From Attention to Education: T5utor Is Really All You Need

MNLPredators
Nicolas Reategui | 311204 | nicolas.reateguirodriguez@epfl.ch
Alessio Desogus | 301705 | alessio.desogus@epfl.ch
Aly Elbindary | 300247 | aly.elbindary@epfl.ch
Seif Hamed | 312081 | seif.hamed@epfl.ch

Abstract

LMs have a wide range of practical applications in the Education sector where they can improve the development of personalized teaching as well as provide tutoring for student communities lacking standard educational resources. We have built an educational assistant using Flan-T5 large, a modestly sized LM able to run on consumer hardware which students can use to assist them in question-answering tasks. To further increase its capabilities we have incorporated RAG to leverage a large knowledge database as well as used 4-bit quantization to build a more memory-efficient model, which is crucial to effectively democratizing LLMs access. Model quantization leads to a 68% reduction in memory footprint without severely impacting performance remaining at 32% accuracy. Furthermore, our flagship model `T5utor` using RAG increased performance by 4.4% with respect to our `MCQA` model. After our training pipelines some of the models reach higher accuracies on common benchmarks w.r.t. the reference model.

1 Introduction

The advent of Large Language Models (LLMs) has revolutionized various sectors, with education being one of the most impacted [1]. In recent years, the focus has shifted towards making these advanced technologies more accessible and practical for everyday use. This is where our work comes in. We have developed an educational assistant `T5utor` based on `LaMini-Flan-T5-783M` [2], a modestly-sized language model that can run efficiently on consumer-grade hardware. This ensures that students from diverse backgrounds can leverage it to assist them with their academic work, particularly in question-answering tasks.

To increase the capabilities of the reference model, we first aligned the model's responses through Direct Preference Optimization (DPO) [3] to high-level EPFL courses content using a custom annotated preference pairs dataset and additional subsets from external datasets. We refined the model using Supervised Fine-Tuning (SFT) utilizing Parameter-Efficient Fine-Tuning (PEFT) [4] methods such as Low-Rank Adaptation (LoRA) [5] on an annotated MCQA-only dataset to specialize the model in question-answering tasks. We employed quantization techniques to make the model more memory-efficient, essential for consumer hardware, and used methods like QLoRA [6] to optimize the model further.

Finally, our approach also integrates Retrieval-Augmented Generation (RAG) [7], enabling the system to access a vast knowledge database based on a domain-specific subset of Wikipedia for more accurate and comprehensive answers.

2 Related Work

Wang et. al. [1] conducted an extensive survey on LLMs for education covering applications ranging from supporting teachers to assisting students providing strong arguments for their use but also showing the need for pedagogically aligned LLMs. Recent work by Adewumi et. al., ProCoT [8], probed Chain-of-Thoughts methods showing LMs potential to stimulate creative and critical thinking of students by allowing them to engage with ChatGPT. Similarly, Santos et. al. [9] explored the potential of four state-of-the-art models, namely ChatGPT, Bing Chat, Bard, and Claude, as agents to reasoning within the context of chemistry by stimulating comprehension and problem-solving skills, as well as allowing tailored learning. Furthermore, Wu et. al. [10], tested GPT4 capabilities using a conversational framework, MathChat, on difficult high school competition problems from the MATH dataset [11]. Prihar et. al. [12] assessed LLMs capabilities on generating explanations to maths problems by comparing them to teacher explanations which could provide useful support for learning. Research has also been performed on

LMs error correcting abilities of academic answers [13] which could prove beneficial by increasing students autonomy when correcting homework. Furthemore, Kim et. al. [14] have recently proposed a summarized retrieval framework, SuRE for open-domain QA questions.

3 Approach

3.1 Generator Model

We chose `LaMini-Flan-T5-783M` as our generator model, belonging to the `LaMini-LM` model series [2]. It builds upon `google/flan-t5-large` [15] a Seq2Seq model by fine-tuning on LaMini-instruction dataset[1] that contains 2.58M samples. We made this choice based on our preliminary training results (see section 4.2.1) and on the benchmarking of different models and architectures we conducted (see Table 7). An overview of our approach is shown in Figure 1.

3.1.1 DPO Fine-Tuning

We explored fine-tuning the entire `LaMini-Flan-T5-248M` model as well as `LaMini-Flan-T5-783M` using LoRA [5] on the DPO datasets (see section 4.1.1). We are using the `DPOTrainer` from HuggingFace which provides rewards margins and accuracy for evaluation. The best performing model was obtained when adding LoRA adapters to `LaMini-Flan-T5-783M`. This DPO generator model, which we will mention now as our `DPO` model, is available on HuggingFace[2].

