some part of the wikitext-2-raw-v1 and parts of our datasets for SFT by concatenating the question and answer (or prompt and completion). This is done to make sure the weights are still optimal for our use case of a teaching assistant.

4. `RAG database`: This database contains vector embeddings of relevant documents and their prompts. These embeddings are high-dimensional representations of the semantic meaning and context of the text given as input strings. Our database size is 2.7 GB.

### 4.2 Evaluation method:

To quantify the improvements of DPO, we used three metrics on our benchmarks:

1. **Accuracy:** Percentage of pairs where

$$\pi_\theta(y_w|x)/\pi_{\text{ref}}(y_w|x) > \pi_\theta(y_l|x)/\pi_{\text{ref}}(y_l|x) \tag{4}$$

2. **DPO Model Alignment:** Percentage of pairs where

$$\pi_\theta(y_w|x) > \pi_\theta(y_l|x) \tag{5}$$

3. **Ref. Model Alignment:** Percentage of pairs where

$$\pi_{\text{ref}}(y_w|x) > \pi_{\text{ref}}(y_l|x) \tag{6}$$

Additionally, we evaluated the model using accuracy on the final Multiple Choice Question Answering (MCQA) dataset. This metric measures the percentage of correct answers provided by the model out of the total number of questions.

### 4.3 Baselines:

We chose Meta Llama 3-8B (AI@Meta, 2024) as the baseline of our model, trained on over 15T tokens. The robust capabilities of Llama 3-8B ensure that we are benchmarking against a high standard, thus validating the effectiveness and efficiency of our methods.

### 4.4 Experimental details:

To ensure optimal performance and manage computational resources effectively, we applied the following model configurations:

**Model Configurations**

- **Learning Rate Adjustments:** We began with a learning rate of 2e-4. However, both DPO and SFT showed a significant jump in evaluation loss initially. Due to the initial instability, we adjusted the learning rate to 2e-5, which resulted in more stable training. For DPO, the learning rate was set significantly lower, reduced by 20-fold from 2e-5 to 1e-6.
- **Max Gradient Norm and Warm-Up Ratio:** Despite reducing the max gradient norm and warm-up ratio, we encountered disproportional gradient values, indicating the need for further fine-tuning.
- Following recommendations from Hu et al. (2021) (Hu et al., 2021), we set *alpha* to 128 and *rank* to 64, adhering to the guideline that *alpha* should be twice the *rank*. Additionally, Rank-Stabilized Adapters (rsLoRA) were employed to adjust adapter weights with a scaling factor $\gamma_r = \frac{\alpha}{\sqrt{\text{rank}}}$ (Kalajdzievski et al., 2023).
- LoRA adapters were attached to key linear layers involved in data transformation and attention mechanisms.
- **Batch sizes** of 4 for SFT and 2 for DPO were chosen to manage memory limits effectively.
- **Gradient accumulation** was implemented over four steps, coupled with **gradient checkpointing** every four steps to optimize memory usage (Chen et al., 2016).
- **Mixed precision training** was utilized, employing BF16 for weights and TF32 for matrix multiplications to enhance computational speed and efficiency (Micikevicius et al., 2018).
- **Flash Attention 2** was integrated to accelerate attention computations and reduce memory consumption (Dao, 2023).
- Efficient memory management was further supported by incorporating **DeepSpeed ZeRO stage 2 (partition of gradients)** with CPU Offload (Rajbhandari et al., 2020).
- Backward processing was facilitated by the **Accelerate** library(Gugger et al., 2022).
- A beta parameter of 0.1 to manage alignment strength and minimize divergence from the reference model.

**Training Environment**

The training process was distributed across both the SCITAS cluster and Google Colab, leveraging the capabilities of an NVIDIA A100 GPU mainly used for model training and a V100 on the SCITAS cluster for the construction of the RAG database. We leveraged around 200 hours of A100 training time and 10 hours of V100 to create contextual embeddings.

**Observations**

Despite the initial challenges with learning rates and gradient values, our experiments yielded better results with both DPO and SFT models after the adjustments. The final configurations led to improved stability and performance in our models, demonstrating the importance of careful parameter tuning in training deep learning models.

## 4.5 Results:

The results of our experiments are summarized in the tables below.

