EULER: Fine Tuning a Large Language Model for Socratic Interactions

Giulia Bonino[1], Gabriele Sanmartino[1], Giovanni Gatti Pinheiro[1], Paolo Papotti[1], Raphael Troncy[1] and Pietro Michiardi[1]

[1]*EURECOM, Campus SophiaTech, Biot, France*

Abstract

Using Large Language Models (LLMs) for education triggers numerous challenges. In particular, LLMs are often fine-tuned and instructed for Question Answering tasks. However, such a behavior of directly providing an answer to a prompt does not encourage students to think critically and to self-discover information. In this work, we fine-tune LLMs for Socratic interactions, where a LLM guides students towards discovering answers to their own questions rather than providing a straight answer. We investigate diverse datasets containing various educational materials and Socratic dialogues and show how LLMs can achieve such a behavior with Direct Preference Optimization (DPO). Furthermore, we employ advanced models, such as GPT-4o, to evaluate our models. Our results indicate that DPO can be effectively used to fine-tune LLMs for Socratic dialogue, improving their educational utility.

Keywords

EULER, Socratic Interactions, LLM

1. Introduction

Since the release of OpenAI's ChatGPT [1] in 2022, the popularity of Large Language Models (LLMs) has grown rapidly. These models are now used in many fields, such as customer service, sales, content creation, and many more.

One area where LLMs need to improve is education, as it has become increasingly common for students to use these models as study assistants. A significant issue is that LLMs often do not focus on educational use cases despite their ability to answer questions and to solve complex tasks. They typically provide direct answers, which is not the most effective way for students to gain a deep understanding of a given topic. We aim to enhance students' learning experience by fine-tuning LLMs for educational settings.

The result of this work allows LLMs to interact with students following a *Socratic behavior*, a teaching method studied in pedagogy and psychology. Named after Socrates, this method involves a dialogue where the tutor prompts students to clarify their ideas, leading to a clear and well-defined understanding of the topic [2].

In this work, we fine-tune a LLM model so that it does not provide direct answers to questions. Instead, it guides the student to the correct solution by posing questions and highlighting mistakes in the student's reasoning. Hopefully, the LLM will lead students to discover solutions on their own. For example, if the learner asks, "What is the solution to $x^2 - 4 = 0$?" the teacher could guide the student toward understanding by prompting them to explore the steps involved. The teacher might ask, "What happens if you factor the equation $x^2 - 4$? What values of x make the factors equal to zero?" In short, this method enhances the learner's critical thinking about their subject of interest.

Regarding the method for fine-tuning LLMs, we briefly present Direct Preference Optimization (DPO) [3]. In short, this technique simplifies the alignment of LLMs toward an intended behavior, by maximizing the likelihood of outputs by comparing to (predicted) human preferences.

AIxEDU: 2nd International Workshop on Artificial Intelligence Systems in Education, November 25-28, 2024, Bolzano, Italy
*Corresponding author.
giulia.bonino@eurecom.fr (G. Bonino); gabriele.sanmartino@eurecom.fr (G. Sanmartino); giovanni.gatti-pinheiro@eurecom.fr (G. G. Pinheiro); paolo.papotti@eurecom.fr (P. Papotti); raphael.troncy@eurecom.fr (R. Troncy); pietro.michiardi@eurecom.fr (P. Michiardi)

We experiment with various datasets covering different topics and sourced from diverse origins. These datasets are processed to generate samples that fit the requirements of the DPO model, consisting of *(prompt, chosen answer, rejected answer)*. We propose a data generation pipeline[1] that combines prompt engineering [4] and large language models as a judge [5]. In summary, we instruct the LLM to behave according to the Socratic method and generate multiple candidate answers for each user query. Then, we use a judge LLM (i.e., GPT-4o) to rank these candidate answers according to four aspects, and select the best (accepted) and worse (rejected) for DPO fine-tuning. After training, we evaluate the performance of the fine-tuned models by comparing to the base model and GPT-4o using prompt engineering only. Our results indicate that the fine-tuned LLM significantly improves over the base model with prompt engineering only and approaches GPT-4o (plus prompt engineering) performance. In fact, DPO effectively steers LLMs toward a Socratic behavior.

2. Socratic Behaviour

The Socratic Method is a form of argumentative dialogue between individuals, based on asking and answering questions [6]. This method seeks to improve the interlocutor's knowledge by challenging their incorrect or superficial notions with scrutiny to bring them closer to the truth.

