Fine-tuning T5 for question-answering task on EPFL courses data: an attempt at replacing teaching assistants

Delabarre Luca, Faure Antonin, Nemo Fabrice

***Abstract*—The task of providing help to students is hard. Students often need lengthy interactions with other humans to understand a problem step by step. Our aim is to train a large language model (LLM) on various subjects, specifically undergraduate courses at EPFL, such that the student can get detailed and quick answers without human contact. Students can then get help faster and assistants can focus on other tasks, such as more complex questions that the model cannot answer. Our results show that getting a LLM to answer course questions is possible but remains a complex task, for which performance greatly depends on the quality of the datasets used and nature of the questions.**

I. Introduction

Providing help to students is a challenging and costly task. It often requires direct, one-to-one human interaction between a teaching assistant and a student. Moreover, the interaction may take time and require significant effort and knowledge from the assistant to be able to grasp the understanding of the student and provide relevant help. Our project aims at training a large language model such that it can take the role of an assistant. This would eliminate the overhead of complexity, length and cost associated with traditional student-assistant interactions and greatly improve the ability of students to learn on their own. Work towards creating personalized assistants already exists but these assistants are often bad at mathematical reasoning and logic. By fine-tuning a large language model with data coming directly from EPFL courses, we are hoping to lay the building blocks of a new way of learning and getting help for students.

II. Related work

This task is not specifically new, there are other examples in the literature of LLM fine-tuning aimed at providing better results for a specific task. Our work is new in that it focuses specifically on Bachelor courses taught at EPFL. The train dataset is new yet the aim is similar to previous works which is why we relied on several papers to get inspiration on how to fine-tune our model with our data.

We used a higher-level language model (GPT-4 from ChatGPT) to generate data, namely the text of an "interaction" between a student (who asks a question) and an instructor (who answers it giving explanations). We got inspiration from prompt patterns[6] to write prompts that allowed us to use ChatGPT for generating proper "interactions" that were then used as examples of the expected behavior of our model.

InstructGPT[3] is a language model based on GPT-3 that has been trained specifically for following instructions from the user. Its training has been done with Reinforcement Learning from Human Feedback (RLHF), and using cross-entropy loss, that we have reused for training our model.

Vicuna[1] is an open-source language model that has been fine-tuned with prompts from a higher-level language model (ChatGPT), this is similar to our approach. Vicuna is different from our model as it is meant to be generalist, whereas ours aims to be specialized for Bachelor courses from EPFL. The fine-tuning of Vicuna relied on user-shared conversations from ShareGPT.com, but we could not rely on it as its API was no longer accessible by the time we had to train our own model, and the API is not convenient for gathering interactions that fit the content of Bachelor courses from EPFL. The paper detailed a qualitative evaluation method that relies on a higher-level LLM (GPT-4), we got inspiration from their method to evaluate our own model. The related blog article doesn't provide a proper way to evaluate quantitatively the outputs of the model.

We also got inspiration from Alpaca[5] which is an instruction-following language model fine-tuned from Meta's LLaMA model. We used their data to teach our model to follow instructions.

III. Approach

We aimed to combine the robustness of a large language model with sophisticated fine-tuning techniques, utilizing both supervised learning and reinforcement learning. Our process, as shown on Figure 1, involves an initial phase of supervised fine-tuning of the T5 model, followed by a reinforcement learning phase using either a Proximal Policy Optimization (PPO) algorithm or a N sampling method .

A) Model Architecture

We started from a well known model T5[4] (Text-to-text Transfer Transformers) that has proved its ability on various tasks such as question answering, summarization, and even

