ChatGPT uses a cheat sheet to supervise an AI tutor for engineering students

Juliette Parchet
AbyssalChat
EPFL
juliette.parchet@epfl.ch

Guillaume Vray
AbyssalChat
EPFL
guillaume.vray@epfl.ch

Leo Wolff
AbyssalChat
EPFL
leo.wolff@epfl.ch

Abstract

The demanding nature of prestigious technical schools often leads to stress, anxiety, and burnout among students. Psychological distress rates at the EPFL (Swiss Federal Institute of Technology of Lausanne) were found to be remarkably high last November 2022, in a study conducted on campus[5]. A central solution at EPFL is the targeted and personalized help provided by knowledgeable assistants to students struggling in their studies. However, it requires significant human resources to support individual students on campus, and with this in mind, AI tutoring has emerged as a field capable of providing personalized and interactive educational support. Various aspects of AI tutoring, including intelligent tutoring systems[11], adaptive learning[13], or natural language processing (NLP)[15], have positively impacted student learning, performance, and engagement. In this work, we introduce an AI tutor tailored for technical school courses (explicitly focusing on EPFL). When given a question from an exercise, the tutor must provide an in-depth explanation to the student to educate them on how to solve the specific exercise. We used a training methodology similar to modern AI chatbots such as ChatGPT. The approach we used involves three key steps.

At first, we collect supervised fine-tuning data by distilling technical question demonstrations via ChatGPT using prompt engineering techniques, while a human-written cheat sheet is employed to guide ChatGPT's answers and reduce occasional inaccuracies. A *step-by-step* prompting strategy is found to be effective to provide a meaningful explanation on how to answer the question. Secondly, the tutors are further trained using reinforcement learning with human feedback (RLHF), enabling them to learn from their generated demonstrations. A reward model is trained to effectively evaluate and assess the quality of the chatbot's demonstrations. Lastly, the supervised demonstrations are used to train the final AI tutor, resulting in a practical educational assistant. The quality of the demonstrations is measured using standard metrics and the trained reward model. The results demonstrate that the AI tutor developed in this study outperforms basic ChatGPT. The AI tutor shows improved accuracy and effectiveness in providing educational support, making it a valuable tool for technical school students. The application of reinforcement learning and the potential for further enhancements are identified as future research directions.

Preprint. Under review.

1 Introduction

The demanding nature of prestigious technical schools often leads to stress, anxiety, and burnout among students. Psychological distress rates at the EPFL (Swiss Federal Institute of Technology of Lausanne) were found to be remarkably high last November 2022, in a study conducted on campus, where 30.7% of respondents reported experiencing psychological distress[5]. A central solution at EPFL is the targeted and personalized help provided by knowledgeable assistants to students struggling in their studies. However, it requires significant human resources to support individual students on campus, and with this in mind, AI tutoring has emerged as a field capable of providing personalized and interactive educational support. Various aspects of AI tutoring, including intelligent tutoring systems[11], adaptive learning[13], or natural language processing (NLP)[15], have positively impacted student learning, performance, and engagement.

One of the most recent promising automatic approaches is "ChatGPT", a recent model trained by OpenAI that can interact conversationally. This model is trained to follow instructions in a prompt to provide appropriate responses in the context of a dialogue. ChatGPT can help answer questions, suggest recipes, write lyrics in a certain style, generate code, and much more. One can think, ChatGPT could easily generalize to tutor student engineers. Unfortunately, ChatGPT is not specifically designed as a demonstration tool. While it can provide examples and explanations, its responses are generated based on patterns and correlations in the training data rather than a deep understanding of the underlying concepts. Therefore, the accuracy and reliability of demonstrations by ChatGPT may vary depending on the user prompt [2, 8, 17] or the academic field [6]. Thus, it cannot be used as a trustworthy AI tutor for engineering students.

In this work, we introduce an AI tutor tailored for technical school courses (explicitly focusing on EPFL). When given a question from an exercise, the tutor must provide an in-depth explanation to the student in order to help them understand how to solve the exercise. We used a training methodology similar to modern AI chatbots such as ChatGPT. Specifically, our work is three-folded.

At first, we collect supervised fine-tuning data by distilling technical question demonstrations via ChatGPT using prompt engineering techniques. The *step-by-step* prompting strategy described in [8] is generally the most convincing and is found to be effective to provide a meaningful explanation on how to answer the question. We also show ChatGPT is often wrong in the details, so we get the help of a human-written cheat sheet that provides solutions to those technical questions to guide ChatGPT's answer.