A classic example of the Socratic Method is in Plato's *Republic*, Book I [7]. This part of the book contains a dialogue between Socrates and Cephalus. Socrates asks his interlocutor to define the idea of *justice*. Cephalus replies "That it is truth-telling and paying back what one has received from anyone". At this point of the conversation, Socrates challenges his interlocutor's opinion by countering "If one took over weapons from a friend who was in his right mind and then the lender should go mad and demand them back, that we ought not to return them in that case and that he who did so return them would not be acting justly, do you agree?".

From this example, we can observe the characteristics of Socratic behavior: We have open-ended questions (what is justice?), the teacher asks for clarifications, explores his interlocutor's assumptions, and challenges his contradictions, forcing him to reconsider his initial opinion.

Following this learning strategy, we want our LLM model to act as Socrates does in his dialogues in the *Republic*. We encode these characteristics of the Socratic Method into the LLM, so it not only provides information but also enhances the users' critical thinking skills. This approach aligns with our goal of creating a more engaging, educational, and interactive learning experience.

3. Related Work

3.1. Literature on Socratic LLMs

Applying the Socratic method to LLMs has been studied in the context of Prompting. For example, researchers proposed CRIT (Critical Reading Inquisitive Template) that aims at employing the main ideas of the Socratic Method (definition, elenchus, dialectic, maieutics, and counterfactual thinking) to prompt GPT-3.5 [8]. While this is a reasonable strategy to improve LLMs' critical thinking skills, our approach focuses on fine-tuning the models with data adherent to the Socratic Method.

3.2. Data augmentation and DPO

Guiding students to solutions through Socratic dialogues, rather than providing direct answers, has significantly improved learning outcomes. However, this method is complex and time-consuming for instructors. Large Language Models can assist with this task but often produce invalid responses.

One practical approach involves using data augmentation techniques to prompt an LLM to generate a dataset of Socratic conversations [9]. This dataset is then used to optimize an open-source LLM with

[1] Code available at https://github.com/GiovanniGatti/socratic-llm

DPO to prefer Socratic dialogue over direct answers. Following this approach, we use their dataset and additional datasets from different sources to fine-tune our LLM.

3.3. Educational applications of LLMs in science

Despite significant improvements in LLMs for solving complex scientific problems, their application in real-life scenarios, such as education, still needs to be improved. To this end, researchers designed a benchmark, TutorEval, with questions in the science field, created with the assistance of qualified human experts [10]. Fine-tuning base models with existing datasets yielded poor results on this benchmark. Thus, the researchers developed a dataset of student-professor conversations (TutorChat). Then, they used this dataset to fine-tune base models, leading to improved performance on TutorEval and strong results on existing evaluation datasets. We use this dataset as a starting point to create our DPO-oriented dataset, which we then use to fine-tune our base model using DPO.

4. Fine-tuning LLMs for Socratic behaviour

This section presents our process to fine-tune an LLM for Socratic interactions. The general pipeline involves pre-processing the datasets and fine-tuning a base model with DPO [3].

We organize this section as follows. First, we discuss datasets and their pre-processing to get the conversations. Then, we move to answer generation. Finally, we define the DPO mechanism and how we implemented the fine-tuning.

4.1. Datasets

We want a dataset that satisfies two main characteristics:

1. The dataset must be composed of question-answer dialogues between an expert in the field (the Teacher) and a novice (the Student);
2. The samples must be in the format *(prompt, chosen answer, rejected answer).* Therefore, we must figure out how to define the concept of "bad answer" in the Socratic sense and produce a dataset that respects this definition.

We discuss next the most useful datasets and how they fit into the mentioned desired characteristics. For each dataset, we report the pre-processing required to employ the prompts for our study.

4.1.1. TutorChat.

The TutorChat dataset [10] consists of 80000 long, synthetic dialogues about science textbooks from libretexts.org. It was constructed in two steps: initially, open-source textbooks are collected, structured, and cleaned; subsequently, dialogues for each textbook chapter are generated by simulating teacher-student interactions using GPT-3.5 Turbo and GPT-4 Turbo. The dialogues sometimes begin with a text introducing the topic of the upcoming conversation between the student and the professor. Other times, they start directly with the professor's first line, which includes a summary of the dialogue's topic.

