SuriChat: Aligning OpenELM using DPO for STEM Chatbot Responses

Joanna Wolski | 299971 | joanna.wolski@epfl.ch
Diogo Valdivieso Damasio Da Costa | 311673 | diogo.valdiviesodamasiodacosta@epfl.ch
Tomas Valdivieso Damasio da Costa | 300086 | tomas.valdiviesodamasiodacosta@epfl.ch
NLPSuricates

Abstract

This project addresses the development of SuriChat, a large language model (LLM) specialized in providing explanations on a wide array of engineering topics covered by EPFL courses and capable of answering multiple-choice questions (MCQ). The challenge lies in the complexity and diversity of these subjects, as well as the need for accurate and coherent explanations. Current methods often fail to deliver precise and contextually relevant information, particularly in specialized domains.

To overcome these challenges, we employed Direct Preference Optimization (DPO) to fine-tune the OpenELM-450M-Instruct model released by Apple in April 2024. We also added the implementation of a new linear layer which maps the model's output dimension to four classes corresponding to the answer of a MCQ. This classifier head enhances the model's ability to categorize.

Our DPO model was able to select the correct prompt from our preference data set with an accuracy of 0.63 % after being trained on 60000 data points.

Our MCQ finetuned model was able to have significant improvements in model performance, achieving an accuracy of 0.36% in selecting the correct multiple choice compared to the baseline accuracy of 0.25%.

1 Introduction

In recent years, the development of new AI tools has generated significant interest in chatbots for educational purposes, offering students immediate assistance and personalized learning experiences. The objective of this project is to develop a large language model (LLM) specialized in providing explanations on a wide array of engineering topics covered by EPFL courses and capable of answering multiple choice questions (MCQ) often found in exams. This task presents several challenges due to the complexity and diversity of engineering subjects, as well as the necessity for the model to produce accurate and coherent explanations. Current methods, while advanced, often fall short in delivering precise and contextually relevant information, particularly in specialized domains like the one often found in EPFL courses.

This project aims to address these challenges by fine-tuning a STEM question-answering model using Direct Preference Optimization (DPO). Unlike traditional methods such as Reinforcement Learning from Human Feedback (RLHF), which can be costly, complex, and unstable, DPO offers a simpler approach to optimizing language models based on human preferences reducing the cost of training a reward model. By aligning OpenELM-450M-Instruct model released by Apple in April 2024 with DPO, we created then the SuriChat chatbot that can serve as a valuable educational tool for students.

We further specialized our model to answer MCQ. To accomplish this we added a classifier head to our DPO model. This layer mapped the output dimension of 1538 to four classes, corresponding to the output letters of multiple-choice questions (MCQ).

2 Related Work

The task we want to address has already been tackled by models like SciBERT (Beltagy et al., 2019). SciBERT is a BERT-based model pretrained on a large corpus of scientific texts. It has been fine-tuned for various tasks, including question answering and multiple-choice question answering (MCQA) in the STEM domain. However, fine-tuning on scientific literature can introduce potential biases due to the nature and composition of the texts used, which might not represent the diversity of the entire scientific community (Gallegos et al., 2024). Fine-tuning language models to produce desired outputs while avoiding undesired ones is then crucial for improving their performance. A common method for achieving this is Reinforcement Learning from Human Feedback (RLHF) (Christiano et al., 2023).

RLHF is typically used to ensure that language models generate appropriate, well-formatted responses and avoid producing harmful content such as hate speech. In RLHF, a reward model is trained to mimic human ratings of model outputs. The language model is then trained using this reward model to generate high-quality outputs without deviating significantly from its original behavior to prevent mode collapse. Mode collapse occurs when a model learns to produce a limited variety of outputs that exploit the reward model without really improving. For these reasons, RLHF is widely used, but it can often be costly, complex, and unstable during training.

Direct Preference Optimization (DPO) (Rafailov et al., 2023) offers an alternative to RLHF by eliminating the need for reinforcement learning. Instead, DPO directly optimizes the language model to increase the probability of human-preferred outputs and decrease the probability of less preferred ones. The authors of the DPO approach have shown that their loss function is mathematically equivalent to that used in RLHF. This equivalence suggests that DPO can achieve similar improvements in model performance without the costs associated with RLHF and training instabilities.

