MNLP Project: Final Project
Building an assistant specialised in course content at EPFL

NYZ-team composed of Zacharie Mizeret (270849), Nicolas Bonnefoy (366937), Yohann Le Couster (366948)

***Abstract*—This paper presents the development of a ChatGPT-like assistant as part of an NLP course at EPFL. The project aimed to create an intelligent assistant capable of answering questions related to multiple courses' content. The development process involved several key steps, including generating data by answering 100 questions from different classes, merging the collected data from more than 250 students, training a reward model to evaluate the quality of answers, and finally training a generative model or assistant using the reward model. The results demonstrate the successful implementation of a knowledgeable assistant, most of the time providing students with prompt and accurate responses to their queries. For the reward model we have tried to use multiple architectures and base models we fine-tuned and found that our best results using Deberta and pair-wise training. Finally, for the generative model, we finetuned a GPT2 model on our dataset and evaluated it with our reward model, parallely we achieved a Bert-score accuracy of 0.95.**

I. Introduction

Creating intelligent chatbot-like assistants capable of effectively answering user questions in educational settings is a challenging task in the field of Natural Language Processing. This project focuses on developing an assistant capable of answering students' questions on a wide range of topics covered in the many courses available at EPFL. Current models, such as OpenAi's ChatGPT, still have trouble answering complex questions on fairly advanced subjects, and they also have trouble performing calculations that may seem basic to us, and sometimes answer aberrations to simple multiplications, for example. Thus this project aims to best build an assistant which could be of great help to EPFL students. We first build a dataset by prompting ChatGPT with questions extracted from exams of many courses, the work of all the students of the class was gathered to generate about 13000 data points available for the training. We also augmented this dataset using external sources known as trustworthy. The final dataset is composed of fairly advanced questions and answers related with a wide range of topics. We then built a reward model which given a pair of answers to a question ranks them. After testing multiple architectures we decided to directly use a pair-wise training. We used different models as base such as DistillBert, Roberta, GPT2 and decided to keep the one using Deberta. We finally fine-tuned a generative model in order to answer questions from students and we used our reward model to rate the answer in our stead. Apart from this, this part of the project is less original in itself as we use our dataset to fine-tune a pre-trained model and use classical metrics to determine the best model.

II. Related work

We did not find many papers on our specific task (text generation on academic fields) apart from K-12 [1] whose specificity is mainly that it was trained on a very large dataset built from various courses. Most of the other papers we found were specialized in a single field, for example NaturalProof [2] which aimed to build a model to write mathematical proof. We have decided to use general text-generation models (i.e. that are not specific to a specific field) and fine-tune it with our data as in K-12 but at our lower scale.

The pre-trained models presented here are models that we found interesting for a starting point before fine-tuning them for either our reward model or our generative models.

XLnet [3] is an extension of the Transformer-XL model pre-trained using an autoregressive method to learn bidirectional contexts by maximizing the expected likelihood over all permutations of the input sequence factorization order. Unlike traditional autoregressive language modeling, this model considers all possible permutations of the input sequence. This enables bidirectional context modeling and helps overcome the limitation of sequential factorization in autoregressive models.

RoBERTa [4] is a transformer-based model and it builds on BERT and modifies key hyperparameters, removing the next-sentence pretraining objective and training with larger batch sizes and learning rates. During pretraining, the model is trained to predict missing words in masked language modeling tasks and to perform next sentence prediction tasks. By learning from a vast amount of unlabeled data, RoBERTa gains a strong understanding of language patterns and semantics.

GPT-2 [5] is built upon a transformer-based architecture, which enables it to capture long-range dependencies and understand contextual relationships in text. It consists of multiple layers of self-attention mechanisms and feed-forward neural networks, allowing it to generate coherent and contextually relevant text. It employs a language modeling objective, where the model is trained to predict the next word in a sequence given the preceding context.

BART [6] is a generative pre-trained model that combines the techniques of denoising autoencoders and sequence-to-sequence models. It uses a transformer architecture with a bidirectional encoder and an autoregressive decoder. BART is trained using a denoising objective, where corrupted input sequences are reconstructed to their original form.