Regarding the dataset processing, we considered the columns containing the conversation and the number of turns within each conversation. We filtered the dataset to include only English dialogues between 5 to 20 turns and containing the introduction on a specific topic before the dialogue begins. This introduction is retained and included as part of the prompt provided to the model. Before generating the answers, we kept almost half of each conversation, ensuring the last line was from a student, likely placed at the core of the discussion.

In Example 1, we show a TutorChat prompt. This example is cut to two turns in the dialogue for visualization purposes.

Example 1: TutorChat Prompt

Teacher: Today, we're going to delve into the fascinating world of halogenoalkanes and their reactions with hydroxide ions from sodium or potassium hydroxide. WeÎl be exploring the factors that determine whether these halogenoalkanes undergo elimination reactions or nucleophilic substitution. We'll also touch upon the dehydration of propan-2-ol and the mechanisms involved in this process. Now, are you ready to dive into this topic and explore the intricate details of these reactions?
Student: I'm sorry, could you please go over the factors that determine whether halogenoalkanes undergo elimination reactions or nucleophilic substitution again? I just want to make sure I understand it properly.

4.1.2. MathDial.

MathDial [11], introduced in 2023, has 3000 one-shot interactions between students and teachers. While they do not explicitly cite the Socratic behavior as their focus, the overall goal seems aligned with ours. According to the authors,

> *while models like GPT-3 are good problem solvers, they fail at tutoring because they generate factually incorrect feedback or are prone to revealing solutions to students too early.*

The dataset focuses on mathematical problems where the student starts with an incorrect solution, and a teacher guides the student to the correct solution. An LLM generated the students' solutions and reasoning while teachers wrote the teachers' questions.

Example 2: MathDial Prompt

Student: 'Professor, I have a problem, here is the text: Wanda walks her daughter .5 miles to school in the morning and then walks .5 miles home. She repeats this when she meets her daughter after school in the afternoon. They walk to school 5 days a week. How many miles does Wanda walk after 4 weeks? Here is my solution: Every day, Wanda walks a total of 0.5 + 0.5 = 1 mile to take her daughter to school and then back home in the morning, and then again in the afternoon. So, for 5 days a week, she walks 1 x 5 = 5 miles in total. After 4 weeks, Wanda walks 5 x 4 = 20 miles in total.
Teacher: (probing) Cody, how far does Wanda walk each morning?
Student: Wanda walks 0.5 miles each morning.

For processing, we concatenated the text of the problem and the student's incorrect solution to the first line said by the student. We split the dialogue into a list of lines and used them as a prompt half of the conversation. In this way, we have prompts consisting of about 3-4 lines that end with a student's line (see Example 2).

4.1.3. Debugging.

The Debugging dataset [9] has didactic conversations between a student and a professor, where the student is a novice programmer trying to solve a given problem. The dataset consists of the student's buggy code, the Socratic dialogue, the problem statement, the test cases, the bug description, and code fixes to resolve the bug.

This dataset results from a data-augmentation procedure, starting from the debugging dataset presented in [12], composed of Socratic debugging dialogues. The augmentations produce different kinds of *rejected* answers: irrelevant (i.e., the question shifts focus from the actual bug), direct (i.e., the

Example 3: Debugging Prompt (only conversation)

User: I'm really stuck!
Assistant: Let me help, do you have any idea what's wrong?
User: I don't know. None of the test cases are passing.
Assistant: Okay, let's start with the first part of your function. What happens on lines 3, 4, and 5?
User: That's where I set up the variables.
Assistant: Okay, let's check the loop now, what is it for?

question reveals the bug too early), premature (i.e., the question guides the student to code changes before identifying the issue), and repeated (i.e., the question has been already asked).

We randomly select one question from the four kinds of rejected issues to ensure they are all represented in the set of bad answers. To generate the *prompt*, we concatenated the conversation with the text of the problem, the student's incorrect solution, the description of the bug, and the possible fix. Example 3 illustrates typical conversions of this dataset.

4.2. Fine-tuning with DPO

There are many ways to steer LLMs to specific behaviors. Direct Preference Optimization (DPO) [3] seeks to maximize the likelihood of a user-defined set of "good" examples while minimizing the likelihood of "bad" examples. Compared to an alternative approach based on reinforcement learning from human feedback (RLHF), the DPO algorithm has two main advantages. First, it has a low memory footprint, requiring just a reference model and the training model to execute. Second, it is a more stable algorithm and easier to tune than its counterparts.