### 4.5.1 SFT + DPO

Table 2 showcases the result of our fine-tuning. It compares the alignment of the fine-tuned model with the alignment of the base model on a subset of the M1 dataset not trained upon (around 800 samples). Here, the alignment is defined as the percentage of pairs where $\pi(y_w|x) > \pi(y_l|x)$.

| | **DPO Align.** | **Reference Align.** |
|---|---|---|
| **Example** | 0.848 | 0.692 |
| **M1** | 0.633 | 0.554 |
| **Orca-Math** | 0.978 | 0.973 |
| **Datascience** | 0.847 | 0.839 |

Table 2: Alignment scores across various datasets. DPO Align. is the policy model (fine-tuned Llama3 with SFT and DPO) and Reference Align. is Llama3 base (without any fine-tuning)

### 4.5.2 Quantization

Table 3 presents the performance of our quantized models on our MCQA dataset compromising stem questions mainly in math, computer science and physics of size 1690 samples based on the MMLU dataset (Hendrycks et al., 2021a).

| | **Accuracy** |
|---|---|
| **No quantization (32-bits)** | 0.449 |
| **bitsandbytes 8-bit** | 0.448 |
| **bitsandbytes 4-bits** | 0.417 |
| **GPTQ 4-bits** | 0.428 |
| **AWQ 4-bits** | 0.404 |

Table 3: Accuracy before and after quantization

### 4.5.3 RAG

Table 4 displays the accuracy before and after using RAG.

| | Accuracy |
|---|---|
| **Base model** | 0.449 |
| **Base model + RAG** | 0.432 |

Table 4: Accuracy with and without RAG

## 5 Analysis

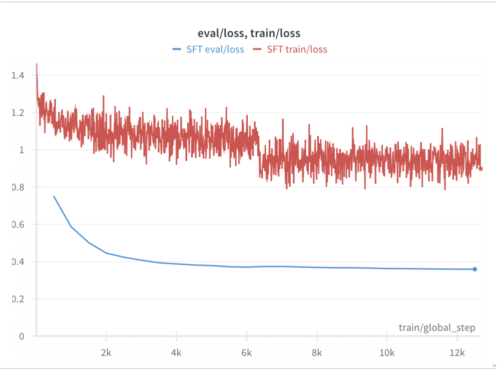

Figure 2: SFT training and evaluation loss

Figure 2 illustrates the training and evaluation loss curves for the Supervised Fine-Tuning (SFT) process over 2 epochs.

- **Training Loss (Red Line):** Starts at approximately 1.4, decreases rapidly in the first epoch, and stabilizes around 1.0 with minor fluctuations in the second epoch.
- **Evaluation Loss (Blue Line):** Begins at around 0.6, steadily decreases in the first epoch, and stabilizes at approximately 0.4 in the second epoch.

The significant reduction in both losses during the initial phase highlights the model's effective learning. The subsequent stabilization suggests minimal improvements with further training, indicating convergence.

The plot 3 shows the evaluation loss over training steps for the Direct Preference Optimization (DPO) model. Initially (1k to 2k steps), the evaluation loss increases from 1.02 to 1.08, indicating initial instability. From 2k to 4k steps, the loss declines, demonstrating effective learning. Between 4k and 8k steps, the loss stabilizes around 1.03 with minor fluctuations, suggesting gradual improvement. Using multiple dataset types might have caused issues due to heterogeneous preferences from different subjects. Although we shuffled the dataset to promote continuous learning, this may not be optimal. Overall, the downward trend in evaluation loss indicates learning and improvement, despite initial instability and minor fluctuations.

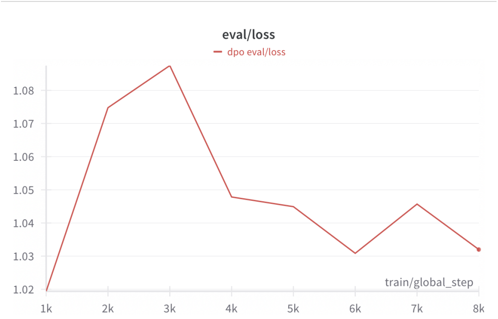

Figure 3: DPO evaluation loss

### 5.1 Quantization

Table 3 demonstrates the accuracy after quantizing our base model with four different methods. As expected, the accuracy drops as the model size diminishes. However, one surprising result is that the 8-bit quantized model (Dettmers et al., 2021) is almost as good as the base model: we get much better efficiency while preserving the performance of the original model.