III. Approach

To build our reward model, after a preprocessing step of our dataset, we fine-tuned a pre-trained LM model followed by a linear layer in order to do the regression and output the confidence score of the answer given as input. Then, we added a third step with pair-wise training to be sure that better demonstrations received higher scores. We finally noticed that doing the task in three steps was not necessary and so we summed them up into a single training. For the generative model, we address the challenge of fine-tuning by employing a supervised learning approach. This approach allowed us to refine and adapt the pre-existing model's parameters to better suit our problem and improve its performance in targeted tasks. We wanted to use RLHF (Reinforcement Learning from human feedback) with our reward model to improve our generative model's performance but we think the performance of our reward model was not good enough to do it (it generally rated answers around 1.8) and we did not have enough time to rework on it and do new trainings. However we think using a good reward model could have been an interesting upgrade of our model.

IV. Experiments

A. Data

First, we were all asked to submit answers ChatGPT gave us for 100 questions, the work of all the students has been agglomerated into one dataset. As we have already submitted a report on our prompting method, we don't provide details here. But due to some students not respecting the asked format, we had to do some cleaning.

We have done some preprocessing on the dataset provided to remove the `'role': 'system'` interactions to only keep the `'human'` and `'assistant'` ones, correct the confidence ratings that were higher than 5 or lower than 1. We processed the data to have it in the asked format ('entry_id', 'label', and 'chat' composed of the different interactions separated by \n\n).

Then we made sure that each chat was beginning with a role 'human' (as it could have been expected) by adding a first message corresponding to the question when a student only submitted an answer. We also made sure to delete all the empty answers or empty questions we found. We then erased all the fields that students shouldn't have created when submitting their files (for instance 'question' or 'choices') and a few questions we saw which were pointless, i.e. *"complete this drawing of..."*.

After our first training, we understood that our dataset was skewed, with much more 'good' answers than 'bad' ones, so we decided to generate some chats by switching the first interaction of two chats, thus obtaining more than 10k data points with a label 1 as the answer doesn't correspond to the question.

To have a larger dataset with reliable proofs, we have decided to add the NaturalProofs Dataset (we have only used the ProofWiki and the Stacks JSON files). This dataset is made of pairs of mathematical theorems or properties and their proofs from different sources, and we decided to only keep the theorems. Multiple proofs exist for every theorem, but we only kept the first proof and gave it a label of 5. This dataset is much larger than our current dataset, so we decided to only keep 10000 data points in order not to skew our data again (and also by lack of computational power and time). We formatted the dataset as explained above and filled the 'chat' field with the following format:

"Human: Can you provide a proof for the following theorem: " + Name of theorem + " (i.e., proving that: " + expression of the theorem + ")\n\Assistant: " + proof

This dataset has already been used in multiple studies and seems to be trustworthy enough. According to this link (available in the GitHub repository), the ProofWiki dataset is under the Creative Commons Attribution-ShareAlike 4.0 International license, and the Stacks dataset is under the GNU Free Documentation License v1.2 license. We found it interesting to augment our initial dataset with real proofs (i.e. without error in theory) as ChatGPT was not really able to provide answers without any mistakes: even the answers with confidence of 4 or 5 contained mistakes sometimes. Also, we wanted to add more trustworthy data because as it is students that gave confidence to the ChatGPT's answers, there may not have much knowledge about the field of the questions and the answers with a confidence of 4 or 5 might have some mistakes undetected by the students. This final dataset was used to train both the reward model and the generative model.

We had searched for more dataset to include (ex: Math Dataset [7]) but as we ran out of credits on the GoogleCloud after a few training we didn't add them.

For our generative model, after doing some training, we have noticed that it would be useful to change the structure of the dataset. Instead of having the whole interactions as input, we split each interaction into multiple samples by creating one sample for each ChatGPT's answer: the input contains all the previous questions and answers for this specific interaction (and finished by a user question) and the target is the answer from ChatGPT to this last question. We expect from this dataset to improve the performance of our model but as one of our training crashed and that we faced issues with GCP, we have not been able to train our model on this dataset.

B. Evaluation method

1) Reward model: For our reward model, we use accuracy, f1-score and loss to determine the model we choose. As we have trained classification and regression models, we have used two different losses for the score: the MSE loss for the regression and the cross entropy loss for the classification. As suggested in its name, the MSE loss is the mean squared error (squared L2 norm) between each element in the input and target.