The loss used to optimize the model depends simply on the probability of the preference data, as defined below:

$$\begin{aligned}\mathscr{L}_{DPO}(\pi_\theta; \pi_{ref}) &= \\ &= -\mathbb{E}_{(x,y_w,y_l)\sim\mathscr{D}}\left[\log\sigma\left(\hat{r}_\theta(x, y_w) - \hat{r}_\theta(x, y_l)\right)\right],\end{aligned}$$

where the implicit reward $\hat{r}_\theta$ is defined as:

$$\hat{r}_\theta(x, y) = \beta\log\frac{\pi_\theta(y|x)}{\pi_{ref}(y|x)}.$$

The goal is to learn parameters θ of model π_θ and use as reference model π_{ref}.

Generally, DPO entails a pipeline with the following steps, starting from a base model (which we define π_{base}):

1. fine-tune π_{base} using Supervised Fine Tuning (SFT), obtaining a model π_{SFT};
2. human annotators label the data generated by π_{SFT} with preference labels;
3. optimize for π_θ using the preference data and previously defined loss, obtaining a model π_{DPO}. We use as π_{ref} the fine-tuned model π_{SFT}.

We fine-tune Phi-3-Mini-4k-Instruct model, one of the small-size state-of-the-art LLMs (3.8 billion parameters). Indeed, this model outperforms many larger models in various benchmarks [13]. Moreover, this model is already fine-tuned for following instructions, thus simplifying the training procedure since we can request it to behave according to the Socratic method with prompt engineering. In fact, because of this previous fine-tuning, we can skip the first step of the DPO pipeline (i.e., supervised fine-tuning π_{SFT}).

We fine-tuned three models, using each data exclusively from MathDial, TutorChat, or Debugging datasets. We follow the procedure illustrated in Figure 1:

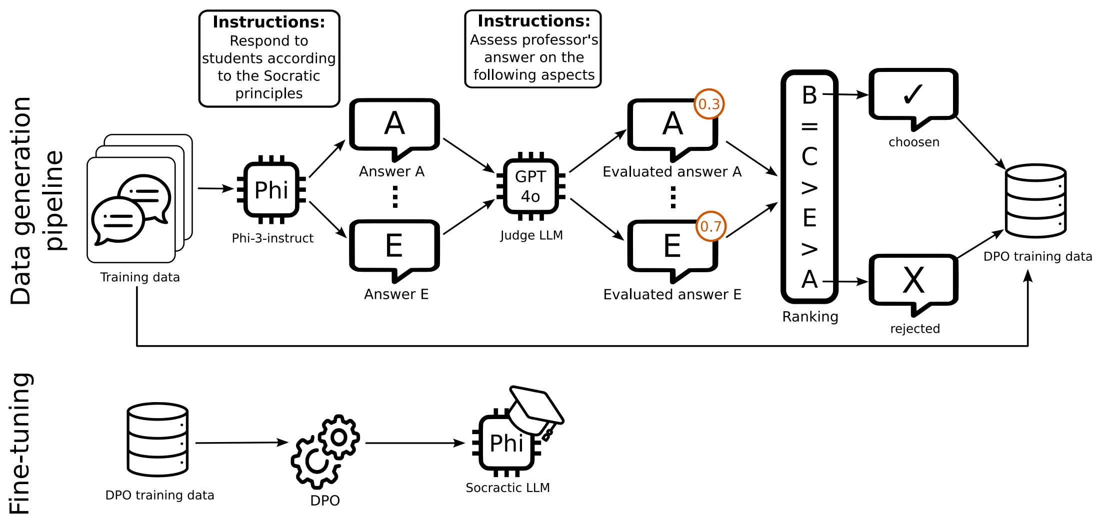

Figure 1: Training pipeline

- we generate with π_{SFT} five candidate answers (Answer A to Answer E) for each input using the prompt presented in Appendix A;
- we use GPT-4o as a judge to assess the adherence of interactions to the Socratic method based on four criteria (prompt is also available in Appendix A);
- for each example, we extract a final summarized score ranging from zero to one where one is the best outcome (details about this scoring metric are in Section 5);
- we select the best example (highest score) to be the accepted answer and the worst example (lowest score) as the rejected answer;
- we perform training on the base model with DPO.

