

Multi-round Educational Assistant Language Model with Automated Context Integration

Authors: Farouk Boukil, Henrique Da Silva Gameiro, Ziwei Liu
Ecole Polytechnique Fédérale de Lausanne
June 6th, 2023

Abstract—Language models have emerged as powerful tools with significant potential to enhance educational experiences. As digital assistants, they provide students, as well as educators, with immediate access to tremendous amounts of information and tailored learning opportunities. In this paper, we detail and discuss our results in attempting to build an educational assistant language model that can handle both French and English and is able to perform multi-round interactions. To that end, use a range of techniques from supervised fine-tuning to reinforcement learning and context focus using BERT similarity. Our best model outperforms the other comparable baselines and models in state-of-the-art discourse coherence metrics, scoring 1.465 in DiscoScore Focus Difference and 0.804 in DiscoScore Sentence Graph [1].

I. INTRODUCTION

Education is a key aspect of human development, and effective instructional support plays a vital role in facilitating learning. However, the challenges of providing personalized and scalable educational assistance persist in that they are often based on anticipation of the difficulties met by the learners, and the solutions are prepared ahead of time. For example, traditional methods, such as tailored courses of different levels of difficulty and depth, as well as online support and tutoring, often fall short in meeting the diverse needs of learners. They leave big gaps in individualized instruction and support, often so that learners need to aggregate different sources of information to answer unanticipated questions. This paper, in the dawn of ChatGPT and other AI systems being used in education by students and educators alike, tries to go even further by designing a language model specifically to fill the role of an educational assistant. Our model aims to bridge the gap between learners and educational resources, offering personalized guidance, answering questions, and providing instructional support.

We fine-tune a collection of PEFT-quantized FLAN-T5 candidates for the question answering task, in order to choose one to fill the role of an educational assistant. We end up choosing FLAN-T5-large which outperformed all the other models on state-of-the-art BERT-based discourse coherence metrics: DiscoScore Focus Difference (**1.465**) and DiscoScore Sentence Graph (**0.804**).

We trained the candidates with Supervised Fine-Tuning (SFT) and Reinforcement Learning using Proximal Policy Optimization (PPO). To train a model with the PPO algorithm, we developed a Roberta-based regression reward model trained so that, given a question, it assigns higher scores to qualitatively better answers.

Our assistant can and is trained to handle multi-round interactions by automatically detecting, via Roberta-based similarity, one or many pieces of context from the previous rounds which are integrated into the generation process for the current round. Moreover, it is able to handle interactions both in French and in English.

We evaluate the assistant’s performance quantitatively using automatic evaluation metrics, namely BERT-score and BERT-based discourse coherence scores (BERT DiscoScores), and qualitatively by inspecting its outputs for queries from a wide range of disciplines and under both single-round and multi-round scenarios.

II. RELATED WORKS

Fine-tuning Large Language models with low resources: While Open AI’s InstructGPT yielded incredible results that lead to ChatGPT, they fine-tuned GPT-3, which has 175 billion parameters. Such models require immense computing power, money and electricity consumption. This is not the kind of experiment that could be replicated with low resources, such as consumer type of hardware. For this reason, there have been approaches to fine-tune large language models with lower amount of compute capabilities: Parameter efficient fine-tuning (PEFT). While the number of PEFT methods is enormous (see [2] for a summary of all methods), LoRa (see [3]) has already shown extensive usage and usefulness as reported in [2]. Adapters (see [4]) has also seen success as a simple yet effective method to fine-tune large models. While these methods reduce the GPU memory required for training, they don’t solve the issue of loading large models when training. The size of state-of-the-art models. For example, an open-source model such as Falcon, which has 40 billion parameters, has a size of approximately 85GB. For this reason, quantization is one of the techniques that has been developed to compress the size of a large language model to a smaller size (see [5]). Inevitably, recent approaches combine LoRa and quantization that show the possibility to fine-tune a 65B parameter model on a single GPU with 48 GB of VRAM (see [6]). Our study uses such approaches to be able to fine-tune a FLAN-T5 model with low resources and in only a few hours with around 10K samples.

Training Language Models to Follow Instructions With Reinforcement Learning: The paper titled *Training Human Models to Follow Instructions with Human Feedback* [7] presents a way to train a large language model to follow instructions using Proximal Policy Optimization (PPO) with a reward model trained on human feedback. In our work, we get inspired from the procedure presented in this paper to try and reinforce a fine-tuned FLAN-T5-base model into not necessarily following human instructions but most importantly producing better quality answers as scored by our reward model.