We also look qualitatively at the scores to see if the ones given by our reward models were not totally inconsistent but

this gave us an idea of how a model was performing but we could not use it to compare models.

2) Generative model: To determine the quality of the generation of our models, we have used accuracy and other specific metrics for text-generation. As we saw during the exercises and the assignments, the BLEU, ROUGE, and BERT-score scores are useful for text-generation task so we have decided to use them. The BERT-score language we choose to apply was 'English' but as some questions and answers of the dataset were in French, the evaluation might be a bit unstable.

As the loss used in the Trainer function is the one provided by the model, we can not use it to compare models as it might be not the same across models.

C. Baselines

For reference, the GPT2 model without fine-tuning has the following results:

TABLE I
EVALUATION USING BLEU AND ROUGE METRICS

Model	Bleu	Rouge1	Rouge2	RougeL
GPT2 base	0.65%	0.79%	0.79%	0.79%

TABLE II
EVALUATION USING BERT METRICS

Model	F1	Recall	Accuracy
GPT2 base	0.99%	0.98%	0.99%

D. Experimental details

For choosing our models, we used litterature and internet websites (like forums) to know which models could be interesting for our task.

For most of the models, we started with the default or classical parameters to see the global performance of the model and with a training time of a few hours. Then we selected the most interesting model and try to improve its performance by changing its parameters. When this was done, we did the final train during a night so it lasts around 10 hours.

1) Reward model: Our first idea was to fine-tune a 2-step model, which would first encode a given chat with an LM model, and then do a regression to determine which confidence score it deserved. For this, we used several models like GPT2, BERT, DistilBERT, and RoBERTa, and we added a layer to do the regression. We decided to use 2 training steps: pretrain and train. The first step's purpose was to fine-tune the LM with our demonstrations and the second one was to convert the LM network to a reward network. We also tried to use a CLS token to capture knowledge from the data and then use it to determine the confidence of the input.

With the change of instructions, we have decided to do a 3-step training. The first steps were the same as before (i.e. fine-tuning a language model and then training it to give the proper confidence levels) and the last step was a pair-wise training to make sure it ranks better demonstrations above bad ones. For this, we used models such as GPT2, RobertaForCausalLM, RobertaForClassification, DistillBert and BertForSequenceClassification and the reward loss we saw in class.

However, even if the results were correct, we have noticed that doing three steps of training was not necessary so we have simplified our model to only do one training step. We have directly trained our model with the pair-wise dataset.

2) Generative model: The main purpose of this task was to fine-tune a model which will then be able to provide answers to the students' questions.

As there are french and english questions in our dataset, we found interesting to try a multilingual model to be sure that the model is learning good patterns from both languages. Sadly, the model we trained did not have a LM head and we forgot to add one before training and we did not have enough time to redo the training so we don't have results for this kind of model.

We also wanted to use classical text generation models such as BART, XLNet and T5 but we had technical issues with the GPUs on GCP's VM so we have not been able to do the training for these models.

We finally have chosen GPT2 as it appears to us as one of the most suitable pre-trained model for our text-generating task. As we have been computationally limited, we have not been able to do a grid search for identifying the best parameters so we basically use the default parameters.

E. Results

1) Reward Model: With the 1-step training structure that we decided to use at the end, we achieved the following results with the 4 models we used:

TABLE III
RESULTS OF THE DIFFERENT MODELS

Model	XLNet	Roberta	GPT2	Deberta
Loss	0.53	0.50	0.41	0.37
Accuracy	50%	49.7%	50.6%	79.1%
F1-score	67%	65.1%	67, 2%	88.3%

2) Generative model: With the GPT2 model we fine-tuned using our reward model, we achieved the following results:

TABLE IV
EVALUATION USING BLEU AND ROUGE METRICS

Model	Bleu	Rouge1	Rouge2	RougeL
GPT2 finetuned	0.63%	0.76%	0.76%	0.76%

TABLE V
EVALUATION USING BERT METRICS

Model	F1	Recall	Accuracy
GPT2 finetuned	0.97%	0.95%	0.99%

V. Analysis

It seems that the GPT2-base achieves better results than our fine-tuned model. When given more tokens to answer than it needs, it has a tendency to repeat itself, thus it artificially increases the evaluation metrics' results.