Secondly, tutors trained with supervised demonstrations are further trained with reinforcement learning with human feedback (RLHF) to learn more from their own generated demonstrations. For RLHF to work, we train a reward model that can effectively reward the demonstrations that our chatbot produces.

Finally, we use our supervised demonstrations to train our final chatbot, which can be a practical educational assistant. We assess the quality of our demonstrations with standard metrics and our trained reward model.

As a result, we demonstrate that our AI tutor is more accurate than basic ChatGPT but less accurate than a well-prompted ChatGPT. Thorough evaluation support our claims both quantitatively and qualitatively. The RLHF training is left as future work.

2 Related work

In the field of education - and especially in the field of Intelligent Tutoring Systems - there exist many pre-existing works, starting as early as 1960 with the PLATO (Programmed Logic for Automatic Teaching Operations) project, and improving greatly to this day on. A common denominator between the previous research is that they explored various techniques for developing intelligent tutoring systems that can adapt to individual student needs to enhance learning outcomes. These intelligent tutoring systems were created from natural language processing models[1], speech models in neural networks[16], data analytic models[3], and more. Overall, prior research in the field of AI tutors has provided valuable insights into the design, development, and evaluation of intelligent tutoring systems that can improve the quality of education and enhance student learning experiences.

In this paper, we will use pre-existing and pre-trained NLP models such as ChatGPT-2.

3 Method

3.1 Dataset

3.1.1 Collecting demonstrations via ChatGPT

This report analyses various prompting strategies for collecting mathematical demonstrations of linear algebra using ChatGPT. In this work, we are granted access to 100 exercise questions from a French linear algebra class, including open and multiple-choice inquiries. An initial strategy would be to select a unique case study, select an optimal prompting strategy, and reiterate the prompt for every question. Unfortunately, an optimal prompting strategy for open questions is sub-optimal for multiple-choice questions and vice-versa. For this reason, we defined one prompting procedure per question type. Therefore, one case study was selected per question type to try and evaluate different prompts. Specifically, we defined four prompting strategies for each case study, from trivial ones to more complex ones. Then, we compared manually the quality of each demonstration and select the best one according to a confidence score set by us. Finally, we prompted ChatGPT on all the questions with the selected prompting strategies, which gave us a complete demonstration dataset. Please see the Milestone 1 for more implementation details.

3.1.2 Reward Model Dataset

In this work, we are given access to a large dataset of human-machine interactions in the scope of engineering classes. These interactions have been generated via ChatGPT by more than 100 NLP students on diverse subjects and questions, all related to Science and Engineering. Each question has three different answers, generated by three different human prompts. Each exchange has been labeled with a confidence score rating the quality of the answer provided by the ChatGPT from 1 (low quality) to 5 (high quality). It means that to train our reward model, we have access to pairable quality-labeled interactions. We won't preprocess further on this data for our dataset, but we still identify two problems:

- There is no ground-truth solution
- Some interactions have the same level of confidence for the same question, so the score cannot rank them

Concerning the first issue, the course's team provided us with a ground-truth solution dataset for the different questions. However, the given answers are often not good examples of qualitative explanations and educative developed answers. For example, MCQ's answers often give the answers without any explanation. To make use of these provided answers and have a bigger total dataset, we created the **Cheater** dataset. The dataset contains new high-quality interactions using ChatGPT, giving ChatGPT the solution and asking it to explain in detail why it is the answer.

For the second issue, every student has generated high-quality demonstrations in milestone 1. Consequently, two different interactions of the same question can have the same quality score. To solve this issue, we will artificially create a difference in the interaction qualities, by spawning a set of low-quality interactions using extremely basic prompting, and by so creating the dataset **Loser**. In the end, our training dataset for the reward model will contain high-quality (via the real solutions), mid-quality (via the student interactions), and low-quality (via the very basic prompting) interactions.

Loser: We first build a dataset with the collected data and artificial bad interactions. We simply group interactions per question id and get all possible pairs of interactions for the same question having distinct qualities. For each pair, the interaction with higher confidence is labeled as chosen while the other is labeled as rejected. This way, it generates more samples we can then label as positive. We can then recombine it with bad samples for the same questions that we generated in part 1 to have new chosen/rejected pairs.