3.1.2 SFT with PEFT

Several approaches are available when specializing our model to MCQA or extending the context length for RAG augmentation. The first is to continue fine-tuning the LoRA adapter introduced when producing our `DPO` model. The proposed alternative is to add an additional LoRA layer specific to the task at hand. The motivation for the latter is to prevent changing the weights of the adapter which had learnt our desired preferences encoding reasoning abilities specific to our academic dataset (see section 4.1.2). Therefore, we freeze the former LoRA adapter and introduce an additional one which would specialize in MCQA formatting as shown in Figure 2. Furthermore, we decided to use a smaller rank for the latter as we expect MCQA specialization to require a smaller number of weights to encode the desired information. Standard `PEFT` models from HuggingFace allows to use only one active adapter preventing us to implement our architecture. We explored using `PEFTMixed` models setting both the `DPO` and the new adapter as active, though only the latter would be trainable. We note the importance of setting both adapters as active when training as training using only one adapter but running inference with both might yield unexpected results. As

[1] HuggingFace LaMini-instruction dataset

[2] HuggingFace LaMini-Flan-T5-783M-DPO model

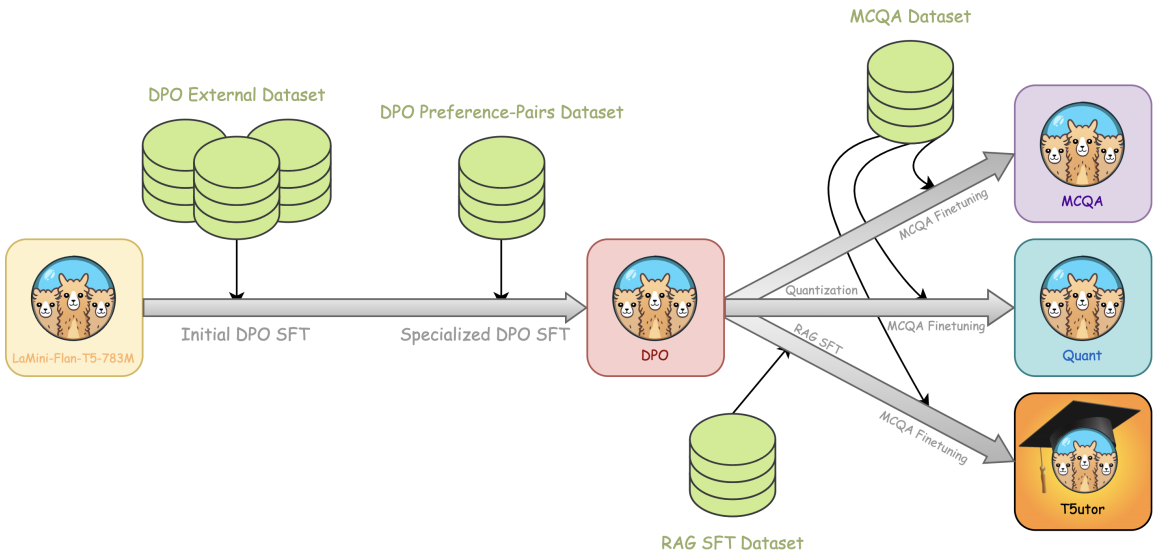

Figure 1: Diagram showing our general approach. Starting from the left, with our chosen generator model `LaMini-Flan-T5-783M`, then going through multiple fine-tuning stages. The name of the models follows the nomenclature in Table 1. The relative sizes of the datasets is also significant, with the `DPO External Dataset` being roughly six times the size of the other smaller datasets.

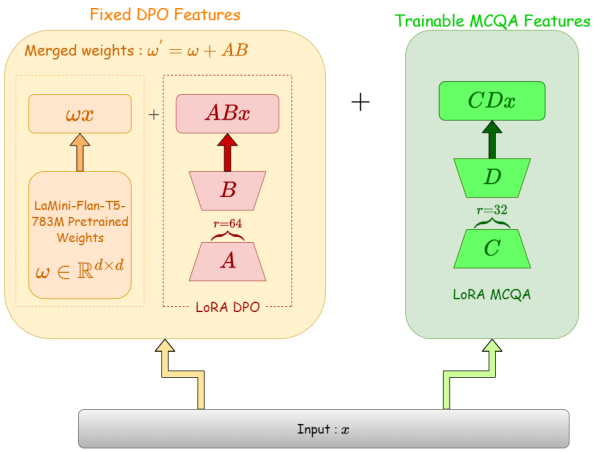

Figure 2: Diagram illustrating our mixed LoRA architecture. Weights on the left block are freezed, only the additional LoRA layer depicted in green is optimized

HuggingFace does not recommend training using `PEFTMixed` models, we decided to merge the first adapter to the Flan-T5 architecture, no additional fine-tuning of the adapter is required, and only optimize the parameters of new LoRA adapter. Additionally, we explored discarding our DPO LoRA layer and fine-tuning a newly introduced adapter directly using the chosen preference pairs formatted with our MCQA template.