Our work will fine-tune a STEM question-answering model using DPO to mitigate biases that could arise from the datasets the model is trained on. By providing pairs of preferred responses annotated by humans, we aim to enhance the model's performance while ensuring fairness and reducing biases.

3 Approach

Our approach to tackle his project was structured into three primary categories:

1. Alignment to human preferences using DPO
2. Finetuning the model to answer multiple-choice questions (MCQ)
3. Quantization of the model to improve inference time and model size

3.1 Datasets

To accomplish the training and testing of our model alligned using DPO, we used a preference dataset of response pairs, where one response is rated higher quality. Preference data was collected from four datasets for preference training. Finally to finetune our model to answer MCQ we collected a fifth dataset.

Dataset generated by students Students used ChatGPT to generate two answers to questions from EPFL courses, ranking them by correctness, relevance, and clarity. From 1,522 unique questions, we collected 26,641 responses pairs.

GSM8K (Cobbe et al., 2021) This dataset features high-quality grade school math problems created by humans, requiring 2 to 8 steps and basic arithmetic operations (+, -, /, *) to solve. We utilized the GSM8K dataset from Cobbe et al. 2021, which includes a ground truth response and four generated responses from four finetuned models for each question. The ground truth was used as the 'chosen' preference response, and the other four as 'rejected', forming four sets of preference responses per question.

Human ChatGPT Comparison Corpus (HC3) (Guo et al.) This dataset comprises questions answered by both humans and ChatGPT. We used only the wiki_csai section of the HC3 dataset, as it was pertinent for a STEM responder model. For each question, the human response was taken as the 'chosen' and the ChatGPT response as the 'rejected'.

Stack Exchange (Lambert et al., 2023) This dataset includes questions and answers from the Stack Overflow Data Dump for preference model training. We selected relevant subsets (math, physics, computer science, etc.), preprocessing the data by removing questions with hyperrefs and extraneous characters (e.g., <p>, ...). Each question had multiple responses, each with a score and an indication of whether it was selected. The selected response was used as the 'chosen' response, and the highest-scored unselected response as the 'rejected', keeping response pairs within the same quality range.

MMLU (Hendrycks et al., 2021) This dataset contains multiple-choice questions across various knowledge branches. We focused on STEM branches relevant to our training. Each question had four unique choices and responses, which we renamed to A, B, C, and D, and processed into the format required for training.

3.2 Selected Base Model

In April 2024 Apple released OpenELM (Mehta et al., 2024a), a family of Open Efficient Language Models. OpenELM uses a decoder-only transformer architecture, similar to other LLMs like GPT-3. We decided to use the OpenELM-450M-Instruct model over it's larger conterparts to minimize training time. OpenELM has key differences in architecture compared to GPT-3:

Layer-wise scaling is a technique which adjusts the number of attention heads and the feed-forward network multiplier in each transformer layer. Layers closer to the input have fewer parameters, while layers closer to the output have more. This efficient parameter allocation improves accuracy without increasing model size.

No learnable bias parameters OpenELM does not include learnable bias parameters in any fully-connected layers, setting it apart from some other language models.

3.3 DPO implementation

DPO was implemented to align the model with preference data effectively. DPO focuses on refining the model's responses to better match user preferences, thereby improving the overall user experience. This alignment process involved collecting a substantial amount of preference data from users and utilizing this data to guide the model's training. By prioritizing responses that are more likely to satisfy user preferences, the model's relevance and accuracy in generating desired outputs were significantly enhanced. This approach also involved iterative feedback loops, allowing continuous improvement and adaptation based on evolving user needs.

3.4 MCQ finetuning implementation

We had two approach to implement the MCQ which had unique challenges:

1. Supervised Finetuning (SFT) using MCQ data and extraction of answer using post-processing
2. Classifier head added on top of OpenELM model and trained using LoRa.