III. DATASETS

A. Reward Model Dataset

We train our reward model on a single dataset: ELI5-RLHF, which is widely used to train reward models for reinforcement learning training of language models. The dataset is composed of Reddit questions and the associated comments. Particularly, we use the partition “askscience” as it is the most similar to the data we use to train our assistant since it contains scientific questions and answers scored using the number of up-votes attributed by the users to each answer. In a similar way to other scientific forums, we expect that most of the time, the answer with the most up-votes is the most helpful, coherent, and accurate given the question.

Given a question provided with at least two answers, we consider composing every combination of the question with two of these

answers where one has more up-votes than the other. For every pair, the answer with more up-votes is labeled as "chosen", while the other is labeled as "rejected". Questions with less than two answers were filtered out.

Some of our experiences while training the final architecture show that demonstrations with a negligible difference in up-votes slow down training, over one epoch to say the least, with no palpable improvement in accuracy. On top of that, with the intuition that answers with comparable up-votes should be qualitatively similar, we can deduce that these samples could even be misleading because there is no clear reason to choose one over another when both are good enough and similarly both should, in theory, be rejected if they are no good. These are edge cases where the choice of the better answer no longer matters, and in which any output of the reward model when comparing such a pair would be acceptable. On the other hand, we want our model to be able to distinguish good and bad answers to a question when the contrast in up-votes, and thus in response quality, is significant.

In that spirit, we filter only the demonstrations where the difference in up-votes is significant. We tried many filtering techniques and thresholds, and the best one over one epoch was to keep demonstrations where the number of up-votes V satisfies:

$$V_{chosen} - V_{rejected} \geq 10$$

and

$$\frac{V_{chosen}}{V_{rejected}} \geq 2$$

The first threshold is meant to eliminate the demonstrations with a small number of up-votes where the difference is small. The second one, on the other hand, is oriented towards demonstrations with a high number of up-votes where it should take considerably more up-votes for the chosen answer to be effectively considered better than the rejected one.

B. Assistant Fine-Tuning and PPO Dataset

To train our assistant, we use a dataset with an original size of 6.5K QA pairs, which we augment to around 10.5K entries. The dataset is itself derived from an interaction history between users and GPT-3.5.

The interactions history contains, in each entry, the logged interaction and a confidence score from 1 to 5 (1 is worst, 5 is best) assigned by the user which reflects its confidence in GPT-3.5's final response given the question. Firstly, we filter out the answers with a confidence score lower than 4, since we do not want our assistant to learn from uncertain replies. Then, for each group of duplicate questions if any, we keep the QA pair with the highest confidence score, where ties are broken randomly. The reason we eliminate duplicate questions is to avoid the model overfitting them at the expense of worse performance on under-represented questions.

Together with the history of the interactions, we have access to the ground truth answers for each question. However, these answers are often a choice indication for a multiple choice question or a few words with little to no explanations and that hardly ever form a sentence. To overcome this issue, since we want our model to produce coherent well-explained multi-sentence replies, we prompt GPT-3.5 to provide a detailed and well-explained response to a given question knowing the ground truth answer. Our experiments show that GPT-3.5 usually does well in this task.

Another aspect of the data augmentation we perform is context integration. In fact, we aim for our model to support multi-round interactions, which, given a follow-up question, should be able to

attend to the relevant context in the previous rounds of interaction to formulate a more appropriate and accurate reply.

To that end, we use a Roberta-base pre-trained model to encode each QA pair from the previous rounds into a vector by averaging token representations, only after removing stopwords to avoid artificial boosts of the similarity. Similarly, we use the same model to encode the follow-up question and compute the cosine similarity between the latter's encoding and each context vector so that we select the two most similar contexts. These contexts are then integrated into the follow-up question's text in a zero-shot fashion. Here is an example of a context-augmented query, where the last question is the follow-up, and the first couple of QA pairs form the added context:

Question: *What are some advantages of using AODs vs scanning mirrors for OT positioning?* **Answer:** *One advantage of using acousto-optic deflectors (AODs) for optical tweezers (OT) positioning is that they have faster response times than scanning mirrors. This allows for more precise and rapid manipulation of particles or cells. Additionally, AODs do not suffer from mechanical wear and tear like scanning mirrors, resulting in improved long-term stability and reliability. AODs also have a larger deflection angle compared to scanning mirrors, enabling larger field-of-view and larger displacement ranges. Finally, AODs are relatively compact and can be easily integrated into existing microscope setups.* **Question:** *So, from the following options, which option would you choose as the answer to the question above: A. faster movements B. lower losses C. more uniform field D. larger range of movements* **Answer:** *The answer to the question is "D. larger range of movements."* **Question:** *There might be more than one correct option, is there any other option that also answers the question above?* **Answer:** *"*

The reason why we average the token encodings to obtain the context representation, or similarly the follow-up, is that taking the dot product between the averages is a cheap way to account for cross-similarity between tokens that do not necessarily occur in the same order or position between context and follow-up. Mathematically, if $\bar{c} = \frac{1}{n_c} \sum_i^{n_c} c_i$ and $\bar{f} = \frac{1}{n_f} \sum_j^{n_f} f_j$ are respectively the context and follow-up representations, then $\bar{c}^t \bar{f} = \frac{1}{n_f \times n_c} \sum_i^{n_c} \sum_j^{n_f} c_i^t f_j$ effectively accounting for cross-similarity via the $c_i^t f_j$ terms.