Without any quantization, the model's accuracy is 0.449, serving as the baseline. The 8-bit quantization using bitsandbytes results in an accuracy of 0.448, almost identical to the baseline, indicating negligible impact on accuracy and highlighting its efficiency in maintaining performance while reducing computational and memory overhead.

When moving to 4-bit quantization, a noticeable reduction in accuracy is observed. The bitsandbytes method yields an accuracy of 0.417, indicating a significant drop. GPTQ 4-bit, however, achieves a higher accuracy of 0.428, making it more effective than bitsandbytes at this quantization level. AWQ 4-bit results in the lowest accuracy of 0.404, suggesting it is less effective in maintaining accuracy compared to the other methods.

Overall, the trend shows that as the number of bits decreases, the model accuracy tends to drop. 8-bit quantization shows minimal impact on accuracy, making it a preferable choice for applications requiring both efficiency and high accuracy. Among the 4-bit quantization methods, GPTQ retains the highest accuracy.

### 5.2 RAG

As seen from table 4, RAG did not improve the performance of question answering. This can be attributed to the use of the naive RAG retrieval algorithm. We also believe that our database can be enhanced and expanded to include more relevant data from the STEM fields at EPFL. It's possible that our current threshold of 0.4 L2 similarity is too strict, which may limit the Retriever's ability to find enough contexts that satisfy this condition. However, we are confident that the performances of RAG can be boosted by utilizing pre and post-retrieval techniques, such as the ones surveyed by (Gao et al., 2023).

Moreover, despite the accuracy drop, RAG offers a significant advantage in the transparency and traceability of its outputs. Because it incorporates text from external sources, the generated content can be directly linked back to its original data. This traceability enhances the explainability of the model's responses, allowing users to verify and understand the origins of the information provided.

## 6 Ethical considerations

In this project, we have utilized LLMs to generate personalized educational content. While these models provide substantial benefits, they also introduce several ethical challenges:

1. **Bias Mitigation:** A significant concern was the potential for bias in the content generated by our model, particularly due to the preferences and data inherently captured in DPO. To address this, we integrated diverse data sources from various course materials to ensure a balanced representation of educational content.
2. **Environmental and Social Impacts:** The energy-intensive nature of training and maintaining LLMs presents significant environmental challenges, in our case we used around 210 hours at 200 Watts. Additionally, the potential for these models to spread misinformation or exacerbate social disparities calls for a cautious approach to ensure equitable access to the benefits of this technology.
3. **Potential for Misuse:** Recognizing that the model could be used for malicious purposes, such as cheating or spreading misinformation, we have put in place safeguards to detect and mitigate such uses.
4. **Resource Optimization:** We ensured the ethical use of computational resources on the SCITAS clusters by optimizing resource utilization and avoiding unnecessary duplication of efforts. Even with a 8B parameter model, we employed multiple libraries like Accelerate and DeepSpeed to optimize resources while maintaining good model performance.
5. **Adaptation for Signed Language Interaction:** To adapt our model for signed language interaction, we will integrate sign language recognition and generation systems using computer vision techniques to interpret hand gestures and facial expressions. This includes generating signed outputs through animated avatars or video. Utilizing diverse datasets, such as WSL Wikipedia, ensures broad context, particularly for challenging STEM content. Real-time processing will enable smooth interactions, and a chain of thought approach will enhance clarity and accuracy for complex information. Sign retrieval techniques will improve efficiency and naturalness, recognizing that efficiency shapes human language. These strategies will make our model effective and inclusive for signed language users.

## 7 Conclusion

We developed the EPFL TA Meister to support students in science and engineering with personalized assistance. Using pre-training, supervised fine-tuning, and Direct Preference Optimization, we created a chatbot that provides accurate and helpful responses. By integrating advanced techniques like quantization and Retrieval-Augmented Generation, we optimized the model for efficient use without sacrificing too much performance and resources.

This project has shown the potential of open-source materials and university courses data in building powerful educational tools. We've addressed key ethical concerns ensuring the models proposed are publicly available on (4pack team, 2024) and environmentally responsible.

Moving forward, we aim to enhance the chatbot further, including support for signed language interaction. Our journey has been one of growth and learning, and we are excited about the future of EPFL TA Meister. We hope it will make a significant difference in students' learning experiences, offering support and clarity in their studies.

## References

4pack team. 2024. Epfl ta meister implementations. https://huggingface.co/PeterAM4.

AI@Meta. 2024. Llama 3 model card.