Math: We added the MATH Dataset which contains problems from high-school math competitions varying in difficulty from 1 to 5. Its solutions are formatted just like we desire: A step by step reasoning explaining how to obtain the final answer (which is always mentioned and boxed). We first load the dataset and convert its labels to fit our dataset's format: the problem becomes a question and the solution is labeled as the answer. We take all these 'interactions' and label them as positive. To complete the MATH Dataset, we need to build the negative parts of each sample. To do so, we

Model / Dataset	MATH	Cheater	Loser
Reward Model	1'125	612	16'812
Generative Model	6'375	3'473	0

Table 1: Number of examples per dataset, per data type. The datasets include the training and testing datasets.

prompt ChatGPT for each question from the dataset. To be sure it gives "bad" answers, we use the following system prompt: You are a bad assistant who answers questions without explaining the reasoning behind them. Sometimes you replace numbers. We created multiple save checkpoints because of the size of the dataset, which we then merged together to obtain the negative part of the MATH dataset. Finally, for both parts of the MATH dataset (positive and negative) contained in different files in the same directory, we merge all of them with positive samples being labeled as chosen for a given question and negative samples with the same question being labeled as rejected.

3.2 Generative Model Dataset

For the generative model dataset, we needed pairs of input and label, with a scientific question as input, and a suitable, correct, and well-developed answer as label. For this reason, we left out entirely the Loser dataset, and only kept 85% of the Math and Cheater datasets, leaving the remaining 15% for the reward model dataset. In these data, we only kept the best answer (the *chosen* answer) each time as labels and discarded the *rejected* answer. We summarize our datasets in Table 1. This dataset is then split in to a training dataset (80%) and a testing dataset (20%).

3.3 Model

3.3.1 Reward Model

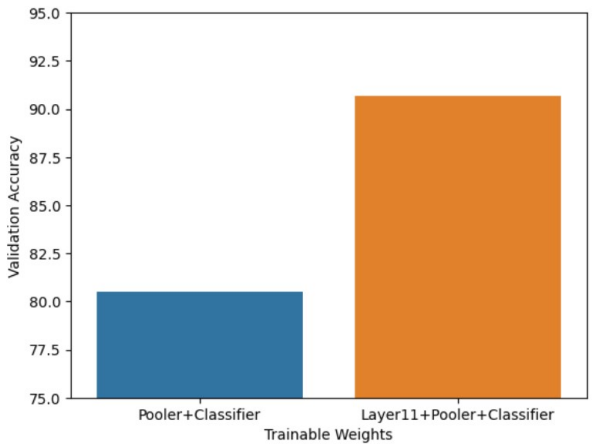

Figure 1: The difference of accuracy on the validation dataset with and without the update of attention layer 11

To implement the reward model, we decided to fine-tune DeBERTa-v2 [7] (Decoding-enhanced BERT with Disentangled Attention) for Sequence Classification. It has all the advantages of the BERT [4] and RoBERTa [10] models but improves on them with two significant refinements. The first one is a disentangled attention mechanism, where each word is represented using two vectors that encode its content and position, respectively, and the attention weights among words are computed using disentangled matrices on their contents and relative positions. The second one is an enhanced mask decoder, used to replace the output softmax layer to predict the masked tokens for model pretraining. These two technics have proved to significantly improve the efficiency of model pretraining and moreover increase the performance of downstream tasks.

To do the finetuning of the DeBERTa-v2 model, as we couldn't finetune all the weights due to our VRAM limit, we decided to implement a pooler with a classifier and we added more capacity by updating attention layer 11. As we can observe from Figure1, the update of attention layer 11 improves a great deal the model's accuracy. The data used for the training of the reward model is the reward model dataset, created as described above, and split into a training set (80%) and a validation set (20%). We chose a maximum sequence length of 4096 tokens, as it fits well on the GPU used for the training (an NVIDIA RTX 3090 of 24GB).

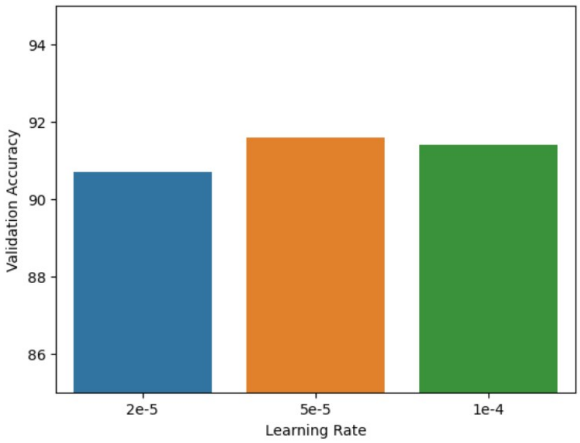

Figure 2: The accuracy of the model on the validation dataset with different learning rates

To choose the hyperparameters, we played with different values of learning rates (2e-5, 5e-5, and 1e-4) as you can observe from Figure2, and selected the best one. As for the reward loss, we went with the one suggested in InstructGPT. Next, for the optimizer, we used the AdamW, a cosine learning rate scheduler for 20 epochs, and a batch size of 2, which was the maximum achievable by our GPU. The regularization with gradient clipping was set to 3 and finally, we used early stopping on validation loss with patience set to 5.