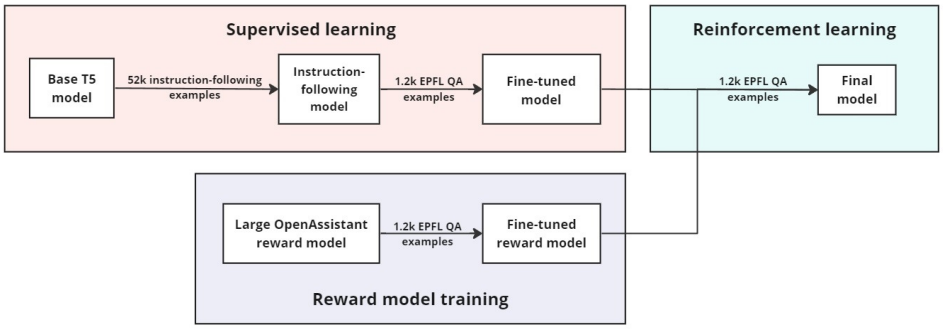

Fig. 1: Training Pipeline

classification when translated in text format. Our idea was to explore its ability to answer questions and extend it to various questions from EPFL courses. We tried two variations of T5, the base version and another publicly available version that was trained on question answering tasks from Huggingface that we refer to as the valhalla T5 model.

B) Supervised Fine-tuning

Following the selection of T5 as our base model, we used supervised fine-tuning to adapt it to the specific context of EPFL undergraduate courses. We first used an external dataset of instruction-following examples. Then, we further fine-tuned the model with a dataset of questions and demonstrations answers from these courses.

C) Reward Model

The reward model serves a pivotal role in evaluating and providing feedback on the quality of the responses generated by our T5 model. It is trained on a dataset of chosen-rejected demonstrations pairs for questions from EPFL courses, enabling it to distinguish between high-quality and low-quality answers.

D) Reinforcement Learning

After the initial fine-tuning phase, we experimented with multiple reinforcement learning approaches to refine the performance of our model, all aiming at enhancing the model's ability to generate high-quality responses.

1) Proximal Policy Optimization (PPO)

First, we tested the Proximal Policy Optimization (PPO) algorithm, a policy optimization method that refines the quality of the model's responses by maximizing the reward defined by our reward model.

During each PPO iteration, we used a decreasing temperature meaning that as the PPO iterations progressed for each question, the model was increasingly encouraged to exploit what it had already learned about the task, rather than exploring new solutions. This technique helped ensure the convergence of our model towards an optimal policy, enhancing its ability to provide high-quality responses to complex queries.

Algorithm 1 Proximal Policy Optimization

for $e \leftarrow 1$ to $epochs$ **do**
  for each $question$ in $dataloader$ **do**
    $action_{old} \leftarrow$ get_action($question$)
    $reward_{old} \leftarrow$ get_reward($action_{old}$)
    for $i \leftarrow 1$ to ppo_epochs **do**
      $temperature \leftarrow \max(0.1, 1 - 0.1 \times ppo\_epochs)$
      $action \leftarrow$ get_action($question$)
      $reward \leftarrow$ get_reward($action$)
      $loss \leftarrow$ get_loss($reward$, $reward_{old}$)
      $optimizer$.zero_grad()
      $loss$.backward()
      $optimizer$.step()
    end for
  end for
end for

2) N Sampling

To ensure the optimal selection of responses, we utilized a "N sampling" approach. In this technique, the model generates a fixed number (N) of responses and applies backpropagation for each response's reward given by the reward model. This process effectively guides the model towards generating higher-quality answers by learning directly from the diverse set of generated responses. By broadening the range of potential responses and reinforcing learning from high-reward answers, we mitigate the risk of the model producing suboptimal answers, thus improving the overall quality and diversity of the model's responses.

3) Comparing Actions with Ground Truth

Another potential approach we considered is directly comparing the reward of the model's generated actions with the reward of the true solution. This approach could enhance the learning process by providing a direct comparison between the model's responses and the ideal answer, thereby refining the model's policy to generate higher-quality responses.

In this proposed method, we calculate the reward for the model's generated action and for the ground truth action. We

Algorithm 2 N Sampling

```
for e ← 1 to epochs do
    for each question in dataloader do
        actions ← get_actions(question, n)
        rewards ← get_rewards(actions)
        for each reward in rewards do
            loss ← −reward
            optimizer.zero_grad()
            loss.backward()
            optimizer.step()
        end for
    end for
end for
```

then compute a loss function, such as cross-entropy or hinge loss, between these two rewards. The model is updated by back-propagation through this loss, encouraging it to generate actions closer to the ground truth.