A problem that can arise with this approach is very long contexts that can result in the truncation of the follow-up question at tokenization time. However, in our dataset, that was hardly ever the case with only two contexts integrated into the follow-up question's text. In practice, we would dynamically set the number of most relevant contexts accounted for based on the length, in number of tokens, required by the augmented follow-up question effectively reducing the number of contexts in the event they are too large for the model to process. One could also alternate between summarization and context integration to make sure that the partially augmented follow-up is shrunk before adding more relevant context.

To keep the above format consistent across all QA pairs for training purposes, the questions in single-round interaction are also formatted the same way.

C. Fine-tuning data augmentation

We augmented our original dataset of 6.5K QA pairs to 10.5K. For this, we used a data augmentation scheme based on the answer to the original dataset's questions. Indeed, the original QA dataset also contains a field which gives a short answer to the question, and also sometimes a field that provides an explanation for the answer. Our idea is to leverage this answer and optional explanation to generate good demonstrations for training our models by adding the answer to the question to the prompt. We query Chat-GPT's API using the same questions with the following scheme: general prompt + answer prompt + explanation prompt + question (see appendix: A). This general scheme provides a GPT-3.5-like output, with an explanation that follows the correct answer. This prompting strategy exploits Chain-of-thought reasoning, which is an effective technique to get the correct answer for reasoning tasks as highlighted in [8]. We also give

the role of “teacher” in the pre-prompt which helps get more pedagogical answers. We handle duplicate answers to some questions as described in the previous subsection III-B.

D. Legal Rights and Ethical Concerns

The datasets we use are open-source either open-source like ELI5-RLHF, or for which we obtained exploitation rights from the Natural Language Processing Lab at EPFL. Therefore, they pose no legal issues.

Regarding ethics, ELI5-RLHF may pose some ethical concerns since it is built from Reddit questions and comments. As with any social media platform, bad behavior is not to be excluded. On the other hand, due to GPT-3.5’s configuration and filters against toxicity, we expect its outputs to be compliant with ethics. However, we cannot rule out that a malicious user could prompt GPT-3.5 into generating harmful replies. As much as we hope that such examples are not in the dataset, it remains difficult to completely eradicate them. Even so, it is impossible to make sure that the model would not learn any hidden biases during fine-tuning or that it wouldn’t generate toxic replies it learned from its pretraining. To mitigate this issue, what we opt to do is to suppress the tokens corresponding to toxic words from a predefined list of words, as well as preventing it from replying to questions that contain any of these words.

IV. REWARD MODEL

A. Model Architecture

The reward model is meant to score a QA pair. Therefore, we require it to have a relatively deep understanding of many aspects of natural language that can be used as quality measurements such as semantics, grammar, coherence, contradiction avoidance, redundancy avoidance..etc. To that end, we chose to build our model using a pre-trained transformer model as a base, on top of which we built a regression head. Transformers are able to encode many of the wanted aspects, to some extent, into the output representations from which the regression head can extract the score information.

At first, in the same spirit of InstructGPT’s reward model, we wanted to train a GPT-based reward model with the advantage that it has been tested and approved given InstructGPT’s success, but such models are very large compared to, for instance, BERT-based ones, and so it is computationally expensive to train and use them as reward models. As we are limited in terms of resources, we chose to use a BERT-based model.

Our regression model is composed of two parts:

- **Roberta-base** is the base of our model, which is frozen throughout training.
- **The Reward Head** is the regression head that computes the reward based on the base’s output.

To deal with inputs that exceed the maximum sequence length of our model, which is 512, we opted to use a pre-trained summarization model that summarizes the question jointly with the answer in the hope that it fits our maximum input size. We will omit discussing the details of such a model since our reward model is orthogonal to the choice of the summarization model.

B. Base Model (Roberta-base)

The base encodes the input into a representation based on which the reward regression head can compute a score that reflects the quality of the response in the context of the given question.