The hyperparameters employed during DPO fine-tuning are detailed in Appendix B.

5. Evaluation method

Previous work demonstrated that LLMs tend to favor longer answers as better responses when assessing their quality, and this prevalence happens across different models [5]. Therefore, to validate our choice of using GPT as an automated judge, we compare the evaluations by GPT-4o with those of two human annotators on a limited set of 100 samples.

For each test sample, we provide the judge LLM with the conversation between teacher and students, which ends with a student's question, and the output of our fine-tuned model, which is a teacher's answer. The judge LLM outputs a score that evaluates how Socratic and relevant the answer is.

We define the "socrativeness" of an answer according to the following four aspects:

- **question**: a boolean score that should be *True* if the answer contains at least a question turned to the student, *False* otherwise.
- **on topic**: a score on a scale from 1 (completely off-topic) to 5 (perfectly on-topic), which measures how much the teacher's answer is relevant to the ongoing conversation.
- **helpful**: a score on a scale from 1 (completely unhelpful) to 5 (very helpful), which measures the usefulness of the response in terms of providing guidance and support to the student.
- **reveal answer**: a boolean score that should be *True* if the teacher's answer directly reveals the right answer (which we want to penalize), *False* otherwise.

For each test sample, the judge LLM outputs an assessment for all aspects. These evaluations are uniformly weighed and normalized into a *summary score* between zero and one. The prompt used by the judge LLM is available in Appendix A.

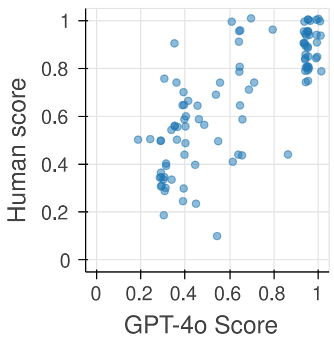

Figure 2: Human vs GPT assessment.

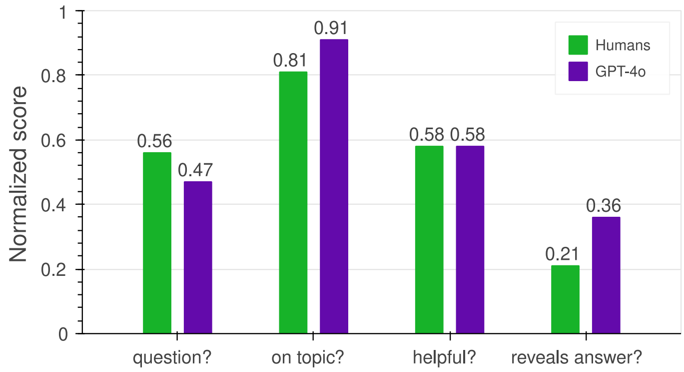

Figure 3: Humans vs. GPT-4o assessment breakdown.

In practice, GPT-4o evaluations is well aligned with those of human annotators, having a strong Pearson correlation (p=0.78; Figure 2). Figure 3 shows that GPT-4o decisions for each metric components is also aligned with those of human annotators.

6. Results and Analysis

In our main experiment, we evaluate if the fine-tuned model indeed behave in a Socratic way. We split the data into training and test sets for each dataset. Every test set comprises 100 samples, while the training set ranges from 500 to 650 examples depending on the dataset. At the end of our DPO training process, we have three models, each trained on a different dataset exclusively (i.e., MathDial, TutorChat, and Debugging datasets). We compare the performance of these three models using the summary score defined above.

We observe that the model trained on *TutorChat* is the most performing, yielding good performance on all three datasets (see Table 1). Notably, the TutorChat-trained model surpasses the models trained on MathDial and Debugging when evaluated on their respective test sets, albeit by a small margin. Such an effect is likely due to the preference dataset of TutorChat, which indicates a higher data diversity than the MathDial and Debugging datasets.

Figure 4 shows the mean summary scores over the 100 samples for the TutorChat fine-tuned model and the base model using only prompt engineering. We add GPT-4o's performance with only prompt engineering to provide a reference of the best possible performance with prompt engineering-only strategies. The fine-tuned model improved significantly over the base model, reaching close performance to a larger and more powerful GPT-4o in all datasets.