While this method presents a clear and direct learning goal, it also has potential drawbacks. The model may overfit to the training data, limiting its ability to generalize to unseen queries. Additionally, the method assumes that the ground truth action is always the best possible action, which may not be the case due to the inherent subjectivity and complexity of conversational tasks.

Algorithm 3 Comparing Actions with Ground Truth

```
for e ← 1 to epochs do
    for each (question, ground_truth) in dataloader do
        action ← get_action(question)
        reward_action ← get_reward(action)
        reward_truth ← get_reward(ground_truth)
        loss ← get_loss(reward_action, reward_truth)
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
    end for
end for
```

IV. Experiments

A) Data

1) Reward model

- m2_reward_dataset_chaipas_basic_math: This dataset contains basic math demonstrations. It is a collection of dictionaries with keys *entry_id*, *chosen* and *rejected*. The chosen string is a valid interaction consisting of a math question and an answer generated by GPT-4. GPT-4 has been prompted with questions generated randomly for 4 basic math tasks: integer multiplication, addition and multiplication of fractions, and matrix multiplication. The questions consisted of a computation, 3 or 4 proposed answers, one being true and the others false. The prompt also included a description of the method to get the result. GPT-4 has been asked to generate an answer with a detailed explanation. The rejected string is a worse interaction where the answer and explanation have been altered to be wrong, by replacing all occurrences of the true answer in the text generated by GPT-4 by a randomly selected wrong answer among the proposed answers.
- m2_reward_dataset_chaipas_better_demonstrations: This dataset contains interactions, either in French or English, with a human and an assistant who answers. It is a collection of dictionaries with keys *entry_id*, *chosen* and *rejected*. The rejected string is the original interaction coming from students prompting ChatGPT (GPT-4). The chosen string is also an interaction with ChatGPT (GPT-4) but it has been leveraged in the sense that when prompting the model with the question, we provide it with the answer and the explanation, and ask it to explain the answer using the explanation. We then use this answer along with the original question as the chosen interaction.

2) Final model

- m3_gen_dataset_chaipas_basic_math: This dataset was derived from its m2 origin, and consists of the basic math demonstrations where we kept the "chosen" key.
- m3_gen_dataset_chaipas_better_demonstrations: This dataset was derived from its m2 origin, and consists of the basic math demonstrations where we kept the "chosen" key.
- m3_gen_dataset_chaipas_alpaca: This dataset comes from the data used for the training of Alpaca [5]. It consists of question-answer pairs and was used to align models to follow instructions when prompted with a question/task.

All of the above are collections of dictionaries with *question* and *answer* keys.

B) Evaluation method

We used several metrics to evaluate the performance of our models.

We took inspiration from the paper on Vicuna[1] to use GPT-4 as a way to evaluate qualitatively the answers of our models. The paper mentioned two prompts for evaluating answers, one is meant to evaluate an answer from the model on quality criteria, the other is meant to compare two answers to the same question. We adapted them as follows:

Prompt for evaluating an answer on quality criteria:

[System]
Please act as an impartial judge and evaluate the quality of the response provided by an AI assistant to the user question displayed below. Your evaluation should consider factors such as the helpfulness, relevance, accuracy, and level of detail of the response. Begin your evaluation by providing a short explanation. Be as objective as possible. After providing your explanation, please rate the response on a scale of 1 to 10 by strictly following this format: "[[rating]]".

[Question]
{question}
[Golden Answer]
{golden_answer}
[The Start of Assistant's Answer]
{answer}
[The End of Assistant's Answer]

Prompt for comparing two answers:

[System]
Please act as an impartial judge and evaluate the quality of the responses provided by two AI assistants to the user question displayed below, with the golden answer provided. You should choose the assistant that follows the user's instructions and answers the user's question better. Your evaluation should consider factors such as the helpfulness, relevance, accuracy, and level of detail of their responses. Begin your evaluation by comparing the two responses and provide a short explanation. Avoid any positional biases and ensure that the order in which the responses were presented does not influence your decision. Do not allow the length of the responses to influence your evaluation. Do not favor certain names of the assistants. Be as objective as possible. After providing your explanation, output your final verdict by strictly following this format: "[[A]]" if assistant A is better, "[[B]]" if assistant B is better, and "[[C]]" for a tie.
[User question]
{question}
[Golden Answer]
answer