The Supervised Finetuning was done using the TRL library from HuggingFace (HuggingFace, 2024). More specifically we used the SFTrainer which optimized to allow fine-tuning of pre-trained models using smaller datasets on supervised learning tasks. Once our model was trained we experimented with two postprocessing strategies to extract a single letter answer from our finetuned model. We elaborate on this experimentation in section [4].

The classifier head was implemented by adding a new linear layer to the model. This layer mapped the output dimension, also known as the model dimension in the configuration settings (OpenELM, 2024), to the desired output classes. The linear layer was designed to map this 1538-dimensional output to four distinct classes. These classes corresponded to the output letters of the multiple-choice questions (MCQ). By incorporating this linear layer, the model could effectively handle the classification task required for the MCQs. No post-processing was required as the model only outputted a single letter. To train this custom model we use the HuggingFace Trainer and configured LoRa to minimize training time:

```
lora_config = LoraConfig(
    r=16,
    lora_alpha=32,
    lora_dropout=0.05,
    bias="none",
    task_type="CAUSAL_LM",
    target_modules=[
         # attention projection
        "attn.qkv_proj",
        "attn.out_proj",
         # feedforward layer
        "ffn.proj_1",
        "ffn.proj_2",
        # custom classifier head
        "classifier",
    ],
)
```

It should be noted that the target module classifier refers specifically to the newly added linear layer head in our classifier implementation.

3.5 Quantization approach

Lastly, quantization was employed to optimize the model's performance and efficiency. Quantization involves reducing the precision of the model's weights, which significantly decreases the computational resources required for inference without substantially affecting the model's accuracy.

In our case, we experimented with quantizing the weights, but because of our model's specific characteristics, libraries like "Quanto" from Hugging Face were not compatible.

We decided to downsample our model instead, which led to worse results but allowed us to reduce its size. By using bf16, we managed to cut the base model's size by half while maintaining similar precision.

4 Experiments

4.1 Data

The data we used to finetune our model for answering MCQs was comprised of 15000 samples. We evaluated our accuracy on 1500 samples. The format of the dataset we used was the following :

```
{"subject":"college_physics",
    "question":"Question: The number of
    values of the quantum number is:
    \n\nOptions:\nA. 5 \nB. 4, \nC. 3
    \nD.\nAnswer:","answer":"A"}
```

4.2 MCQ Evaluation Method

For the model enhanced with a classifier head, it outputs an array of four numbers, each representing the probability of selecting a specific multiple-choice option. The option with the highest probability is chosen as the predicted answer. To determine the accuracy, we compare the predicted answers to the actual answers, calculating the percentage of correct predictions. This straightforward approach allows us to assess how well the model performs in selecting the correct multiple-choice option.

For the finetuned model, we have to perform some post-processing to extract the answer from the outputted tokens to calculate accuracy. We used two post-processing strategies:

Occurence based extraction - This method worked by counting the number of times on the options 'A', 'B', 'C' or 'D' was generated. From there we would output the model answer to be the most frequent letter. For this method we considered the whole output, and ignored the attention mask/when the sequence ended.

First Output extraction - This method worked by looking at the first letter outputted after the series of token "Answer :". The first letter outputed was selected as the final response of the model.

Once either method used, the accuracy can be calculate the same as with the model enhanced with a classifier head.

4.3 Baselines

We considered a theoretical baseline accuracy of 25%, assuming the model would choose answers randomly.

4.4 Experimental Details

For the MCQ Model the training process for the full dataset across 5 epochs took approximately 2 hours. Additionally, we conducted smaller training sessions using 3,000 samples to explore different hypotheses, which typically took between 15 and 30 minutes. We experimented with various learning rates and configurations of Lora. We evaluated both the model enhanced with a classifier head and the finetuned model with different preprocessing techniques.

For the DPO model used the total 60000 data samples from the data sources. Similarly to the MCQ model we experimented with different learning rates in training which took between 15 to 30 minutes.