3.2 MCQA Specialization

Our main goal is to produce a generator able to provide a single answer to MCQA questions. To do so, we decided to further fine-tune our `DPO` model using SFT methods provided by HuggingFace. During the previous DPO training, our model learned to produced detailed answers to both MCQA and open-ended questions. We decided to leverage the learnt reasoning capabilities of our model and teach it to output the explanation followed by the correct answer by adhering to a specific template, shown in Listing 1.

```
User: {
    [MCQA question]
}
Chatbot: {
    [explanation] \n
    ### Correct answer: [letter]
}
```

Listing 1: User and Chatbot template

We filtered our preference pairs dataset by selecting only the chosen answers containing the correct letter determined by majority voting. We then used HuggingFace's `Trainer` to fine-tune our model.

3.3 Quantization

In order to quantize our model for MCQA generation we decided to first continue fine-tuning our DPO LoRA adapter with a quantized Flan-T5 model to adapt our weights from working using a full precision model to a 4-bit quantized model. We used the same script as for DPO training with `Bitsandbytes` configuration for `NF4` 4-bit quantization. Furthermore, we decided to explore both standard quantization as well double quantization `bnb_4bit_use_double_quant` to save an additional 0.4 bits per parameter. The compute dtype was set to full-precision (`fp32`) as only Volta-architecture GPUs were available to us. We then used the same procedure as described in the previous section for MCQA specialization with our quantized model. For this last step we explored merging our DPO layer with T5 and applying quantization then introducing an additional adapter to specialize on MCQA answering.

3.4 RAG Augmentation

We used an Advanced RAG [7] approach at the inference stage leveraging the specialized framework LangChain[3] and the retrieval source described in the section 4.1.4. Our pipeline is showed in the Figure 3 and includes a multi-step process to optimize the performance.

3.4.1 Embedding Vector Generation

First, we split the documents from our retrieval source (see section 4.1.4) into smaller chunks using a recursive approach to prepare a collection of semantically relevant snippets. Then, using the small but efficient embedding model `gte-small` [16] we compute the embedding for all the chunks and store them in a vector database. The query is embedded by the same model, and a similarity search using cosine similarity retrieves the n closest documents from the vector database. To improve the accuracy, we initially retrieved more documents than needed and then re-rank the results using a more powerful retrieval model before selecting the *Top-k* documents. For re-ranking, we employ `Colbertv2` [17], a cross-encoder that computes fine-grained interactions between the query and chunk tokens. Finally, we built the final prompt that will be given to the reader model using the initial query and the *Top-k* selected document.

[3] `https://python.langchain.com`

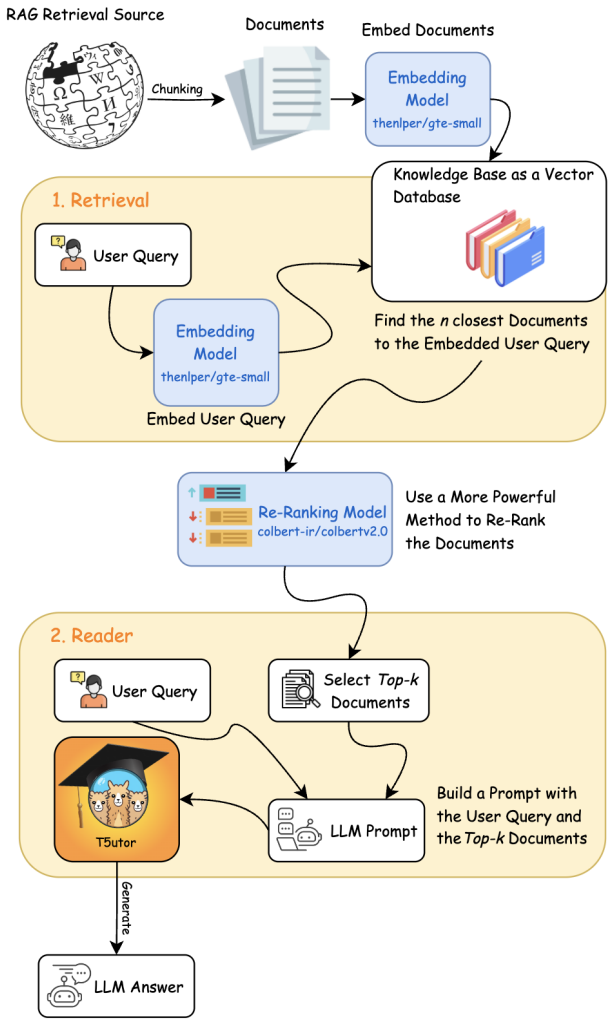

Figure 3: RAG Pipeline

3.4.2 Context Extension and Specialization

However, to address the limitation of our model's 512-tokens context window we have extended this window using a SFT approach with a specialized dataset we created (see section 4.1.5). This approach aims at extending the context window to at least 3K tokens which improves the model's ability to understand and process information relevant to the questions.