The base is frozen throughout training for two main reasons. First, we want to encourage generalization by making the regression head align with Roberta’s output which already encodes significant information about the input sequence. We believe by doing so that we avoid incorporating score information into the base’s weights during training which, based on the model’s capacity, could discard information learned from much wider corpora. We believe that the latter is important to derive scores in a more general setting. Second, training is cheaper because the regression head accounts for only about 10% of the model’s total complexity.

C. Reward Regression Head

The reward regression head computes the reward from the encodings computed by the base model. Figure [2] shows our final reward model’s architecture. Note that to simplify the schema, we omit representing the activations and normalization layers.

In the following sections, we describe each component and give the intuition and the justifications behind it, while discussing the improvements we made on previous architectures.

1) *Positional Encoding (PE)*: This component is necessary since the following is a (modified) transformer encoder layer. In fact, changing the order of tokens can affect the quality of the response, and we know that the attention mechanism does not take into account the token occurrence order. We use a sin-cos positional encoding as presented in the paper *Attention is All You Need* [9].

2) *Transformer Encoder Layer (with Integrated Projection)*: The base is frozen throughout training, therefore, we add a transformer encoder layer to learn contextual token representations that are prone to score information extraction. The architecture of this component is given in figure [3].

The inner feed-forward expands each input embedding to twice its size ($768 \times 2 = 1536$) with GELU activation since it was proven to outperform ReLU in deep neural networks and to accelerate learning due to the avoidance of vanishing gradient around zero. After that, instead of shrinking the dimension back to the original size like in the usual transformer encoder layer architecture, it projects the output directly to size 24. This is the reason we talk about integrated projection.

There are three main reasons for this design choice. Firstly, due to the concatenation operation that follows, it is unreasonable to directly concatenate 512 vectors of 768 parameters each as that would dramatically increase the complexity of the subsequent projections leading to overfitting, loss of information by aggressive subsampling, or even hardly stable training. Consequently, it was necessary to scale down the dimension of each token representation. Second, we have tried to use the usual transformer encoder layer with an extra projection block but it was highly unstable during training. Third, we believe that the information relevant to determining the impact of a single token on the overall score is minor, and that the interaction between different tokens is most relevant to measure the quality of a response. Consequently, we allowed ourselves to use a smaller token representation (24 instead of 786) hoping to learn a condensed representation of each token that summarizes its individual scoring capabilities.

Finally, to improve on other architectures we tried, we initialize the self-attention layer of this transformer using the parameters of the base’s last layer of self-attention parameters. We noticed that, compared to Xavier initialization used in previous attempts, this has significantly sped up convergence as the validation loss diminished considerably in the first epoch. The feed-forward however still uses Xavier initialization. On top of that, we added layer normalization after merging each skip connection. Dropout is used after every layer.

3) *Concatenation*: The token encodings of a sequence (dimension 512×24) are concatenated to form a single vector of dimension $512 \times 24 = 12288$ per sequence. This would be the final reward representation of the sequence, from which the reward score shall be derived.

4) *Projection Block*: This is a 2-layer feed-forward meant to process the learned reward-oriented sequence representation resulting from concatenation. Its first layer has a GeLU activation, while the second is linear. Dropout is used after every layer, and layer normalization comes after merging with the skip connection. The latter is a single linear layer with dropout.

The goal is to extract up to 64 of what we consider as reward metrics that can be interpolated to get the reward. The idea is that the sentence representation is projected via the skip connection to dimension 64 where the sentence level reward information is densely represented into up to 64 “metrics”, which are re-adjusted via the residual connection’s more progressive and non-linear projection block.

5) *Interpolation*: This is a single-layered linear feed-forward with dropout that computes the reward based on the extracted reward metrics. Its aim is to linearly interpolate those metrics to determine the reward.

D. Training Procedure

Our approach is inspired by InstructGPT’s training [7]. For each pair of sequences (g, b) answering the same question, where “g” is the question concatenated with the better answer, and “b” is the same but with the worse answer, we define the following log-sigmoid pairwise loss, where R is our model:

$$l(g, b) = -\log(\sigma(R(g) - R(b)))$$

The overall loss we minimize is:

$$E_{(g,b) \sim \mathbb{D}}[-l(g, b)]$$

The same question can occur in many demonstration pairs during the same training epoch which may lead to overfitting some questions that appear more often than others in separate training batches. To mitigate that, we batch the demonstrations that use the same question together in a single batch during training, so that their gradient update is combined and applied only once per epoch.

Figure [1] below shows the evolution of the reward model’s validation accuracy in the last training epoch. In the point-wise deviation from the mean accuracy, we can see that the variance of the validation accuracy is gradually diminishing, so the model is converging. Moreover, we decrease the learning rate following a linear schedule.