[The Start of Assistant A's Answer]
$\{answer_a\}$
[The End of Assistant A's Answer]
[The Start of Assistant B's Answer]
$\{answer_b\}$
[The End of Assistant B's Answer]

The first prompt has been used to evaluate the results of our models on the 100 prompts that were provided, with a grade on a scale from 1 to 10, determined by GPT-4. Compared to the prompts from the paper about Vicuna, we decided to include the "golden" answer that was provided with the questions in `prompts.json`, with the hope that it can help the model figure out whether the generated answer is factually correct or not.

The second prompt has been used to compare our models with three baseline models: T5, GPT-2, and GPT-4. We computed the share of answers that were deemed better than the baseline by GPT-4, to check whether our training has been helpful at specializing our models to the task we want.

In addition to a qualitative evaluation done with GPT-4, we performed a quantitative evaluation with commonly used similarity scores: BLEU, ROUGE, and BERTScore. We computed them for each model and question, with the "golden" answer, provided in `prompts.json`, as reference and the output of the model as hypothesis.

Finally, we also manually assessed the quality of some of the answers.

C) Baselines

We used a couple of different baselines to compare against our model's performance. The first was the T5 model before any fine-tuning or reinforcement learning, providing a measure of the 'raw' model's capabilities. The second was the performance of the model after supervised fine-tuning but before reinforcement learning, providing a basis for understanding the incremental improvements brought about by our RL strategies. We also compared the results to GPT-2 and GPT-4 models results.

D) Experimental details

1) Reward model

Our experiments involved testing a variety of configurations to optimize the performance of our model. We experimented with different reward models, namely "OpenAssistant/reward-model-deberta-v3-base", "OpenAssistant/reward-model-deberta-v3-large", and "microsoft-deberta-v3-xsmall".

These models were incorporated with two types of mapping layers: a Linear mapping and a Multi-Layer Perceptron (MLP) mapping, as shown on Figure 2, with a hidden layer (output size $\rightarrow$ 64), ReLU activation, and a final layer (64 $\rightarrow$ 1) followed by a sigmoid activation.

Each of the models were trained using these different loss functions:

- Hinge loss:

$$\max(0, margin - reward_{chosen} + reward_{rejected}) \quad (1)$$

- Binary Cross-Entropy loss:

$$-\log(\sigma(reward_{chosen} - reward_{rejected})) \quad (2)$$

The training was done using Adam optimizer with learning rate of 1e-4, in one epoch with a batch size of 4 and a 90% training split.

In terms of model size, the "OpenAssistant/reward-model-deberta-v3-large" model was particularly large and resource-intensive, which resulted in slower training times, while on the other hand the "microsoft-deberta-v3-xsmall" model, with its smaller size, showed significant improvements in training time.

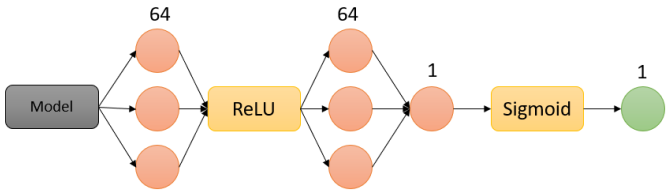

Fig. 2: Reward model with MLP reward head architecture

2) Model fine-tuning

We fine-tuned the model using a 90-10 train-test split with an AdamW optimizer. The optimizer was kept to default settings that is $learningrate = 10^{-4}$.

3) Reinforcement learning

For the reinforcement learning phase, we tested the use of PPO algorithm with 1 epoch and 3 PPO epochs with a batch size of 6. Our optimization scheme was based on the Adam optimizer with a learning rate of 1e-4, and we trained on 90% of *"m3_gen_dataset_chaipas_better_demonstrations"*. We also experimented with the "N" sampling strategy using a parameter of n=3, replacing the PPO epochs in this context.