3.3.2 Generative Model

For the generative model, we fine-tune GPT-2 [14] weights on our training generative dataset. It's a causal (unidirectional) transformer pre-trained using language modeling on a very large corpus of 40 GB of text data. GPT-2 was trained with a causal language modeling (CLM) objective and is therefore powerful at predicting the next token in a sequence. For this reason, we set the input of our dataset to be of the form "<question> <eos> <answer> <eos>" where we force the model to ignore loss for the question part in order to learn at predicting the next tokens for the answer part. To fit the input of GPT-2, we truncate the inputs to a maximum of 1024 tokens. On the other hand, if the number of tokens is below 1024, we fill the leftovers with pad tokens. We train the model for 100 epochs with AdamW optimizer, a learning rate of 1e-5, and a batch size equal to 4. We apply early stopping with patience five based on loss on the generative model validation set.

3.4 Evaluation

To evaluate the quality of our generative model, we use both standard metrics and our reward model. Upon generation of predictions from our test dataset both by our generative model and by ChatGPT, using primary and more advanced prompting, we compute BLEU [12], ROUGE [9] and score from our reward model.

Prompting: To generate predictions from ChatGPT, we use two types of prompting. For the first type of prediction, ChatGPT is prompted with the question from the dataset without any additional directions as to how it should answer it. The second type of prediction is generated from the prompt "$<$ question $>$ Let's think step by step..." to emphasize that the reasoning should be explained and reduce computation mistakes.

The dataset used for evaluation contains 1970 samples over which predictions from all three types of strategies (GPT Basic, Advanced, and generations from our model) are generated. The reward score is computed as the average of the scores over all the samples.

4 Results

The results of the evaluation are presented in Table 2. It shows that our model has poor quantitative scores, considering BLEU and ROUGE metrics, especially compared to the ChatGPT variants. On the other hand, considering the evaluation from the reward model, it performs better than the Basic version of the ChatGPT model. We can draw a number of interpretations from these results.

First, the Advanced version of ChatGPT has ROUGE and BLEU results comparable to the basic version. An examination of their predictions and their brevity penalty scores gives us some insights as to why this might be the case. We can observe that the predictions of both the Advanced and Basic models make mention of similar words and notions which are key to the question's answer. However, because of the way the Advanced model is prompted, it produces lengthier explanations, where it gives more details about the steps needed to derive the desired result. These additional information, although useful to a student, are not valued by the ROUGE and BLEU score which only measure recall of specific n-grams (which, in our situations, are these "key word").

Concerning our model, we can notice that its reward score is closer to the Advanced ChatGPT variant. This can be seen as its generation being produced with a format similar to what we might want from a chatbot that should help students with their questions. Yet, its ROUGE and BLEU score are lower than both of the ChatGPT versions. It seems to have trouble identifying information relevant to the answer to the student's question.

Table 2: Results of the evaluation over the three models

Models	ROUGE			BLEU		Reward Model's Score
	ROUGE 1	ROUGE 2	ROUGE L	BLEU	Brevity Penalty	
Our Model	0.24	0.04	0.12	0.04	1.0	1.7
ChatGPT Basic	0.35	0.18	0.26	0.20	0.87	-0.1
ChatGPT Advanced	0.38	0.18	0.26	0.20	1.0	2.4

5 Contributions

Collecting demonstrations via ChatGPT: Juliette Parchet, Guillaume Vray and Léo Wolff.

Reward Model:

MATH dataset for reward model: Léo Wolff

Cheater dataset for reward model: Léo Wolff & Juliette Parchet

Loser dataset for reward model: Guillaume Vray

Reward model training: Juliette Parchet & Guillaume Vray

Generative Model:

Dataset: Juliette Parchet

Training: Guillaume Vray

ChatGPT Baselines: Léo Wolff

Evaluation: Léo Wolff

Final Report:

Abstract & Introduction: Juliette Parchet & Guillaume Vray

Related Work: Juliette Parchet

Method: Juliette Parchet, Léo Wolff and Guillaume Vray

Experiments & Results: Léo Wolff

6 Conclusion

Our study focused on training a chat bot using machine learning techniques and evaluating its performance compared to the baseline model, ChatGPT. We found that the trained tutor bot outperformed ChatGPT when the latter was provided with trivial prompts, demonstrating its ability to generate responses providing more insights to a student.