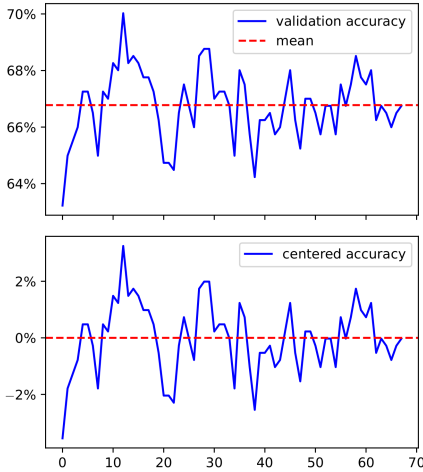


Fig. 1. The last training epoch’s validation accuracy and point-wise deviation from the mean.

E. Performance

Our model’s accuracy on ELI5-RLHF’s test set is 63.70%. In comparison, DeBERTa-V3-base, a regression reward model with 184m parameters so 37% more parameters than our model, has an accuracy of 59.92% on the same test set. Consequently, our model is quite performant for its size. In fact, on similar datasets such as Anthropic’s HH-RLHF dataset, other much larger pre-trained models have a performance that hardly ever exceeds the 70% mark.

On top of this, since we want to use our reward model in the PPO algorithm, and due to the fact that we are limited in terms of resources, it is an advantage to have a relatively small reward model that still performs well relatively to its size. In fact, with a very large model, PPO training would be so computationally expensive and time-consuming.

V. EDUCATIONAL ASSISTANT

A. Models

We considered many models for the generative part of the architecture. Namely, we considered: Llama-based models (Vicuna and Alpaca),

Falcon-7b, lmsys’ Fastchat T5, GPT-J, and FLAN-T5. These considerations are motivated by Huggingface’s recent [open LLM leaderboard](#), particularly the ARC benchmark. By leveraging the LoRa PEFT method (see [3]) and 4-bit quantization, it’s possible to fine-tune models with around 7B parameters using cheap hardware resources, as demonstrated by Qlora [6]. However, we chose Google’s FLAN-T5-large which has 783M parameters since we wanted inexpensive fine-tuning mainly due to the resource limitation, given we also wanted to experiment with PPO which is computationally very intensive. Due to these constraints, we picked a model with about 10x times fewer parameters compared to state-of-the-art open-source models, which we trained using LoRa. Our experiments could be repeated with such SOTA models, which would produce better results as highlighted by the scaling laws of LLMs described in [10] and [11]. We also fine-tuned FLAN-T5-base, which has 300M parameters, to assess the extent of the performance improvement when scaling up.

B. Supervised Fine-Tuning (SFT)

We fine-tune our model on the dataset described here III-B. As typically done for text generation tasks, we use the Negative Log-Likelihood Loss (NLL) on the log-softmax of the logits in the last layer so that, for a given question, the model is trained to greedily generate the next token. For FLAN-T5-base, we fine-tune for 4 epochs with a base learning rate of 1e-3 and cosine scheduling, which is suggested by this blog article: <https://www.philtschmid.de/fine-tune-flan-t5-peft>. FLAN-T5-large is fine-tuned with 2 epochs instead due to the increase in training time when increasing the number of parameters.

C. Proximal Policy Optimization (PPO)

The goal is to further calibrate the token distribution of the fine-tuned model to produce answers that are qualitatively better according to humans. Qualities of good answers for humans are such as helpfulness and non-toxicity, which we aim to reach with PPO. We apply this procedure only to FLAN-T5-base. On the larger model, it was too costly given our resources.

The PPO algorithm’s Markov Decision Process (MDP) is a one-armed bandit problem where the state is the query, the action is the response, and the reward is given by the reward model. Therefore, our PPO training works as follows: for each batch of queries in the training set, we generate responses using the language model, which are scored jointly with their queries using the reward model. The result is split into mini-batches used in PPO’s single training epoch per batch of queries. We use the PPO-clip variant of the algorithm.

During training, the language model’s generation configuration is similar but much more permissive than the final generation configuration. In training, we use $top-p$ sampling with $p = 0.9$, and each fifth of the training set, the probability is reduced to focus more on the calibration of more likely tokens. We use Google’s LION optimizer [12], which symbolically explores the optimization space before performing the next step, with learning rate $\frac{1.41 \times 10^{-5}}{3}$ since it is recommended to use one-third of the recommended learning rate for the task at hand. No learning rate scheduler was used.

VI. PERFORMANCE EVALUATION & ANALYSIS

A. Quantitative Evaluation

We evaluate our assistant, the fine-tuned T5-large model, against previous models we tried such as T5-base finetuned with and without PPO, as well as a pretrained GPT2-large model as an example of relatively sizable causal language model.