E) Results

1) Reward Model

We report the performance of our various attempts of creating our reward model in Table I. For our final model, we used the large open-assistant model with a final MLP layer, trained on demonstrations with a hinge loss.

2) Fine-tuned models

Here are our results for the evaluation methods stated above.

Average grade (on a 1-10 scale) according to GPT-4	
Model name	Grade
Baseline GPT-2	1.708
Baseline T5	3.667
Baseline GPT-4	7.385
valhalla_T5_demonstrations	2.146
valhalla_T5_demonstrations_math	1.635
valhalla_T5_alpaca	1.667
valhalla_T5_alpaca_demonstrations	2.031
T5_demonstrations	2.063
T5_demonstrations_math	1.667
T5_alpaca	1.573
T5_alpaca_demonstrations	1.823
T5_alpaca_demonstrations_NSampling_large_hinge	2.052
T5_alpaca_demonstrations_PPO_large_hinge	2.010

Model name	Grade
Answers that were better than baseline T5	
valhalla_T5_demonstrations	51.0%
valhalla_T5_demonstrations_math	33.3%
valhalla_T5_alpaca	25.0%
valhalla_T5_alpaca_demonstrations	40.6%
T5_demonstrations	51.0%
T5_demonstrations_math	41.7%
T5_alpaca	21.9%
T5_alpaca_demonstrations	38.5%
T5_alpaca_demonstrations_NSampling_large_hinge	33.3%
T5_alpaca_demonstrations_PPO_large_hinge	30.2%
Answers that were better than baseline GPT-2	
valhalla_T5_demonstrations	58.3%
valhalla_T5_demonstrations_math	56.3%
valhalla_T5_alpaca	47.9%
valhalla_T5_alpaca_demonstrations	50.0%
T5_demonstrations	55.2%
T5_demonstrations_math	57.3%
T5_alpaca	43.8%
T5_alpaca_demonstrations	49.0%
T5_alpaca_demonstrations_NSampling_large_hinge	49.0%
T5_alpaca_demonstrations_PPO_large_hinge	47.9%
Answers that were better than baseline GPT-4	
valhalla_T5_demonstrations	9.3%
valhalla_T5_demonstrations_math	4.3%
valhalla_T5_alpaca	6.3%
valhalla_T5_alpaca_demonstrations	4.4%
T5_demonstrations	5.3%
T5_demonstrations_math	4.2%
T5_alpaca	3.1%
T5_alpaca_demonstrations	4.4%
T5_alpaca_demonstrations_NSampling_large_hinge	6.3%
T5_alpaca_demonstrations_PPO_large_hinge	6.3%

The various scores we computed are listed on Table II on Page 6.

V. Analysis

We can see that the model that performed the best according to GPT-4, given the prompt, is unsurprisingly baseline GPT-4. Then the second model that performed the best is baseline T5. This surprised us, when we manually read its outputs and compared it to the models we got fine-tuning it, it was clear that baseline T5 provides less convincing outputs, sometimes simply copying the question or some keywords. We think that ChatGPT (GPT-4) might be interpreting such answers as more qualitative than they really are.

Also, the results show that GPT-4 is not convinced by the outputs of our models, average grades are very low. This method has limits as it relies on an evaluation done by a LLM that still has flaws, we could have gotten better results with human evaluation.

The second evaluation method shows that our models are performing overall better than baseline GPT-2 and worse than baseline T5 and GPT-4 according to the evaluation done with GPT-4. We were not surprised by the results of the comparisons with GPT-2 and GPT-4, but the results of the comparison with baseline T5 again go against the impression we got reading manually the outputs. This evaluation method has the same flaws as the previous one.

