Crafting an AI Tutor: From Preference Data to Generator Model Training

Andrea Miele | 302925 | andrea.miele@epfl.ch
Luca Mouchel | 324748 | luca.mouchel@epfl.ch
Elia Mounier-Poulat | 314771 | elia.mounier-poulat@epfl.ch
Frederik Gerard de Vries | 369939 | rik.devries@epfl.ch
REAL

1 Introduction

In this project, our objective is to create an AI tutor chatbot to support learning in EPFL courses.

Based on training time and memory capacity, we will see which model is the most suitable. The candidates are Meta's Llama 2 (7B) model and Google's Gemma (2B) model. In case these are too costly to train, we will consider a smaller model such as Pythia's 1B model or TinyLlama (1.1B). Our direct preference optimization (DPO) training strategy involves two steps. First, we will perform supervised fine-tuning (SFT) on the pre-trained model to learn the question-answering task. Then, DPO will optimize the model to favor preferred samples and reject dispreferred ones. To further improve our model, we will implement both Retrieval Augmented Generation (RAG) and Quantization. Several classical NLP evaluation metrics, GPT feedback, and human evaluation will help us iteratively refine and select the optimal model.

2 Model

2.1 Generator Model

We will use the `trl` (von Werra et al., 2020) library which implements useful classes to run SFT and DPO. Moreover, if we use Llama-2 (and perhaps Gemma-2B too), we will leverage parameter efficient fine-tuning (PEFT) (Xu et al., 2023), using Low-Rank Adaptation (LoRA) (with the `peft` package) (Hu et al., 2021). LoRA is a PEFT method that decomposes a large matrix into two smaller low-rank matrices in the attention layers. This drastically reduces the number of parameters that need to be fine-tuned.

2.2 RAG Specialization

To improve the generator model with RAG, we will add some context to the prompts and retrain the whole pipeline (SFT + DPO with the added context). This should make the model be more factual. We will use `transformers` implementation of RAG to retrieve extra context from an external database.

2.3 Quantization Specialization

We will implement quantization using the `transformers` library with the `bitsandbytes` module all the while using the LoRA layers on top of the quantized base model (if we use LoRA) (Xu et al., 2023; Dettmers et al., 2023). We will use the `prepare_model_for_kbit_training` function to preprocess the quantized model for training and then we can train SFT and DPO using the quantized models.

3 Data

The aggregated data collected in milestone 1 can be denoted as $\mathcal{D} = \{x^i, y_w^i, y_l^i\}_{i=0}^N$ where y_w stands for the preferred response (win) and y_l for the dispreferred (lose). x^i is the prompt, containing the question. For SFT, we will only use $\mathcal{D}' = \{x^i, y_w^i\}_{i=0}^N$ since we do not need ranked pairs of responses. This way, the first step consists of fine-tuning the models on the preferred responses so it already knows about generating "good" answers and explanations. We might use an external dataset such as *ScienceQA* (Lu et al., 2022) or *MMLU* (Hendrycks et al., 2021), which consists of STEM-related questions among other types of questions, for supervised fine-tuning (SFT). We will use these to augment the dataset we will be given and do SFT on the augmented dataset. We will only use a subset of the additional datasets, given questions from *MMLU* for example contain only the answer to the MCQ without explanation, so we would need to generate these explanations with the GPT wrapper we were provided in milestone 1. After SFT, we will train our model with DPO on $\mathcal{D}$, the whole dataset collected in milestone 1, as described above.

3.1 Generator Model

To run SFT, samples must have the format described in Appendix .1 Where "prompt" is the prompt we give the model. The completion will be extracted from the preference dataset we will be given, as the preferred response for the given question. The prompt we will use will have the format according to Appendix .2. For DPO, we will use the whole preference data we are given to train, using the SFT model as the reference model. The format for the data should be as described in Appendix .1. Where we will extract the chosen and rejected samples by taking the `Overall` ranking as the chosen response and the other one as the rejected one. The prompt will have the same format as the one we use for SFT.

3.2 RAG Specialization

We will use the `wiki-dpr` database and use the HuggingFace implementation of RAG using the `RagRetriever` class. For each question, we will extract knowledge on the subject by giving the retriever the question and it will find relevant knowledge from the database. This way, we can embed the knowledge extracted into the prompt we give the model.