Our assistant was not designed with a data retrieval module that allows it to answer questions based on large educational corpora, so we do not expect it to be able to accurately answer every educational question. However, we expect the model to generate a coherent discourse that stays on track with the question, but without necessarily being able to truthfully and accurately answer that question. In other words, we expect our model to deliver a response that seems natural as if it were generated by a human, though the content might not be truthful.

In this spirit, we use a range of evaluation metrics that capture discourse coherence:

- DiscoScore Focus Difference [1]: A measure of the focus difference between two consecutive sentences, where a shared focus between two consecutive sentences can be through referencing or co-referencing. Note that, unlike the other metrics, lower is better.
- DiscoScore Sentence Graph [1]: A measure of the continuity between two consecutive sentences, where two sentences are considered to be continuous if the second is a plausible continuation of the first. It is derived using a graph-based approach.
- Entity Graph [13]: Another graph-based approach to text coherence. It uses a bipartite graph where entity transitions encode local coherence information.
- Lexical Chain [14]: Yet another graph-based approach to text coherence. It is similar to the "Entity Graph" approach, but generalizes more since it uses all lexical terms for its graph transitions to represent semantic connections and local coherence.
- Local Coherence (LC) [15]: It aims to capture how well adjacent sentences are related and how effectively they contribute to the overall understanding of the text.
- Rhetorical Coherence (RC) [15]: It aims to capture the logical flow and structure of discourse, including the proper sequencing of words, the use of cohesive devices (e.g., pronouns, connectors), and the establishment of relationships between different segments of a text.

On top of these metrics, we use BERT-score precision, recall, and F1-score to assess the contextual similarity between question and response.

Metric	Model			
	FLAN-T5-large (SFT)	FLAN-T5-base (SFT)	FLAN-T5-base (SFT/PPO)	GPT2-large
DS FocusDiff	1.465	1.825	1.970	0.160
DS SentGraph	0.804	0.803	0.665	0.701
Entity Graph	0.516	0.683	0.048	0.374
Lexical Chain	0.292	0.380	0.0278	1.405
RC	0.579	0.866	0.291	0.418
LC	0.220	0.286	0.120	0.171
BERT Precision	0.678	0.674	0.653	0.452
BERT Recall	0.624	0.637	0.481	0.521
BERT F1-score	0.643	0.649	0.549	0.478

Notice that FLAN-T5-large (SFT) performs better in terms of BERT discourse coherence metrics, FocusDiff and SentGraph, but it is outperformed by FLAN-T5-base (SFT) when it comes to more traditional discourse coherence metrics (LC, RC, Lexical Chain, and Entity Graph). We suspect one reason for that would be the base model’s repetitions during generation, although we penalize that, which may artificially boost these traditional metrics. This issue can also be reflected in BERT similarity scores (precision, recall and f1). Our inspection of the outputs of both models on around 100 QA pairs push in the direction of this hypothesis, and our team agreed that the large model’s generations were at least slightly better than those of the base one, even with the same generation configuration. The next section will dive more into the qualitative results.

The main advantage of the large model is, based on these results, its ability to keep the generation flow more continuous by having common focus elements between consecutive sentences. As a side effect, the model is more able to transition to different focus elements by having transition sentences that perhaps share different focus elements with the previous and subsequent sentences. Furthermore, its generation is richer in the sense that the similarity and graph-oriented discourse coherence and non-discourse metrics are overall lower than for the base model’s generations yet the former’s generations are qualitatively better.

FLAN-T5-base with PPO performs quite poorly, with the main issue being discourse coherence as reflected by the DiscoScore metrics. In

fact, the consecutive sentences it generates seem to be unrelated at times. However, the issue may be in the choice of hyperparameters, since we trained only for a single PPO epoch per mini-batch. With the resources we had available to us, it was too expensive to train for more than that even for such a relatively small model. Moreover, using a more powerful reward model and subtracting a baseline in the PPO update may have helped converge to a better model. That is because the value head appended to the model, which estimates the value of a state in the PPO algorithm, is also optimized by the PPO algorithm and may require more than one epoch per mini-batch to improve its estimates of the state values.

The GPT2-large pre-trained model performs relatively well, knowing it was not fine-tuned. Nevertheless, it remains widely dominated by FLAN-T5-large almost across every metric, most importantly when it comes to discourse coherence metrics.