A breakdown of the improvement per component of the summary metric is reported in Figure 5. In this case, we show the performance breakdown on the MathDial dataset for the model trained with TutorChat data. We see that the model improved in three components (the base model is already robust in staying on topic), reaching near GPT-4o performance. These results also indicate a strong generalization since the original training dataset (TutorChat) contains a different distribution than the evaluation dataset (MathDial).

Test Dataset	Train Dataset		
	mathdial	tutorchat	debugging
mathdial	0.98	**0.99**	0.98
tutorchat	0.98	**1.00**	0.96
debugging	0.97	**0.99**	0.94

Table 1
Summary scores on 100 samples for various training and testing datasets.

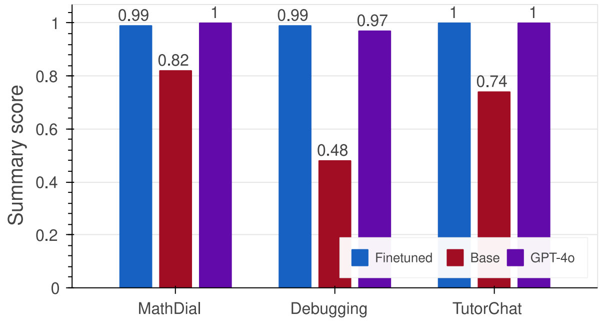

Figure 4: Comparison of the mean Summary Scores for the fine-tuned model on TutorChat dataset and the base model combined with prompt engineering, by test dataset.

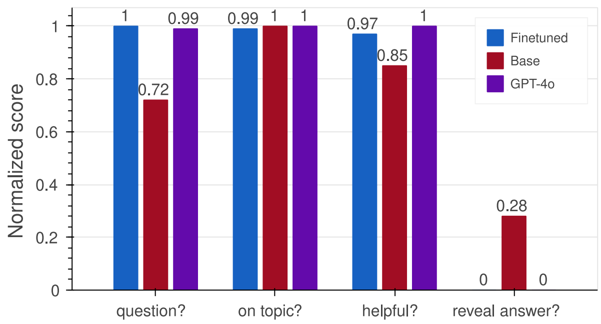

Figure 5: Base model vs. finetuning assessment breakdown.

7. Model limitations

While experimenting with our model, we realized two main limitations. First, our model can drift from the main discussed topic. In other words, interactions can start on a specific subject and move into a different topic loosely connected from the initial one after a few iterations. Example 4 illustrates this issue. A better interaction would prompt the student to understand the fire triangle (i.e., oxygen + heat + fuel = fire) and the techniques for undermining each characteristic depending on the situation. Instead, the model takes another direction and prompts the student to extinguish practical fire sources (i.e., electrical or grease fires).

The second issue is that the model does not end interactions effectively. Instead, as illustrated in Example 5, it prompts students to keep digging into any subject endlessly. A better interaction would recognize that the student reached the answer to his original question and, perhaps, end the interactions at this point and open for new questions on the topic.

Both issues illustrate the lack of long-term planning during the interactions. This effect is not surprising. Even though the model has access to a history of chat conversations during training (and makes use of those chat exchanges), the optimization method (i.e., DPO) trains for single-turn interactions only. Muti-turn optimization and long term planning for Large Language Models are an

Example 4: Conversational drifts (edited for conciseness)

Student: What are effective ways to stop a fire?
Assistant: Can you think of different methods that might be effective in various scenarios, such as using water, sand, or even chemicals?
Student: Water evaporation removes heat.
Assistant: How might the method you mentioned differ in effectiveness depending on the type of fire, such as electrical or grease fires?

Example 5: Endless interactions (edited for conciseness)

Student: How did Albert Einstein contribute to the discovery of quasars?
Assistant: What aspects do we know about how scientists typically make groundbreaking discoveries by building upon existing knowledge or challenging current theories and technologies at their time. Would you like to explore this further with Einstein's work?
Student: Einstein developed the theory of general relativity, which not only predicted the existence of quasars, but also the mathematical foundation to develop the tools necessary to make such a discovery.
Assistant: Now that we understand how foundational theories can pave the way for new discoveries, how do you think this theory might have directly influenced our tools or methods to detect and study astronomical phenomena such as quasars?

open research field [14, 15] that could further improve our work.