4 Experiments

4.1 Datasets

In this first sub-section, we will be exploring the different datasets that have been utilized at different stages of our approach. There are mainly two types of datasets depending on the training approach either DPO or SFT, each having possible subsets. Finally, there is the RAG retrieval source used as knowledge vector.

4.1.1 DPO External Dataset

We have carried out 2 consecutive DPO fine-tuning stages with different datasets. The first one, `dpo_general.jsonl`, is composed of `NeuralNovel/Neural-DPO`[4] (1.07K rows), `M4-ai/prm_dpo_pairs`[5] (93.9K) as well as the `explainlikeimfive_train` subset of `stanfordnlp/SHP`[6] (19K). The first of these contains high-school-level mathematical problems, the second relates to AI and the last one is a subreddit thread where explanations of different topics are given in lay terms.

4.1.2 DPO Preference-Pairs Dataset

The second dataset of 26K rows, named `dpo_preference_pairs.jsonl`, is a filter version of the preference pairs collected by the class. The selection of the `chosen` preference was based on the better `overall` field.

4.1.3 MCQA Dataset

To generate our MCQA dataset we used the provided preference pairs, for each multiple-choice question (13K out of the original 26K rows) we prompted `GPT3.5` to extract the correct answer from the explanation. Although, in certain instances, the pipeline yielded wrong results it was able to extract the correct letter on a large majority of the explanations. We then grouped the answers by question and applied majority voting to approximate the correct answer. There are 792 unique multiple-choice questions and 16 answers for each on average which allows us to properly estimate the correct answer through majority voting. Our MCQA dataset is named `mcqa_dataset.jsonl`.

4.1.4 RAG Retrieval Source

Our retrieval source is based on the English Wikipedia Dump[7]. More precisely, we have filtered the initial Wikipedia dataset, which contains more than 6M documents, using 8 keywords such as *computer science* and *machine learning* (see Table 6) related to the course topics of the MCQA dataset, resulting in a reduced set of 133K documents. Our retrieval source is named `rag_retrieval_source.json`.

[4] HuggingFace NeuralNovel/Neural-DPO dataset
[5] HuggingFace M4-ai/prm_dpo_pairs dataset
[6] HuggingFace stanfordnlp/SHP dataset
[7] HuggingFace Wikipedia Dump dataset

Model Name	Model N°	DPO	MCQA	QT	RAG
MZBUAI/LaMini-Flan-T5-783M	I (Generator)				
NicolasRR/LaMini-Flan-T5-783M-DPO	II (DPO)	√			
DPO-MCQA-FalseMixin	III (MCQA)	√	√		
DPO-MCQA-TrueMixin	IV	√	√		
DPO-DoubleQuant	V	√		√	
DPO-DoubleQuant-MCQA-TrueMixin	VI (Quant)	√	√	√	
DPO-DoubleQuant-MCQA-FalseMixin	VII	√	√	√	
DPO-RAGcontext-FalseMixin	VIII	√			√
DPO-RAGcontext-TrueMixin	IX	√			√
DPO-RAGcontext-TrueMixin-MCQA-FalseMixin	X (T5utor)	√	√		√

Table 1: Summary of the variations of the initial Generator model and their specific adaptations for different tasks: DPO, MCQA, Quantization (QT) and RAG. A check-mark in the respective columns indicates the tasks for which each model variant has been adapted.

4.1.5 RAG SFT Dataset

This SFT specialized dataset leverages the 26K questions derived from the dpo_preference_pairs.jsonl dataset, our RAG pipeline, and GPT3.5 as the reader model. This process involves generating a prompt for each question that includes the n documents from the retrieval source. These documents are not re-ranked and the *Top-5* selection is done randomly to allow training on documents relevant to the question but potentially different from the ones used at inference time. Subsequently, an answer based on these documents is generated. Our context extension dataset is named rag_sft_dataset.json.

4.2 Results & Analysis

Both training and evaluation of all models was performed on V100 GPUs, used 190 jobs consuming 860 GPU hours collectively and producing 35.6 kg of eCO_2. During training we used full precision fp32 for non-quantized models as initial exploration with mixed-precision fp16 showed model divergence. Furthermore, we set the learning rate at $1e^{-4}$ as T5 models require it to be higher than for other LMs. We used Weights and Biases for logging. A detailed summary of all runs can be found on our wandb[8] report