4.5 DPO Results

To find the optimal learning rate we decided to train on smaller dataset of 5000 samples and evaluation on each epoch with 1000 samples. The table shows the learning rate and the accuracies during training and evaluation obtained:

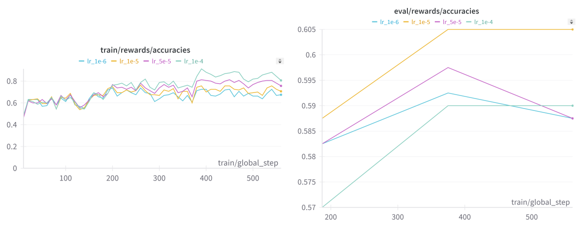

Figure 1: Training accuracy at each training step & Evaluation accuracy at each epoch.

Based on this, a learning rate of 1e-5 was used for the rest of the training as it showed the best accuracy.

Table 1 shows the final training arguments for the model which reached 0.63 % accuracy;

Argument	Value
Learning rate	1e-5
Train batch size	16
Epochs	3
Weight decay	0.01
Gradient accumulation steps	2

Table 1: Final Training Arguments

We have decided on batch size of 16 as this was the max batch size that did not max out the memory usage in the GPU. With a gradient accumulation step of 2, we can effectively double the batch size to 32 by accumulation the gradient over multiple iterations before updating the model weights.

4.6 MCQ Results

SFT finetuned model with post processing In the table below you can find the results of SFT finetuned model, with the two post processing strategy - occurence based extraction and first output extraction

Extraction	lr	Epoch	Accuracy
Occurences	1e-4	5	0.23
First Output	1e-4	5	0.27

Table 2: Results for the finetuned model

Interestingly, the results hover around, or even dip below, what we would expect from random guesses. We dive deeper into possible reasons for this in the Analysis section.

Model enhanced with classifier head For the model enhanced with the classifier head we tested various learning rates to identify an optimal setting. Here's how it looked:

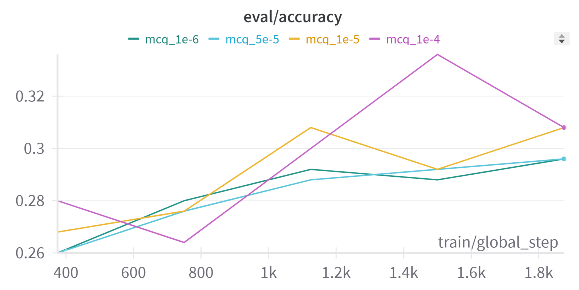

Figure 2: Evaluation Accuracy during Training

It's clear from the graph above that all tested learning rates achieved accuracies above our baseline. The standout was the learning rate of 1e-4, which delivered the best performance. Based on these results, we decided to extend our training sessions using this optimal learning rate of 1e-4:

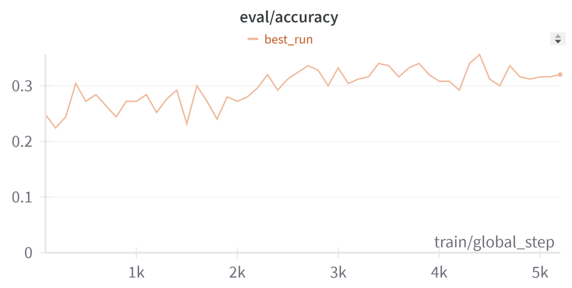

Figure 3: Evaluation Accuracy during Training

This training session yielded our best results, with the model achieving an accuracy of 0.36. It started at a baseline accuracy of 0.25 and gradually improved. The training process showed several fluctuations in performance, but overall, there was a steady upward trend in accuracy.

5 Analysis

SFT model for MCQ To dive deeper on why the accuracy hovered around 0.25 with both extraction strategy we will look at the output. Here's an example of the output before extraction:

```
'\n', 'A', '.', '', '1', ',', '\n','B',
'.', '', '1', ',','', '\n', 'C', '.',
'', '2', ',','', '', '\n','D'
```

We can directly see that the model is only outputing what seems to be random letters. However the letter always were either 'A', 'B', 'C' or 'D'. Suggesting the model was attempting to minimize loss by averaging out predictions, which effectively meant it wasn't learning properly. Unfortunately, the various extraction strategies we tried didn't address the core issue.