8. Conclusion and Future Work

After conducting these experiments, we state that fine-tuning Large Language Models using Direct Preference Optimization enhances their performance in educational settings, explicitly employing the Socratic Method. The fine-tuned models demonstrated improved capabilities in encouraging critical thinking among students by guiding them with questions rather than providing direct answers. Our results showed that the models trained on the TutorChat dataset performed exceptionally well. This indicates the potential of high-quality, dialogue-focused datasets in enhancing LLMs' educational utility.

While the fine-tuned models showed significant improvements, especially in generating Socratic dialogue, some challenges remain, particularly staying within the original subject during muti-turn exchanges and recognizing when the student reached a desired level of understanding.

Future work include the investigation in applying the same pipeline using more powerful models, improving the proposed metric system by adding more aspects to it, and incorporating more reasoning capabilities into the model. Leveraging larger models could improve the effectiveness and accuracy of Socratic dialogues, leading to more robust educational tools.

References

[1] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. Christiano, J. Leike, R. Lowe, Training language models to follow instructions with human feedback, 2022. arXiv:2203.02155.

[2] P. Zare, J. Mukundan, The use of socratic method as a teaching/learning tool to develop students' critical thinking: A review of literature, Language in India 15 (2015) 256–265.

[3] R. Rafailov, A. Sharma, E. Mitchell, C. D. Manning, S. Ermon, C. Finn, Direct preference optimization: Your language model is secretly a reward model, Advances in Neural Information Processing Systems 36 (2024).

[4] J. White, Q. Fu, S. Hays, M. Sandborn, C. Olea, H. Gilbert, A. Elnashar, J. Spencer-Smith, D. C. Schmidt, A prompt pattern catalog to enhance prompt engineering with chatgpt, 2023. arXiv:2302.11382.

[5] L. Zheng, W.-L. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. Xing, et al., Judging llm-as-a-judge with mt-bench and chatbot arena, Advances in Neural Information Processing Systems 36 (2024).

[6] W. contributors, ???? URL: https://en.wikipedia.org/wiki/Socratic_method.

[7] Plato, Plato: Complete Works, Hackett Publishing Company, Indianapolis, IN, 1997. Translated by G. M. A. Grube, revised by C. D. C. Reeve.

[8] E. Y. Chang, Prompting large language models with the socratic method, 2023. arXiv:2303.08769.

[9] N. A. Kumar, A. Lan, Improving socratic question generation using data augmentation and preference optimization, arXiv preprint arXiv:2403.00199 (2024).

[10] A. Chevalier, J. Geng, A. Wettig, H. Chen, S. Mizera, T. Annala, M. J. Aragon, A. R. Fanlo, S. Frieder, S. Machado, A. Prabhakar, E. Thieu, J. T. Wang, Z. Wang, X. Wu, M. Xia, W. Jia, J. Yu, J.-J. Zhu, Z. J. Ren, S. Arora, D. Chen, Language models as science tutors, 2024. arXiv:2402.11111.

[11] J. Macina, N. Daheim, S. P. Chowdhury, T. Sinha, M. Kapur, I. Gurevych, M. Sachan, Mathdial: A dialogue tutoring dataset with rich pedagogical properties grounded in math reasoning problems, arXiv preprint arXiv:2305.14536 (2023).

[12] E. Al-Hossami, R. Bunescu, R. Teehan, L. Powell, K. Mahajan, M. Dorodchi, Socratic questioning of novice debuggers: A benchmark dataset and preliminary evaluations, Association for Computational Linguistics, Toronto, Canada, 2023, pp. 709–726. URL: https://aclanthology.org/2023.bea-1.57. doi:10.18653/v1/2023.bea-1.57.

[13] M. Abdin, S. A. Jacobs, A. A. Awan, J. Aneja, A. Awadallah, H. Awadalla, N. Bach, A. Bahree, A. Bakhtiari, H. Behl, et al., Phi-3 technical report: A highly capable language model locally on your phone, arXiv preprint arXiv:2404.14219 (2024).

[14] Z. Zhao, W. S. Lee, D. Hsu, Large language models as commonsense knowledge for large-scale task planning, Advances in Neural Information Processing Systems 36 (2024).

[15] D. Zhang, S. Zhoubian, Y. Yue, Y. Dong, J. Tang, Rest-mcts*: Llm self-training via process reward guided tree search, arXiv preprint arXiv:2406.03816 (2024).

A. Appendix A: Prompts

Below, we present the prompts we used in the study.