4.2.1 DPO Fine-Tuning

When fine-tuning using DPOTrainer, we performed evaluation every 250 steps and used both the reward margins and accuracy as metrics on a holdout dataset consisting of 5% of our data. The former evaluates the mean difference between the chosen and corresponding rejected rewards and the latter indicates whether the margins are positive (1) or negative (0). We explored fine-tuning the entire LaMini-Flan-T5-248M model as well as LaMini-Flan-T5-783M using LoRA [5] on the dpo_general.jsonl dataset for 3 epochs as well as on the dpo_preference_pairs.jsonl dataset for 4 epochs, consuming for each run around 24 and 7 GPU hours respectively. We used the an effective batch size of 32 combining both batch size (2) and gradient accumulation (16) parameters for better stability. Furthermore, we used the default $\beta = 0.1$ for DPO as well as $\alpha = 8$ for LoRA. We observed that fine-tuning only the LoRA adapters allows us to reach high accuracy. Exploration experiments used prm_dpo_pairs as it allow us to perform fast iterations, results are shown in Figure 5. Fine-tuning the entire Flan-T5 large model would not only be slower due to gradient checkpointing but would lead to training instability as well as very slow convergence. Therefore, we decided to use the larger model and leverage LoRA for efficient and stable training. We also explored varying the rank of the added LoRA layers to increase the expressivity of our network but found no significant change between 32, 64 and 256. We explored several other hyperparameters amongst which the loss function. We observed no major differences between the sigmoid with and without label smoothing (0.1), SPPO [18] and BCE [19]

[8] Weights and Biases report

loss as shown in Figure 6. Nevertheless, using `IPO loss` [20] function seems to severely impact performance by reducing our accuracy from more than 90% down to 62%. Our final model was obtained by fine-tuning `LaMini-Flan-T5-783M` using LoRA with a rank of $r = 64$ and $\alpha = 8$, named hereby `DPO` model.

```
reponse = model.generate(input)
if response.startswith("True")
    answer = "A"
elif response.startswith("False")
    answer = "B"
else:
    C = count_occurences(answer,
    {"A","B","C","D"})
    if sum(C)>0:
        answer = argmax(C)
    else:
        answer = impute()
```

Listing 2: MCQA extraction

4.2.2 SFT Fine-Tuning

We started all our fine-tuning procedures for M3 from our `DPO` model. Evaluation was performed every 125 steps using the default `cross-entropy` loss function. MCQA fine-tuning was performed using a maximal input context length of 2048 tokens and output length of 1024. We increased the former from 512 to 2048 to train on a larger datasets as all prompts not fitting within that context were discarded. When introducing an additional adapter we observed an increase in the evaluation loss after 0.81 epochs stabilizing at 0.6. On the other hand, fine-tuning our DPO LoRA layer resulted in values as low as -0.29. Therefore, we decided to deliver for our MCQA specialization the `DPO-MCQA-FalseMixin` referred as only `MCQA`.

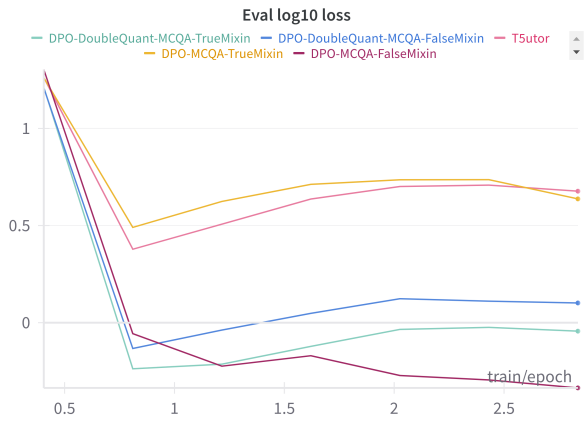

Figure 4: Cross-entropy evaluation loss of our model for MCQA specialization. The nomenclature used for the traces names is described in 3

Model N°	DPO	MCQA		QT
	Acc.	*"A"*	*Rand.*	*Size*
I (`Generator`)		33.3	32.6	
II (`DPO`)	**62**	32.1	31.7	
III (`MCQA`)	58	31.7	31.7	
IV	58	31.7	31.1	
V	61	32.6	31.3	-67
VI (`Quant`)	57	33.2	32.6	**-68**
VII	59	33.2	32.4	-67
VIII	48	32.8	31.6	
IX	18	33.2	32.6	
X (`T5utor`)	18	**33.5**	**32.8**	

Table 2: Evaluation of the different models across the DPO and MCQA benchmarks. The size reduction (QT) for the quantized models is displayed.

When prompting our model with MCQA questions we observed that it had not learned the desired format and had to developed heuristics to post-process the response and extract the answer described in Listing 2. Furthermore, we tested our models on 4 common benchmarks with some of the specializations reaching higher accuracy compared to the `Generator` model, see Table 4.