Model enhanced classifier head By using a classifier head, we restrict the model's outputs to just one of the letters. This allows us to eliminate the extraction process and ensures the model can't choose all four letters—it has to pick just one!

For the model enhanced with the classifier produced better results however we encountered overfitting problems. The orange line represents our best model, and longer training times led to overfitting the training data. You can clearly see the evaluation loss increasing rapidly once the model starts overfitting.

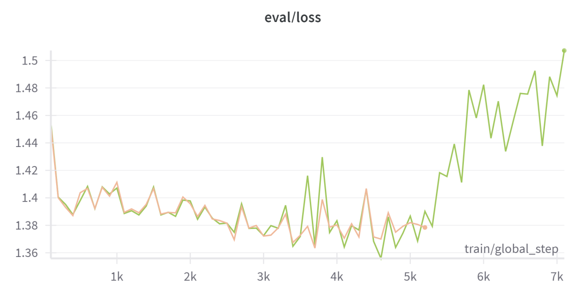

Figure 4: Model with Classifier Head Overfitting

With more time and resources, we could increase the number of samples and the model's complexity to prevent it from overfitting as quickly.

DPO Analysis In the figure below you can see the reward margin, defined as the difference between the rewards of the chosen prompts and the rejected prompts, increases as the number of training steps progresses. This increasing reward margin indicates improvement in the model's ability to distinguish between favorable and unfavorable outputs.

6 Ethical considerations

Language Adaptation To adapt our pipeline for different languages, simple translation of the model's inputs and outputs could yield good results for high-resource languages like French and German but may not suffice for accurate responses in all languages. Research showed (Lai et al., 2023) that while translation models perform well on high-resource languages like French and German, they struggle with low-resource languages like Urdu and Swahili.

Sign Languages Access to STEM resources in sign language is challenging due to the limited availability of materials. A model like ours that provides responses to STEM questions could be very useful to provide new resources. However, translating the inputs and outputs of our model into sign languages is difficult, much like with other low-resource languages. Finding datasets to train a model on STEM lexicons in sign languages is particularly challenging. One approach is to use a broad dataset of sign language to train a translator and employ fingerspelling for words not found in the dictionary. Despite this, there will still be a gap between the training data and the target STEM-specific content.

Impact on Learning and Dependency risks Our work aims to provide more accurate assistance to students in understanding STEM problems and working more efficiently. However, students may benefit from more accurate responses but could also be disadvantaged by reduced opportunities for critical thinking and collaboration with peers and professors. Over-reliance on our model may diminish the social aspects of learning. Beyond academia, our model could be used in industry to replace experts, posing potential problems due to over-reliance. Users must critically evaluate the model's outputs to avoid errors or misunderstandings.

Addressing Biases in Training Data Our model is trained on preference datasets from EPFL students, which inherently introduces biases, such as demographic imbalance. Strategies like data augmentation (e.g., including phrases in masculine and feminine forms) and adversarial training can help mitigate biases in both the questions answered and the preference responses chosen by users. According to the authors of OpenELM (Mehta et al., 2024b), models like ours trained on publicly available datasets lack safety guarantees, potentially leading to outputs that are inaccurate, harmful, biased, or objectionable. Therefore, rigorous safety testing and implementation of filtering mechanisms are essential. To assess potential harm in model outputs, we can employ tools such as dehatebert-mono-english (Aluru et al., 2020), which classifies text as offensive or not, thereby helping to mitigate risks associated with biased or harmful outputs.

Environmental Impact Finally, we recognize that large models significantly contribute to the carbon footprint. Using Direct Preference Optimization (DPO) instead of Reinforcement Learning from Human Feedback (RLHF) means we do not need to train reward models in addition to the primary model, resulting in less overall training. Nevertheless, training models use resources (Wei et al., 2023), so techniques to reduce model size are crucial to consider, that is why we used quantization techniques.

By addressing these ethical considerations and implementing appropriate safeguards, we aim to maximize the benefits of our model while minimizing potential risks to users and society at large.