Base Model	Reward Head	Training Data	Testing Data	Loss	Accuracy (Test)
OA Base	Linear	-	Demonstrations	-	7%
OA Base	MLP	-	Demonstrations	-	60%
OA Base	MLP	Demonstrations	Demonstrations	Cross Entropy	61.6%
OA Base	Linear	Demonstrations	Demonstrations	Cross Entropy	37.5%
XSmall	Linear	Demonstrations	Demonstrations	Cross Entropy	99.1%
OA Large	MLP	-	Demonstrations	-	54.2%
OA Large	MLP	Demonstrations	Demonstrations	Cross Entropy	0%
OA Large	MLP	Demonstrations	Demonstrations	Hinge	71.7%

TABLE I: Results for the reward models training

Model (training dataset) (RL) (Reward model) (Loss)	BLEU	ROUGE-1			ROUGE-2			BERT-SCORES		
		P	R	F	P	R	F	P	R	F
Baseline GPT-4	28.8%	35.0%	51.2%	38.1%	35.0%	51.2%	38.1%	0.499	0.629	0.552
Baseline GPT-2	1.0%	3.1%	23.3%	4.1%	3.1%	23.3%	4.1%	0.344	0.514	0.402
Baseline T5	1.5%	8.2%	7.3%	6.7%	8.2%	7.3%	6.7%	0.491	0.584	0.529
valhalla_T5_demonstrations	9.3%	4.7%	16.7%	6.5%	4.7%	16.7%	6.5%	0.372	0.560	0.437
valhalla_T5_demonstrations_math	8.1%	4.1%	16.0%	5.8%	4.1%	16.0%	5.8%	0.377	0.562	0.442
valhalla_T5_alpaca	1.0%	8.8%	7.3%	6.1%	8.8%	7.3%	6.1%	0.502	0.576	0.530
valhalla_T5_alpaca_demonstrations	0.9%	4.6%	16.7%	6.3%	4.6%	16.7%	6.3%	0.378	0.566	0.443
T5_demonstrations	1.0%	4.9%	18.9%	6.8%	4.9%	18.9%	6.8%	0.369	0.562	0.435
T5_demonstrations_math	0.9%	4.1%	18.9%	6.0%	4.1%	18.9%	6.0%	0.372	0.561	0.437
T5_alpaca	1.0%	7.1%	8.1%	6.1%	7.1%	8.1%	6.1%	0.479	0.570	0.511
T5_alpaca_demonstrations	0.9%	4.6%	16.0%	6.3%	4.6%	16.0%	6.3%	0.379	0.563	0.443
T5_alpaca_demonstrations_NSampling_large_hinge	1.0%	4.8%	15.7%	6.4%	4.8%	15.7%	6.4%	0.380	0.562	0.444
T5_alpaca_demonstrations_PPO_large_hinge	1.0%	4.8%	15.7%	6.4%	4.8%	15.7%	6.4%	0.380	0.562	0.444

TABLE II: Metrics for the different models trained

About the various metrics reported, we observe that: (1) Our models generally have poor overlap with the gold answers. This is partly due to gold answers having no explanation, whereas our models were trained to provide an explanation. BLEU score maxes-out at only about 10% for the valhalla T5 model fine-tuned on demonstrations. (2) ROUGE 1 and 2 scores also show the same trend, but we can still see a clear difference between the recall ROUGE-1 and ROUGE-2 score of models that were trained on demonstrations against model that were not. We observe a 10% increase in these metrics when the model was trained using the demonstrations we collected. (3) Finally, BERTScores are not very conclusive but still show that the best model might be valhalla T5 trained on the Alpaca dataset. We think that this model is the best one when it comes to question comprehension and producing an answer, but upon manual inspection of the answers, though better constructed, were of poor content quality.

For our final model, we used valhalla T5 demonstrations. Despite not using reinforcment learning, this pretrained model further fine-tuned on the demonstrations performed better across the board according to ChatGPT (GPT-4), and also according to our human evaluation.

A) Reward model

About the performance, we think that the model ”XSmall” with very high accuracies, as shown in Table I, is finding some kind of shortcut in our data such as the length of non-pad token or identifying some recurring token in the better/worse demonstrations. Thus, we rather used the OpenAssistant large model for the reinforcement learning phase as it came second with a more realistic accuracy. We also observed that the external data might not be a good fit for this task as its distribution might be too far from that of our demonstrations. During our experiments using the *basic_math* dataset, the reward models obtained almost always 100% accuracy on the test set. We think that this is due to the model using other features such as length of the answer to create a suitable reward. This is confirmed by the very low accuracy of the model on a different distribution of question-answer pairs. Given the low robustness of the models trained on this math dataset, we excluded it from our following experiments.