When extending the model's context length for RAG, we observed a stable decrease in the evaluation loss as shown in Figure 4. Fine-tuning directly the DPO adapter yield better performance than our alternative Mixed model adapter approach. Once the training for the RAG model (`T5utor`) was completed we evaluated it's performance using multiple hyperparameters. The first one is the number of retrieved documents obtained after maximizing the cosine similarity with the query vector, we used 10, 50 and 100 documents, see Table 3. The second parameter explored was the number of documents included in the context after re-ranking using `Colbertv2`. Despite extending the context length, our model was initially trained on 512 tokens potentially leading to strong locality effects, i.e. better performance on tokens on the last 512 tokens. After exploring adding the question at the beginning of the context we decided to include both at the beginning and end as shown in the Listing 3. When including a single document in the context we can benefit from first retrieving a large number of documents as the similarity measure

Model	MCQA with RAG											
Retrieved Doc.	**10**				**50**				**100**			
Top-k	*1*	*2*	*3*	*4*	*1*	*2*	*3*	*4*	*1*	*2*	*3*	*4*
`Generator`	30.6	32.1	28.3	27.1	31.8	30.4	28.4	30.4	32.3	31.3	27.5	28.0
`DPO`	30.2	30.6	27.7	26.1	31.8	29.9	29.7	28.5	31.9	29.9	27.5	27.4
`T5utor`	**30.9**	30.7	29.0	25.8	**31.9**	30.4	28.3	28.4	**33.1**	31.2	28.7	26.2

Table 3: Performance comparison of different models augmented with RAG using varying the number of retrieved document before re-ranking (Retrieved Doc.) as well as the number of *Top-k* documents included in the context. Values were inputted randomly (same as *Rand.* in Table 2) when not following the correct output format.

Model	SciQ [21]	ARC-easy [22]	ARC-challenge [22]	PIQA [23]
`Generator`	90.80	59.01	29.69	71.11
`DPO`	90.90	58.84	**30.20**	**71.16**
`MCQA`	**91.10**	**59.13**	29.86	70.84
`Quant`	**91.10**	59.09	29.01	71.06
`T5utor`	90.80	59.01	29.69	71.11

Table 4: Performance of the selected models on various benchmarks using `lm-harness` [24].

provided by the re-ranker model is more performant than the cosine similarity. This effect is less clear when including more than 1 document in the context. We suspect underlying context-size prevents the model from using information for early documents despite trying to extend the context size as mentioned before. Furthermore, we decided to test whether RAG augmentation would also lead to better performance when directly applied to the `Generator` and `DPO` models which did not undergo the context-extension procedure.

```
prompt = f'''Answer the following question: "{question}".\n
    Use the following context if you deem necessary: "{context}".\n
    Answer the following question: "{question}".
    Specify the ID of the correct answer (A, B, C or D).\n
    Think step by step and explain your reasoning'''
```

Listing 3: Prompt structure of the RAG pipeline combining the user query both at the beginning and at the end with the *Top-k* documents in the middle

Additionally, our `T5utor` model using RAG reaches an accuracy as high as 33.1 % compared to 31.7% of the `MCQA` model and 32.8 % of `T5utor` both without RAG. Despite these small improvements we believe using a larger model could lead to a higher increase in performance when augmenting our prompt with RAG.

5 Ethics

We identified potential ethical problems arising from our work: the first is a potential misuse of our model, the second is the factual correctness and reliability of the generated answers and the last is related to language adaptation.

5.1 Potential Misuse of the Model

Our model could be used maliciously to answer exams MCQs, cheating, where the model may be given a few different answers from different students and asked to identify which of them is the correct answer.

5.2 Factual Correctness and Reliability

As mentioned before, our model's API can be modified such that the user can interact with it. However, from this, an automatic problem arises which is the factual correctness and reliability of the model: if the model outputs a false answer to the question with a convincing enough reasoning behind it, the student can be misled, and it can even lead into a misunderstanding in the broad scientific concept behind the answer, which would make the Chatbot counterproductive in that case.

5.3 Language Adaptation

Adapting the model to handle both multiple languages and sign-languages presents its own set of

ethical challenges. One major concern for handling multiple languages is ensuring the quality and fairness of the model across different languages. If the training data for some languages is of lower quality or less comprehensive, this could result in biased or inaccurate outputs, disadvantaging speakers of those languages. Additionally, the model might inadvertently propagate cultural biases present in the training data, leading to potentially offensive or harmful outputs in certain languages [25]. Our model can be extended to interpret sign-languages using the framework proposed by Gong et. al. [26].

6 Conclusion

We explored building an educational tutor to help students in MCQA answering tasks. Our experiments showed standard DPO loss can be leveraged to teach the Flan-T5 large model our preferred reasoning approaches on academic questions. Furthermore, LoRA adapters provide an efficient solution to encode the desired information and lead to better stability as well as lower training times. After training, our `DPO` model reaches 62% accuracy on the class preference pairs. Furthermore, we specialized the generator to solve MCQA questions by fine-tuning our LoRA layers or adding new adapters. Additionally, we implemented RAG to leverage a knowledge database and retrieve the most relevant documents to answer the question at hand, increasing the performance by 4.4% w.r.t our `MCQA` model. Nevertheless, we found that our model was not able use the whole context provided. Therefore, using a model with a larger context size is required to fully utilize RAG.