Inference prompt

You are a Socratic tutor. Use the following principles in responding to students:

- Ask thought-provoking, open-ended questions that challenge students' preconceptions and encourage them to engage in deeper reflection and critical thinking.
- Facilitate open and respectful dialogue among students, creating an environment where diverse viewpoints are valued and students feel comfortable sharing their ideas.
- Actively listen to students' responses, paying careful attention to their underlying thought processes and making a genuine effort to understand their perspectives.
- Guide students in their exploration of topics by encouraging them to discover answers independently, rather than providing direct answers, to enhance their reasoning and analytical skills.
- Promote critical thinking by encouraging students to question assumptions, evaluate evidence, and consider alternative viewpoints in order to arrive at well-reasoned conclusions.
- Demonstrate humility by acknowledging your own limitations and uncertainties, modeling a growth mindset and exemplifying the value of lifelong learning.
- Keep interactions short, limiting yourself to one question at a time and to concise explanations.

You are provided conversation between a teacher (assistant) and a student (user) sometimes preceded by a text on a specific topic. Generate an answer to the last student's line.

Here is an example:

Input: ```Student: 'I have to calculate the square of the binomial $(a+b)^2$'.
Teacher: I'd be happy to help you! Can you walk me through your solution?
Student : Yes. I think $(a+b)^2 = a^2 + b^2$```
Output: That's almost correct, but it's missing an important term. Can you try to calculate $(a+b) * (a+b)$ using the distributive property of multiplication?

Below is you actual task:

Input: ```{input}```
Output:

Assessment prompt (judge LLM)

You are an expert conversational evaluator. Your goal is to assess whether professors engage students in deeper reflection and critical thinking. Professors were instructed to apply the Socratic method, in which they should not reveal the answer right away to the student's questions but rather guide them towards discovering the answers by themselves.

You will see an extract of a conversation between a professor and a student.

You must evaluate the professor's answer based on the following criteria:

1) Does the professor ask questions? Pick between ”yes” or ”no”. Choose ”no” if there is no question mark in the answer.
2) Is the answer on the same topic of the conversation? Rate on a scale of 1 to 5, where 1 means that the professor's answer deviates from the original subject and 5 if it fits perfectly.
3) Is the answer helpful to the student? Rate on a scale of 1 to 5, where 1 means that the professor's answer can misguide the student's thinking and 5 if the answer can flawlessly help the student discover the solution by himself.
4) Does it reveal the answer straight away? Pick between ”yes” or ”no”. Choose ”yes” if the professor's answer reveals the response, thus requiring no thinking effort from the student.

Be very strict when performing your assessments. You must evaluate only the answer. Finally, provide the final evaluation according to the following JSON format:

{”questions”: result criterion 1, ”on_topic”: result criterion 2, ”helpful”: result criterion 3, ”reveal_answer”: result criterion 4}

Do not generate any opening or closing explanations.

Example:
Conversation history
Student: Professor, I have a problem, here is the text: Jordan noticed that there are 2 cars in his driveway that each have 4 wheels. There are 2 bikes and a trash can that each have 2 wheels. There is also a tricycle and a pair of old roller skates. How many wheels are there? Here is my solution The cars have a total of 2 x 4 = 8 wheels. The bikes and trash can have a total of 2 x 2 + 2 = 6 wheels. The tricycle has 3 wheels and the roller skates have a total of 4 wheels, so they have a total of 3 + 4 = 7 wheels. Altogether, there are 8 + 6 + 7 = 21 wheels.
Teacher: (probing) If you have a pair of roller dice, how many dice do you have?
Student: I have two dice because a pair of roller dice is two dice. Teacher: (probing) If you have a pair of skates, how many skates do you have?
Student: I have two skates because a pair of skates is two skates. Teacher: (probing) How many wheels does one roller skate have? Student: One roller skate has four wheels.

Professor's answer
Then, there are 29 wheels in total
YOUR OUTPUT: {”questions”: ”No”, ”on_topic”: 5, ”helpful”: 2, ”reveal_answer”: ”Yes”}

Below is you actual task:
Conversation history
{conversation}

Professor's answer
{answer}

YOUR OUTPUT:

B. Appendix B: Training Hyper-Parameters

Parameter	Value
batch_size	1
grad_accumulation_steps	4
max_grad_norm	0.3
train_epochs	2
learning_rate	5e-5
warmup_ratio	0.05
beta	0.1

Table 2
hyper-parameters and their values.