B) Reinforcement learning

We encountered numerous challenges while implementing reinforcement learning techniques to our model. The main challenge we faced is that it is hard to grasp what the ”true” answer is for a given question and which score one should assign to it. While training our reward model, reward values to different answers were generally very close. Another problem is that we used the same data used during fine-tuning, possibly over-fitting its distribution.

Overall, we are conscious that our results are less than ideal. Given that this is our first attempt at such a task and the short two weeks time-frame in which this project had to be finalized, we do think it still remains a solid attempt at solving such a complex challenge.

VI. Conclusion

We successfully implemented our own assistant using both fine-tuning and reinforcement learning. We have learned how difficult such a task can be and how computationally intensive reinforcement learning is. We have also observed that the quality of the reinforcement learning phase also greatly depends on the quality of the reward model itself.

Human evaluation clearly shows that our final model is an improvement over the original T5. There is still a lot of work to be done for the model to produce high quality answers. We do think the main limitations of our work reside in the quality and quantity of data that was collected, as we only used about 1.2k questions and answer pairs and these questions were not of the highest quality.

VII. Future work

Recently Microsoft published a new model Orca [2] where they develop three models: a teacher model, a student model and a data generator. This approach could be interesting to develop further for our case study.

Another path to explore could be the use Tree of Toughts (ToT)[7] reasoning, which significantly enhances language models' problem-solving abilities, for the training of our models

VIII. Team contributions

A) Milestone 1

This milestone was mostly individual. We schduled a meeting to share our ideas on the project plan and wrote it together.

B) Milestone 2

Luca:

- Collected the better demonstrations using GPT-4
- Created the first, second, third and fourth reward model notebook
- Found microsoft/deberta-v3-xsmall model
- Trained base model, GPT-2, xsmall models
- Found, downloaded and adapted external source of data
- Introduced cross entropy loss from Instruct-GPT
- Trained several models

Antonin:

- Introduced margin loss
- Trained final models

Fabrice:

- Created math demonstrations and prompted GPT-4

C) Milestone 3

Luca:

- Transform milestone 2 data to be used in milestone 3
- Setup model fine-tuning
- Found and adapted Alpaca dataset
- Fine-tuned T5 and valhalla/t5-base-e2e-qg in various configurations

Antonin:

- Trained new reward models with OpenAssistant large
- Introduced PPO and N sampling
- Trained the RL phase

Fabrice:

- Model evaluation through ROUGE, BLEU, BERTScores (quantitative evaluation)
- Answer quality evaluation via ChatGPT (GPT-4): research for prompt-engineering to do answer grading and comparison with baseline models

References

[1] Wei-Lin Chiang et al. *Vicuna: An Open-Source Chatbot Impressing GPT-4 with 90%* ChatGPT Quality*. Mar. 2023. URL: https://lmsys.org/blog/2023-03-30-vicuna/.

[2] Subhabrata Mukherjee et al. *Orca: Progressive Learning from Complex Explanation Traces of GPT-4*. 2023. arXiv: 2306.02707 [cs.CL].

[3] Long Ouyang et al. *Training language models to follow instructions with human feedback*. 2022. arXiv: 2203.02155 [cs.CL].

[4] Colin Raffel et al. *Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer*. 2020. arXiv: 1910.10683 [cs.LG].

[5] Rohan Taori et al. *Alpaca: A Strong, Replicable Instruction-Following Model*. 2022. URL: https://crfm.stanford.edu/2023/03/13/alpaca.html.

[6] Jules White et al. *A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT*. 2023. arXiv: 2302.11382 [cs.SE].

[7] Shunyu Yao et al. *Tree of Thoughts: Deliberate Problem Solving with Large Language Models*. 2023. arXiv: 2305.10601 [cs.CL].