Author Contributions

The authors ordering has been established according to the contributions to the team's work.

Author	Contribution
	M2 milestone
NR	Building the DPO datasets
NR	Building the MCQA dataset
NR	Writing training scripts for DPO
NR	Team's cluster setup
NR	Hyper-parameter exploration
NR	Writing report
AD	Building RAG retrieval source
AD	Writing RAG pipeline
AD	RAG model selection
AD	Writing report
SH	Building MCQA dataset
AE	Model training on izar
	M3 milestone
NR	Writing quantization script
NR	Writing SFT script for context ext.
NR	Writing SFT script for MCQA
NR	Model training on izar
NR	Writing report
NR	Model evaluation
AD	Building RAG SFT dataset
AD	Model training on izar
AD	Writing report
AD	Model evaluation
AE	Model training on izar
AE	Writing report & figures
AE	Model evaluation
SH	Writing report & ethics assessment

Table 5: Detailed breakdown of individual contributions for the M2 and M3 milestones.

References

[1] Shen Wang, Tianlong Xu, Hang Li, Chaoli Zhang, Joleen Liang, Jiliang Tang, Philip S. Yu, and Qingsong Wen. Large language models for education: A survey and outlook, 2024.

[2] Minghao Wu, Abdul Waheed, Chiyu Zhang, Muhammad Abdul-Mageed, and Alham Fikri Aji. Lamini-lm: A diverse herd of distilled models from large-scale instructions, 2024.

[3] Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model, 2023.

[4] Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, and Benjamin Bossan. Peft: State-of-the-art parameter-efficient fine-tuning methods. https://github.com/huggingface/peft, 2022.

[5] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021.

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

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

[8] Tosin Adewumi, Lama Alkhaled, Claudia Buck, Sergio Hernandez, Saga Brilioth, Mkpe Kekung, Yelvin Ragimov, and Elisa Barney. Procot: Stimulating critical thinking and writing of students through engagement with large language models (llms), 2024.

[9] Renato P. dos Santos. Enhancing chemistry learning with chatgpt, bing chat, bard, and claude as agents-to-think-with: A comparative case study, 2023.

[10] Yiran Wu, Feiran Jia, Shaokun Zhang, Hangyu Li, Erkang Zhu, Yue Wang, Yin Tat Lee, Richard Peng, Qingyun Wu, and Chi Wang. An empirical study on challenging math problem solving with gpt-4, 2023.

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

[12] Ethan Prihar, Morgan Lee, Mia Hopman, Adam Tauman Kalai, Sofia Vempala, Allison Wang, Gabriel Wickline, Aly Murray, and Neil Heffernan. Comparing different approaches to generating mathematics explanations using large language models. In Ning Wang, Genaro Rebolledo-Mendez, Vania Dimitrova, Noboru Matsuda, and Olga C. Santos, editors, *Artificial Intelligence in Education. Posters and Late Breaking Results, Workshops and Tutorials, Industry and Innovation Tracks, Practitioners, Doctoral Consortium and Blue Sky*, pages 290–295, Cham, 2023. Springer Nature Switzerland.

[13] Xiaowu Zhang, Xiaotian Zhang, Cheng Yang, Hang Yan, and Xipeng Qiu. Does correction remain a problem for large language models?, 2023.

[14] Jaehyung Kim, Jaehyun Nam, Sangwoo Mo, Jongjin Park, Sang-Woo Lee, Minjoon Seo, Jung-Woo Ha, and Jinwoo Shin. Sure: Summarizing retrievals using answer candidates for open-domain qa of llms, 2024.

[15] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. Scaling instruction-finetuned language models, 2022.

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

[17] Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. Colbertv2: Effective and efficient retrieval via lightweight late interaction, 2022.

[18] Yue Wu, Zhiqing Sun, Huizhuo Yuan, Kaixuan Ji, Yiming Yang, and Quanquan Gu. Self-play probabilistic preference optimization for language model alignment, 2024.

[19] Seungjae Jung, Gunsoo Han, Daniel Wontae Nam, and Kyoung-Woon On. Binary classifier optimization for large language model alignment, 2024.

[20] Mohammad Gheshlaghi Azar, Mark Rowland, Bilal Piot, Daniel Guo, Daniele Calandriello, Michal Valko, and Rémi Munos. A general theoretical paradigm to understand learning from human preferences, 2023.

[21] Matt Gardner Johannes Welbl, Nelson F. Liu. Crowdsourcing multiple choice science questions. 2017.

[22] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv:1803.05457v1*, 2018.

[23] Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*, 2020.

[24] Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation, 12 2023.

[25] Isabel O. Gallegos, Ryan A. Rossi, Joe Barrow, Md Mehrab Tanjim, Sungchul Kim, Franck Dernoncourt, Tong Yu, Ruiyi Zhang, and Nesreen K. Ahmed. Bias and fairness in large language models: A survey, 2024.