L. G. A. Alves et al. 2021. Camel-ai-physics dataset. https://huggingface.co/datasets/lgaalves/camel-ai-physics.

argilla. 2024. Math dpo dataset. https://huggingface.co/datasets/argilla/distilabel-math-preference-dpo.

Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. 2016. Training deep nets with sublinear memory cost.

Tri Dao. 2023. Flashattention-2: Faster attention with better parallelism and work partitioning.

deepset GmbH. 2024. Haystack. Accessed: 2024-06-13.

Tim Dettmers, Mike Lewis, and Sam Shleifer. 2021. Bitsandbytes: 8-bit optimizers and matrix multiplication routines. https://github.com/facebookresearch/bitsandbytes. Accessed: 2024-06-03.

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

Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. 2024. The faiss library.

Dayou Du, Yijia Zhang, Shijie Cao, Jiaqi Guo, Ting Cao, Xiaowen Chu, and Ningyi Xu. 2024. Bitdistiller: Unleashing the potential of sub 4-bit llms via self-distillation. *ACL*.

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

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*.

Roberto Garcia et al. 2024. Fusion approaches in retrieval-augmented generative models. *Journal of Computational Intelligence*.

Sylvain Gugger, Lysandre Debut, Thomas Wolf, Philipp Schmid, Zachary Mueller, Sourab Mangrulkar, Marc Sun, and Benjamin Bossan. 2022. Accelerate: Training and inference at scale made simple, efficient and adaptable. https://github.com/huggingface/accelerate.

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. *arXiv preprint arXiv:2103.03874*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Inc. Hugging Face. 2024. Hugging face. Accessed: 2024-06-13.

Internet Archive. 2024. Stackexchange data dump.

jondurbin. 2024. Python dpo dataset. https://huggingface.co/datasets/jondurbin/py-dpo-v0.1.

Brian Jones et al. 2024. Retrieval-augmented fine-tuning for language models. *Proceedings of ACL*.

Stefan Kalajdzievski et al. 2023. Rank-stabilized adapters for efficient transfer learning. In *Proceedings of the 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

David Kim et al. 2024. Development of self-reflective algorithms in retrieval-augmented systems. *Artificial Intelligence*.

LangChain. 2024. Langchain. Accessed: 2024-06-13.

Programming Language. 2023. Codeagent-python dataset.

Changhun Lee, Jungyu Jin, Taesu Kim, Hyungjun Kim, and Eunhyeok Park. 2024a. Owq: Outlier-aware weight quantization for efficient fine-tuning and inference of large language models. *AAAI*.

Michelle Lee et al. 2024b. Incorporating reason and action into deep language models. *Proceedings of ICLR*.

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.

Ji Lin et al. 2024. Awq: Activation-aware weight quantization for llm compression and acceleration. *MLSys*.

Llama-Index. 2024. Llama-index. Accessed: 2024-06-13.

M-A-P et al. 2022. Codefeedback-filtered-instruction dataset. https://huggingface.co/datasets/m-a-p/CodeFeedback-Filtered-Instruction.

Luis Martinez et al. 2024. Dynamic function calling in ai models during inference. *International Journal of Advanced Computer Science*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models.

Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. 2018. Mixed precision training.

Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. 2022. Mteb: Massive text embedding benchmark. *arXiv preprint arXiv:2210.07316*.

Emily Nguyen et al. 2024. Temporal dynamics in retrieval-augmented generation models. *Proceedings of EMNLP*.

openAI. What are tokens and how to count them.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback.

Richard Y Pang, Wen Yuan, Kyunghyun Cho, He He, Sainbayar Sukhbaatar, and Jason Weston. 2024. Iterative reasoning preference optimization. *arXiv preprint arXiv:2404.19733*.

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.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. 2020. Zero: Memory optimizations toward training trillion parameter models.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Yusuke Shinyama. Pdf miner.

Alexander Smith et al. 2024. Implementing corrective mechanisms in retrieval-augmented generation models. *Journal of Machine Learning Research*.

thewordsmiths. 2024. Stem dpo dataset. `https://huggingface.co/datasets/thewordsmiths/stem_dpo`.

Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, and Nathan Lambert. Trl: Transformer reinforcement learning.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. 2023a. Smoothquant: Accurate and efficient post-training quantization for large language models. *ICML*.

Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighoff. 2023b. C-pack: Packaged resources to advance general chinese embedding.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2023. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*.