3.3 Quantization Specialization

We will use the same data for this specialization.

4 Evaluation

4.1 Generator Model

We plan to evaluate our model during two separate phases of our pipeline: i) after the SFT level, and ii) after the DPO training, using the same evaluation scheme. We will compare the improvement without any specialization added and when RAG and/or Quantization is incorporated into the workflow.

For quantitative evaluation, we will use standard metrics such as ROUGE, BLEU, and BERTscore (Lin, 2004; Papineni et al., 2002; Zhang et al., 2020). Additionally, we will incorporate DiscoScore (Zhao et al., 2023), a novel metric designed to assess discourse coherence and factual consistency. This metric will be particularly relevant to our project, as our goal is for our AI tutor to learn and generate responses that are logical and correct.

For qualitative evaluation, we plan on using a GPT wrapper to assess the quality of the final generated answers. This method is not only fast but also consistent. We will then compare the machine-generated evaluations with human assessments on a subset of the generated sequences and calculate an agreement score. This score will help us determine the degree to which the machine-generated evaluations align with human judgments. Additionally, it will enable us to identify areas for improvement in the model's performance.

4.2 RAG Specialization

To evaluate the model after adding RAG, we plan to use the following methods. First, it could be interesting to look at RAGAS (Es et al., 2023) or RGB, frameworks for reference-free evaluation of RAG pipelines. It evaluates RAG based on several metrics such as faithfulness, answer correctness, and more.

Finally, two other relevant metrics are the Hit Rate and the Mean Reciprocal Rank (MRR) which are metrics to evaluate the efficacy of our retrieval system.

4.3 Quantization Specialization

To evaluate the models after quantization, we will benchmark the performance on zero-shot benchmarks (by reporting the average accuracy) before and after quantization. Relevant benchmarks for this task are MMLU (Bugliarello et al., 2022) or PIQA(Bisk et al., 2019). We will also evaluate with the metrics described in 4.1. The latency and memory saved when quantizing or not our model will also be monitored.

5 Ethics

As our model aims to provide good answers to university-grade questions, it could be widely used to cheat on assignments and exams. This might also demotivate students to achieve their goals as this tool can do what is expected of them more efficiently. However, as our model will only output a single-letter answer to a multiple-choice question, we believe that the AI chatbot will not negatively affect students' learning but rather guide them towards the correct reasoning thanks to the given final solution. Although biases do exist in STEM and other fields, given that we should only output the final answer to MCQ, we do not believe any specific type of bias plays a sufficient role to be an ethical issue in the annotation process and dataset creation.

References

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2019. Piqa: Reasoning about physical commonsense in natural language.

Emanuele Bugliarello, Kai-Wei Cheng, Desmond Elliott, Spandana Gella, Aishwarya Kamath, Liunian Harold Li, Fangyu Liu, Jonas Pfeiffer, Edoardo Maria Ponti, Krishna Srinivasan, Ivan Vulić, Yinfei Yang, and Da Yin, editors. 2022. *Proceedings of the Workshop on Multilingual Multimodal Learning*. Association for Computational Linguistics, Dublin, Ireland and Online.

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

Shahul Es, Jithin James, Luis Espinosa-Anke, and Steven Schockaert. 2023. Ragas: Automated evaluation of retrieval augmented generation.

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

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.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

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.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

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

Lingling Xu, Haoran Xie, Si-Zhao Joe Qin, Xiaohui Tao, and Fu Lee Wang. 2023. Parameter-efficient fine-tuning methods for pretrained language models: A critical review and assessment.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert.

Wei Zhao, Michael Strube, and Steffen Eger. 2023. Discoscore: Evaluating text generation with bert and discourse coherence.

Appendix

.1 Data Format

For SFT, the format of the data is the following:

```
{ "prompt": <>, "completion":<>}
```

For DPO, it is the following:

```
{"prompt": <>, "chosen":<>, "rejected":
                <>}
```

.2 Prompt Formatting

The prompt we will use will have the following format:

```
<bos_token> [INST]
### Task: You will be given a question
related to stem ...
### Question: What is ... [/INST]
### Answer: *** <eos_token>
```