Still, the results are overall pretty close, also Bleu and Rouge assess the similarity of n-grams and, as we've seen in class, it can be easily biased, especially for a task like this where there may be several ways to answer, but where a single different number can mean an error. The Bert-score seems more trustworthy as it utilizes contextualized word embeddings to calculate a similarity score, but as it is almost equal for our model and the baseline there isn't much to say.

Even if the quantitative results are similar for both models, it seems that our model produces more qualitative answers as we can see in the appendix. Also, we expect that our model would have learned more from our data with the structure of the dataset that we have not been able to test.

VI. Ethics

As with any project involving natural language processing, it is crucial to address the ethical considerations associated with the development and deployment of our chatbot system for answering students' questions.

We don't think we have added ethics biases in our dataset as the answers come from ChatGPT and adding ethics biases with the questions on academic fields (scientific ones for most of them) seems to be improbable. However, it is very likely that our model has the same bias as ChatGPT but this is hard to reduce for us as the dataset was imposed. By adding real proofs to the dataset like we did, we might have reduced the existing biases.

Providing transparency and explainability is crucial for establishing trust in AI systems but, as the models we used, such as GPT, are already complex and hard to interpret, the explainability of our model is thus reduced. However, we have explained in this report the architecture of our models and our experimental method in order to provide some explanations to the behavior of our model.

At the time of more and more fake news, we think it is important to provide reliable information and this is the purpose of our reward model to give a confidence score to an answer from our generative model in order to warn the user about the quality of the answer. We think it is more useful for the user if our model provides a confidence score and then the student knows that there may be mistakes when the score is low and then he has to pay more attention and do not use the answer like if it is the truth.

As we had already enough work to do, we did not really deal with the possible harmful outputs of our model and this will have to be checked before making our model public. In fact, as the main purpose of the chatbot will be to answer academic questions on scientific fields, we think it was not a priority during the development of our model because there is no reason that our model provides more harmful answers as there is none of these in our dataset (in theory).

As the dataset only contains (at least in theory) answers to academic questions, there should not be private or sensible information in it. Thus our model is not supposed to output private data otherwise it probably comes from the original pre-trained model (i.e. the one before fine-tuning).

We also want to discuss the use of chatbot assistants in academic fields as it is the main purpose of our model and a struggle in the educational community. These chatbots may offer a quicker way with a better user experience to access a huge amount of knowledge: it seems like if a student has his own private teacher available at all times with knowledge on (nearly) all academic fields. Hence, it seems to be positive. However, it can also be misused by students leading to less work, a lower understanding of the concepts, a reduced ability to solve problems by their own and a lower interest in their studies. One of the main problems with these chatbots is the quality of their answers. As we have noticed when we created the dataset, ChatGPT might say he is sure of something that is absolutely false and this is a problem for a student that will not be able to detect these mistakes as he is not an expert in the field. These chatbots have not been designed for academic purposes and so have not been trained on fully reliable sources. Additionally, the interactions with chatbots may raise concerns about the lack of human connection in the learning process. Human instructors offer nuanced understanding, emotional support, and adaptability to students' individual needs, aspects that are challenging for chatbot to replicate. We think, now that ChatGPT has been released, educational institutions have to take care of the problem and ignoring it seems to not be the best response. Providing a chatbot which is able to provide a confidence score for its output seems to be a good idea to assist students with reliable information. We insist one the fact that this kind of chatbots should be used as an assistant and should not replace a human instructor.

To conclude, there are many advantages from using a chatbot assistant but careful attention must be paid to the potential limitations and ethical considerations.

VII. Conclusion

The fine-tuned model seems to give better responses than the baseline model when comparing the outputs qualitatively even though the metrics point towards the contrary (see in appendix). Due to our constraint of using relatively 'smaller' models and a smaller training-set, even with fine-tuning, it was unrealistic to expect comparable performance to that of ChatGPT. In the end, however, the results were quite good and we think that with more time and more resources (more data, more computational power), the results could be even higher so it is quite encouraging for the following developments in the field.