B. Qualitative Evaluation

Automated evaluation metrics do not always correlate well with human judgment, therefore, we also provide an analysis of a few examples of questions and answers generated by the following models: T5-large (SFT), T5-base (SFT), T5-base (SFT/PPO), GPT2-large. The goal of this analysis is to understand the qualities of the answers of each model with a human perspective, and also identify failure cases. We selected 3 questions not seen by the models during training that cover different ranges of questions relevant to education. Full questions and generation can be found in appendix B1. All examples were produced using the same seed and not cherry-picked. The generation parameters are the same for all 3 models: top-p sampling with $p = 0.9$, a temperature of 0.3 and penalization for repetition (full details can be found on the generation script).

1) *Question 1 analysis:* This first question about ML was selected because it requires very complex reasoning and is fairly long. It will test the reasoning ability of the model (see here B2 for the outputs of the different models).

With this first question, we already notice that none of the models produce coherent answers. It seems like some parts of the answer make sense in isolation like: "the training process for a neural network with 10 layers is very challenging" for FLAN-T5-large’s answer. However, all models fail to pick up one choice and argue about why this choice is correct. This may be due to the training data, where there was sometimes only 1 correct answer, and sometimes multiple.

Although we see that the lack of coherence prevents the models from reasoning capabilities, we can see a gap in quality between the models trained without PPO and the model trained with PPO. This suggests that PPO training doesn’t improve the model in our case. In fact, the outputs of the model trained with PPO are all shorter compared to those without PPO.

2) *Question 2 analysis:* This second question was selected because it doesn’t require a lot of reasoning, but requires heavy domain knowledge (see here B3 for the outputs of the different models).

With this question, we can see that FLAN-T5-large’s answer explains each of the options. This a desirable quality in terms of an educational assistant. It means that the model is able to reproduce this scheme, which is present in the training samples, for out-of-sample data. FLAN-T5-base also gives an explanation for the answer. Although the explanation given in the output might be completely incorrect for both, it shows that both models learned desirable patterns during training. Also, it might suggest that both models struggle less with questions that are not entirely complex reasoning oriented.

3) *Question 3 analysis:* This second question was selected to test the abilities of the model outside multiple choice questions, and this is also a question which is not science oriented (see here B4 for the outputs of the different models).

We notice with this question that the beginning of the answer for both the large and base models are the same. One hypothesis about this occurrence is that both models learned during training to re-use some of the words of the question in the answer. This is not entirely surprising;

likely, a word present in the question is also part of the answer. Then both answers continue differently, but with similar reasoning. The answer produced by both models makes sense, and this is clearly what most people would answer. Unfortunately, both answers are wrong, since the correct answer is not obvious.

4) *General observations:* These examples show clearly that none of the models have reasoning capabilities, in particular math reasoning. Long answers from these models also seem to lack coherence (coherence was an issue already highlighted in our quantitative analysis). However, we see as well that some of the answers follow common-sense, and also provide an explanation in addition to the answer.

We don't find noticeable differences between the quality of the outputs from FLAN-T5-large and FLAN-T5-base in these 3 examples. This is not entirely surprising since both models have much fewer parameters compared to GPT-3.5, which was used to create the sample for training (the models we used have 300M and 783M parameters respectively, and GPT-3 has 175 billion parameters). Additionally, it is suggested in [11] that bigger models are required to have good performance in reasoning tasks. Another reason is that the questions we selected are very complicated and require a lot of specific knowledge and reasoning, it's unrealistic to expect correct answers to these questions from our model trained with low resources.

5) *Multi-Round interaction:* We also provide an example of multi-round interaction using our assistant model in appendix C. The question is the same as the third question in appendix B1, but following questions in the interaction are manually typed, and all answers are provided by FLAN-T5-Large. The previous set of questions + answers is provided as input to the model for the next generation at each round.

We can observe that our assistant model reacts well to follow-up questions and is able to answer by taking into account previous answers, despite the original answer to the question being still incorrect.

VII. CONCLUSION

In conclusion, our study explores the training of an educational assistant using a similar procedure as ChatGPT, incorporating techniques such as supervised fine-tuning (SFT) and Reinforcement Learning training using a reward model. Though we use a dataset of samples rated implicitly by humans to train our reward model, our reward model could be improved by rating outputs of the assistant model directly, and having more explicit feedback from humans (RLHF). Despite operating under limited resources for SFT and PPO training, our study yielded promising results, thanks to techniques such as PEFT, in creating an AI-powered educational assistant capable of providing helpful support to learners. We also highlight limitations in terms of the truthfulness of answers and coherence for such models, and the importance of model size for reasoning tasks.

LLMs could soon revolutionize education by offering tailored assistance anywhere, anywhen, and to almost any subject. However, there are still big challenges that need to be solved before deployment such as ethics and privacy when training an educational assistant on a large amount of data from unreliable user sources. There is also a responsibility to create models which answer factually and doesn't spread misinformation or bias, one of the missions of education. While our study only highlights these issues without providing direct solutions, we hope that future studies will also focus on them.