7 Conclusion

In this project, we developed SuriChat, a large language model (LLM) designed to provide detailed explanations on various engineering topics covered in EPFL courses and to accurately answer multiple-choice questions (MCQs). Our primary goal was to address the challenges associated with the complexity and diversity of engineering subjects and to improve the accuracy and coherence of explanations provided by the model.

To achieve this, we employed Direct Preference Optimization (DPO) to fine-tune the OpenELM-450M-Instruct model. To specialize even further our model we trained a classifier head significantly enhanced the model's ability to answer multiple choice questions.

Our experimental results demonstrated significant improvements in model performance, achieving an accuracy of 0.36% in selecting the correct multiple-choice answer, compared to the baseline accuracy of 0.25%. This indicates that the approach of combining DPO with the new classifier head is effective in improving the model's performance for MCQ answering.

Beyond achieving higher accuracy in selecting the correct answers, it would be highly interesting to delve into understanding the model's chain of thought. Analyzing how the model arrives at its conclusions can provide valuable insights into its decision-making process, revealing the logical steps it follows and the information it prioritizes.

While our approach showed promising results, there are several areas that could be improved. These include a better quantization process, exploring more advanced methods like QLoRa, and expanding the dataset to include a more diverse range of questions and topics.

References

Sai Saketh Aluru, Binny Mathew, Punyajoy Saha, and Animesh Mukherjee. 2020. Deep Learning Models for Multilingual Hate Speech Detection. ArXiv:2004.06465 [cs].

Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. SciBERT: A Pretrained Language Model for Scientific Text. ArXiv:1903.10676 [cs].

Paul Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2023. Deep reinforcement learning from human preferences. ArXiv:1706.03741 [cs, stat].

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. *arXiv preprint arXiv:2110.14168*.

Isabel O. Gallegos, Ryan A. Rossi, Joe Barrow, Md Mehrab Tanjim, Sungchul Kim, Franck Dernoncourt, Tong Yu, Ruiyi Zhang, and Nesreen K. Ahmed. 2024. Bias and Fairness in Large Language Models: A Survey. ArXiv:2309.00770 [cs].

Biyang Guo, Xin Zhang, Ziyuan Wang, Minqi Jiang, Jinran Nie, Yuxuan Ding, Jianwei Yue, and Yupeng Wu. How close is chatgpt to human experts? comparison corpus, evaluation, and detection. *arXiv preprint arxiv:2301.07597*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*.

HuggingFace. 2024. Supervised fine-tuning trainer.

Viet Dac Lai, Nghia Trung Ngo, Amir Pouran Ben Veyseh, Hieu Man, Franck Dernoncourt, Trung Bui, and Thien Huu Nguyen. 2023. ChatGPT Beyond English: Towards a Comprehensive Evaluation of Large Language Models in Multilingual Learning. ArXiv:2304.05613 [cs].

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

Sachin Mehta, Mohammad Sekhavat, Qingqing Cao, Max Horton, Yanzi Jin, Frank Sun, Iman Mirzadeh, Mahyar Najibikohnehshahri, Dmitry Belenko, Peter Zatloukal, and Mohammad Rastegari. 2024a. Openelm: An efficient language model family with open training and inference framework.

Sachin Mehta, Mohammad Hossein Sekhavat, Qingqing Cao, Maxwell Horton, Yanzi Jin, Chenfan Sun, Iman Mirzadeh, Mahyar Najibi, Dmitry Belenko, Peter Zatloukal, and Mohammad Rastegari. 2024b. OpenELM: An Efficient Language Model Family with Open Training and Inference Framework. *arXiv.org*.

Apple OpenELM. 2024. Openelm-450m-instruct config.

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. ArXiv:2305.18290 [cs].

Xiaokai Wei, Sujan Kumar Gonugondla, Shiqi Wang, Wasi Ahmad, Baishakhi Ray, Haifeng Qian, Xiaopeng Li, Varun Kumar, Zijian Wang, Yuchen Tian, Qing Sun, Ben Athiwaratkun, Mingyue Shang, Murali Krishna Ramanathan, Parminder Bhatia, and Bing Xiang. 2023. Towards Greener Yet Powerful Code Generation via Quantization: An Empirical Study.