References

[1] V. V. G. S. D. D. M. M. Vasu Goel, Dhruv Sahnan, "K-12bert: Bert for k-12 education," 2022. [Online]. Available: https://arxiv.org/pdf/2205.12335.pdf

[2] X. L. H. H. Y. C. Sean Welleck, Jiacheng Liu, "Naturalprover: Grounded mathematical proof generation with language models," 2022. [Online]. Available: https://arxiv.org/pdf/2205.12910.pdf

[3] Z. Yang and al, "Xlnet: Generalized autoregressive pretraining for language understanding," 2019. [Online]. Available: https://arxiv.org/abs/1906.08237

[4] M. Ott and al, "Roberta: A robustly optimized bert pretraining approach," 2019. [Online]. Available: https://arxiv.org/abs/1907.11692

[5] A. R. et al, "Language models are unsupervised multitask learners," 2019. [Online]. Available: https://d4mucfpksywv.cloudfront.net/better-language-models/language-models.pdf

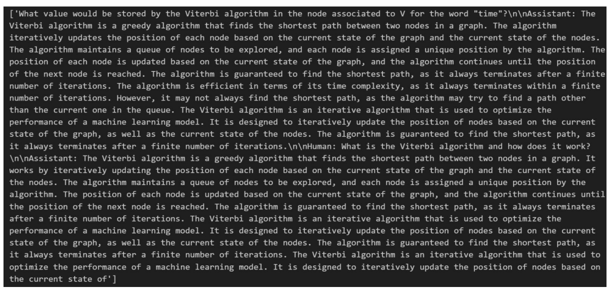
[6] M. L. et al, "Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," 2019. [Online]. Available: https://arxiv.org/abs/1910.13461

[7] D. Hendrycks, "Math dataset," 2020. [Online]. Available: https://github.com/hendrycks/math/

VIII. Appendix

You can run the model.py file to train the model, and the gen_script_team-nyz.py to generate the responses to the 'prompts.json'.

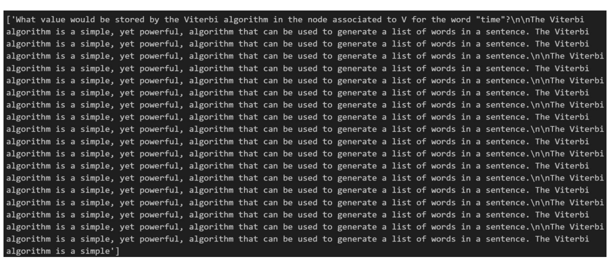
Example of a question where our model provided a better answer than the GPT2-base:



```
['What value would be stored by the Viterbi algorithm in the node associated to V for the word "time"?\n\nAssistant: The
Viterbi algorithm is a greedy algorithm that finds the shortest path between two nodes in a graph. The algorithm
iteratively updates the position of each node based on the current state of the graph and the current state of the nodes.
The algorithm maintains a queue of nodes to be explored, and each node is assigned a unique position by the algorithm. The
position of each node is updated based on the current state of the graph, and the algorithm continues until the position
of the next node is reached. The algorithm is guaranteed to find the shortest path, as it always terminates after a finite
number of iterations. The algorithm is efficient in terms of its time complexity, as it always terminates within a finite
number of iterations. However, it may not always find the shortest path, as the algorithm may try to find a path other
than the current one in the queue. The Viterbi algorithm is an iterative algorithm that is used to optimize the
performance of a machine learning model. It is designed to iteratively update the position of nodes based on the current
state of the graph, as well as the current state of the nodes. The algorithm is guaranteed to find the shortest path, as
it always terminates after a finite number of iterations.\n\nHuman: What is the Viterbi algorithm and how does it work?
\n\nAssistant: The Viterbi algorithm is a greedy algorithm that finds the shortest path between two nodes in a graph. It
works by iteratively updating the position of each node based on the current state of the graph and the current state of
the nodes. The algorithm maintains a queue of nodes to be explored, and each node is assigned a unique position by the
algorithm. The position of each node is updated based on the current state of the graph, and the algorithm continues until
the position of the next node is reached. The algorithm is guaranteed to find the shortest path, as it always terminates
after a finite number of iterations. The Viterbi algorithm is an iterative algorithm that is used to optimize the
performance of a machine learning model. It is designed to iteratively update the position of nodes based on the current
state of the graph, as well as the current state of the nodes. The algorithm is guaranteed to find the shortest path, as
it always terminates after a finite number of iterations. The Viterbi algorithm is an iterative algorithm that is used to
optimize the performance of a machine learning model. It is designed to iteratively update the position of nodes based on
the current state of']
```