However, the trained Chat bot fell short of the performance achieved by well-prompted ChatGPT. This limitation highlighted the importance of incorporating reinforcement learning with human feedback to improve the formatting of the bot's answers. By leveraging the reward model we developed, we could have guided the bot towards generating responses that align more closely with desired formats, enhancing its overall effectiveness as a personal tutor for EPFL students.

Furthermore, we identified some incoherence in the chat bot's answers, which makes it important for students using this technology to be attentive to the model's output. Addressing these incoherences represents a significant challenge, as they may require a deeper understanding of mathematical concepts, which still remains a complex task for recent models. Nevertheless, with further research and refinement, it is possible to enhance the bot's coherence and correctness.

Lastly, we recognize that exploring alternative underlying models is a promising avenue to improve the correctness of the Chat bot's outputs. By investigating different architectures or incorporating additional training data, we can enhance the bot's ability to generate accurate and reliable responses, which is crucial for its practical applicability as an educational tutor. A first path to explore could be the use of GPT3 for our base model.

Overall, our findings shed light on the strengths and limitations of the trained Chat bot, highlighting the potential for reinforcement learning with human feedback and model adjustments to further enhance its performance. As technology continues to advance, addressing these challenges will contribute to the development of more sophisticated and reliable conversational agents, fostering their adoption in education and being conducive to an education fitted to each individual.

References

[1] Shazia Afzal, Tejas Dhamecha, Nirmal Mukhi, Renuka Sindhgatta Rajan, Smit Marvaniya, Matthew Ventura, and Jessica Yarbro. Development and deployment of a large-scale dialog-based intelligent tutoring system. In M Morales, editor, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Industry Papers)*, pages 114–121. Association for Computational Linguistics, 2019.

[2] Stephen H Bach, Victor Sanh, Zheng-Xin Yong, Albert Webson, Colin Raffel, Nihal V Nayak, Abheesht Sharma, Taewoon Kim, M Saiful Bari, Thibault Fevry, et al. Promptsource: An integrated development environment and repository for natural language prompts. *arXiv preprint arXiv:2202.01279*, 2022.

[3] Maiga Chang, Giuseppe D'Aniello, Matteo Gaeta, Francesco Orciuoli, Demetrios Sampson, and Carmine Simonelli. Building ontology-driven tutoring models for intelligent tutoring systems using data mining. *IEEE Access*, 8:48151–48162, 2020.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[5] Associate Vice Presidency for Student Affairs and led by Prof. Kathryn Hess Bellwald Outreach. Mental health & well-being survey. *Task Force Mental Health & Well-Being*, 2022.

[6] Simon Frieder, Luca Pinchetti, Ryan-Rhys Griffiths, Tommaso Salvatori, Thomas Lukasiewicz, Philipp Christian Petersen, Alexis Chevalier, and Julius Berner. Mathematical capabilities of chatgpt. *arXiv preprint arXiv:2301.13867*, 2023.

[7] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654*, 2020.

[8] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *arXiv preprint arXiv:2205.11916*, 2022.

[9] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Annual Meeting of the Association for Computational Linguistics*, 2004.

[10] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[11] Elham Mousavinasab, Nahid Zarifsanaiey, Sharareh R. Niakan Kalhori, Mahnaz Rakhshan, Leila Keikha, and Marjan Ghazi Saeedi. Intelligent tutoring systems: a systematic review of characteristics, applications, and evaluation methods. *Interactive Learning Environments*, 29(1):142–163, 2021.

[12] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, page 311–318, USA, 2002. Association for Computational Linguistics.

[13] Vincenzo Piuri Monica Bianchini Rabindra Nath Shaw, Sanjoy Das. Employing adaptive learning and intelligent tutoring robots for virtual classrooms and smart campuses: Reforming education in the age of artificial intelligence. *Advanced Computing and Intelligent Technologies, Volume 914*, 2022.

[14] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[15] Thanveer Shaik, Xiaohui Tao, Yan Li, Christopher Dann, Jacquie McDonald, Petrea Redmond, and Linda Galligan. A review of the trends and challenges in adopting natural language processing methods for education feedback analysis. *IEEE Access*, 10:56720–56739, 2022.

[16] R. Venkatesh, Naganathan Ealai Rengasari, and N. Maheswari. Intelligent tutoring system using hybrid expert system with speech model in neural networks. *International Journal of Computer Theory and Engineering*, pages 12–16, 01 2010.

[17] Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Anjana Arunkumar, Arjun Ashok, Arut Selvan Dhanasekaran, Atharva Naik, David Stap, et al. Benchmarking generalization via in-context instructions on 1,600+ language tasks. *arXiv preprint arXiv:2204.07705*, 2022.