[26] Jia Gong, Lin Geng Foo, Yixuan He, Hossein Rahmani, and Jun Liu. Llms are good sign language translators. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18362–18372, June 2024.

A Appendix

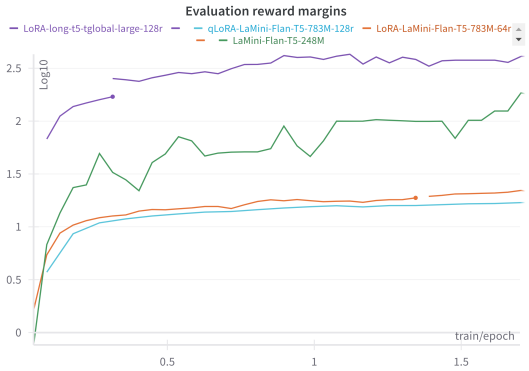

Figure 5: Evaluation reward margins in a logarithmic scale for different fine-tuning approaches on the `M4-ai/prm_dpo_pairs` dataset (subset of the `dpo_preference_pairs.jsonl` dataset). We compare different model architectures and configurations.

N°	Keywords
1	computer science
2	computer software
3	computer systems
4	machine learning
5	artificial intelligence
6	mathematics
7	physics
8	cybersecurity

Table 6: List of the 8 keywords used in the creation of the `RAG Retrieval Source`.

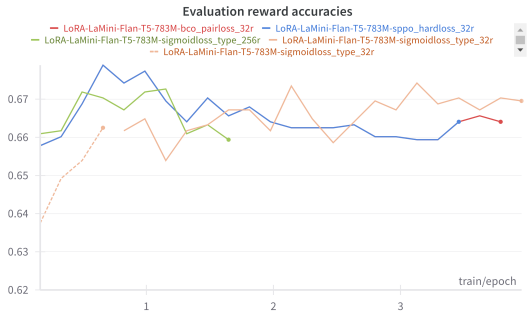

Figure 6: Evaluation reward accuracies for different fine-tuning approaches of the `LaMini-Flan-T5-783M` model, already fine-tuned on the `dpo_general.jsonl`, using LoRA on our preference pairs dataset `dpo_preference_pairs.jsonl`. We compare different loss functions, including `bco_pairloss`, `sigmoidloss`, and `sppo_hardloss`, as well as varying the rank of the LoRA layers between 32 and 256.

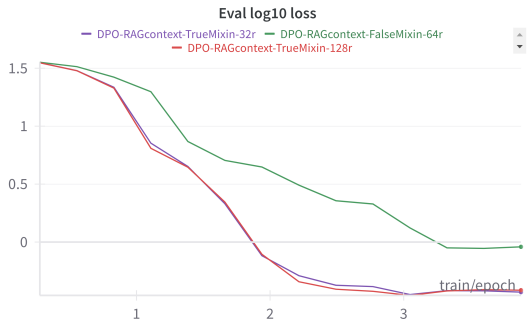

Figure 7: Evaluation cross-entropy loss for rag-context extension in logarithmic scale.

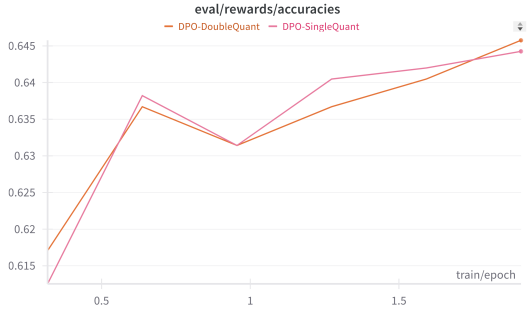

Figure 8: Evaluation reward margins in a logarithmic scale for different fine-tuning approaches on the `M4-ai/prm_dpo_pairs` dataset (subset of the `dpo_preference_pairs.jsonl` dataset). We compare both standard 4bit and double quantization.

Model	Params	SciQ [21]	ARC-easy [22]	ARC-challenge [22]	PIQA [23]
dolphin-2_6-phi	2.7B	95.10	79.84	50.09	79.43
LaMini-Flan-T5	783M	**90.80**	**59.01**	**29.69**	**71.11**
LaMini-Flan-T5	248M	90.20	52.06	24.40	66.59
LaMini-GPT	774M	88.50	58.54	25.94	69.31
Qwen1.5-Wukong	464M	88.10	58.80	25.34	69.37
t5-large	738M	86.10	32.70	20.99	58.92
gpt2-large	812M	80.40	53.16	21.67	70.35
Alpaca-tuned-gpt2	774M	79.80	52.86	24.32	70.89

Table 7: Performance of various well-known models under 1B parameters against the dolphin-2_6-phi model with 2.7B parameters on different benchmarks using lm-harness [24].