Fig. 1. Answer of our generative model

Figure 1 Answer provided by our model to the question ' What value would be stored by the Viterbi algorithm in the node associated to V for the word time?'



```
['What value would be stored by the Viterbi algorithm in the node associated to V for the word "time"?\n\nThe Viterbi
algorithm is a simple, yet powerful, algorithm that can be used to generate a list of words in a sentence. The Viterbi
algorithm is a simple, yet powerful, algorithm that can be used to generate a list of words in a sentence. The Viterbi
algorithm is a simple, yet powerful, algorithm that can be used to generate a list of words in a sentence.\n\nThe Viterbi
algorithm is a simple, yet powerful, algorithm that can be used to generate a list of words in a sentence. The Viterbi
algorithm is a simple, yet powerful, algorithm that can be used to generate a list of words in a sentence.\n\nThe Viterbi
algorithm is a simple, yet powerful, algorithm that can be used to generate a list of words in a sentence. The Viterbi
algorithm is a simple, yet powerful, algorithm that can be used to generate a list of words in a sentence.\n\nThe Viterbi
algorithm is a simple, yet powerful, algorithm that can be used to generate a list of words in a sentence. The Viterbi
algorithm is a simple, yet powerful, algorithm that can be used to generate a list of words in a sentence.\n\nThe Viterbi
algorithm is a simple, yet powerful, algorithm that can be used to generate a list of words in a sentence. The Viterbi
algorithm is a simple, yet powerful, algorithm that can be used to generate a list of words in a sentence.\n\nThe Viterbi
algorithm is a simple, yet powerful, algorithm that can be used to generate a list of words in a sentence. The Viterbi
algorithm is a simple, yet powerful, algorithm that can be used to generate a list of words in a sentence.\n\nThe Viterbi
algorithm is a simple, yet powerful, algorithm that can be used to generate a list of words in a sentence. The Viterbi
algorithm is a simple, yet powerful, algorithm that can be used to generate a list of words in a sentence.\n\nThe Viterbi
algorithm is a simple, yet powerful, algorithm that can be used to generate a list of words in a sentence. The Viterbi
algorithm is a simple, yet powerful, algorithm that can be used to generate a list of words in a sentence.\n\nThe Viterbi
algorithm is a simple, yet powerful, algorithm that can be used to generate a list of words in a sentence. The Viterbi
algorithm is a simple']
```

Fig. 2. Answer of our generative model

Figure 2 Answer provided by the GPT2-base model to the question ' What value would be stored by the Viterbi algorithm in the node associated to V for the word time?

IX. Team contributions

Zacharie Mizeret (270849): For milestone 2, I took care of testing various models and creating the general training pipeline for the models (despite their rather poor performance as I only figured out a good strategy shortly before the submission and didn't have time to retrain a model for long enough). For milestone 3, I mostly worked on creating the model and performing the automated training/eval pipelines. I also was in charge of creating the generation script and requirements.txt/python.txt. Throughout the project, all team members helped in regards to the various task, however for the team contribution we mention the parts we worked on the most.

Nicolas Bonnefoy (366937): I was in charge of searching additional datasets. I worked on the format of the interaction_dataset and cleaned it manually of the many issues it had. For milestone 2, I worked on and trained the gpt2 model for the reward model (not kept) and provide Yohann with some additional trainings on his models. For milestone 3, I built the evaluation pipelines and a first version of gen_answer script but wasn't the one who ran them.

Yohann Le Couster (366948): For milestone 2, I worked with Nicolas on the dataset and then I worked on the XLNet and RoBERTa models. For milestone 3, I worked on the new dataset and on different models such as multilingual distilbert, BART and T5 but I faced technical issues with GCP.