REFERENCES

- [1] W. Zhao, M. Strube, and S. Eger, "Discoscore: Evaluating text generation with bert and discourse coherence," 2023.
- [2] V. Lialin, V. Deshpande, and A. Rumshisky, "Scaling Down to Scale Up: A Guide to Parameter-Efficient Fine-Tuning," Mar. 2023. arXiv:2303.15647 [cs].
- [3] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "LoRA: Low-Rank Adaptation of Large Language Models," Oct. 2021. arXiv:2106.09685 [cs].
- [4] N. Housley, A. Giurghi, S. Jastrzebski, B. Morrone, Q. de Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly, "Parameter-Efficient Transfer Learning for NLP," June 2019. arXiv:1902.00751 [cs, stat].
- [5] A. Gholami, S. Kim, Z. Dong, Z. Yao, M. W. Mahoney, and K. Keutzer, "A Survey of Quantization Methods for Efficient Neural Network Inference," June 2021. arXiv:2103.13630 [cs].
- [6] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, "QLoRA: Efficient Finetuning of Quantized LLMs," May 2023. arXiv:2305.14314 [cs].
- [7] 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, and R. Lowe, "Training language models to follow instructions with human feedback," 2022.
- [8] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le, and D. Zhou, "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models," Jan. 2023. arXiv:2201.11903 [cs].
- [9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017.
- [10] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei, "Scaling Laws for Neural Language Models," Jan. 2020. arXiv:2001.08361 [cs, stat].
- [11] J. Wei, Y. Tay, R. Bommasani, C. Raffel, B. Zoph, S. Borgeaud, D. Yogatama, M. Bosma, D. Zhou, D. Metzler, E. H. Chi, T. Hashimoto, O. Vinyals, P. Liang, J. Dean, and W. Fedus, "Emergent Abilities of Large Language Models," Oct. 2022. arXiv:2206.07682 [cs].
- [12] X. Chen, C. Liang, D. Huang, E. Real, K. Wang, Y. Liu, H. Pham, X. Dong, T. Luong, C.-J. Hsieh, Y. Lu, and Q. V. Le, "Symbolic discovery of optimization algorithms," 2023.
- [13] C. Guinaudeau and M. Strube, "Graph-based local coherence modeling," in *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, (Sofia, Bulgaria), pp. 93–103, Association for Computational Linguistics, Aug. 2013.
- [14] Z. Gong, M. Zhang, and G. Zhou, "Document-level machine translation evaluation with gist consistency and text cohesion," in *Proceedings of the Second Workshop on Discourse in Machine Translation*, (Lisbon, Portugal), pp. 33–40, Association for Computational Linguistics, Sept. 2015.
- [15] B. T. M. Wong and C. Kit, "Extending machine translation evaluation metrics with lexical cohesion to document level," in *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, (Jeju Island, Korea), pp. 1060–1068, Association for Computational Linguistics, July 2012.

APPENDIX

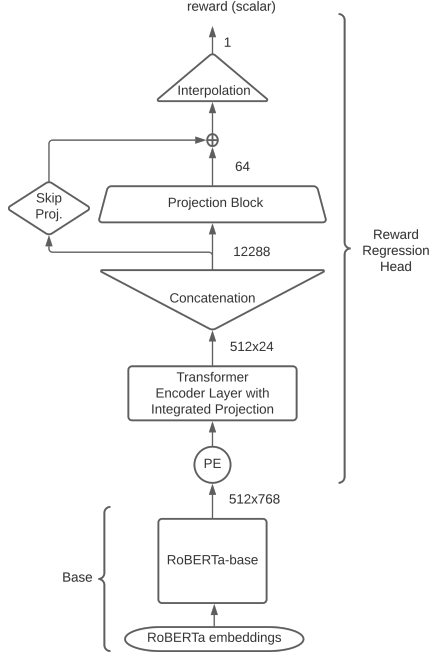


Fig. 2. Reward Model Architecture: The model is a combination of a pre-trained transformer encoder (Roberta-base) as base, and a learnable regression head. This is the architecture we retained.

A. Data augmentation prompt

General pre-prompt for multiple-choice questions: "From now on, you will act as a teacher in a science and engineering university. When a student asks you a question, you must provide all the reasoning steps before answering the question, you are forbidden to give away the answer before explaining your reasoning. Questions are multiple choice questions with possibly multiple correct answers"

Simple question pre-prompt: "From now on, you will act as a teacher in a science and engineering university. When a student asks you a question, you must provide all the reasoning steps before answering the question, you are forbidden to give away the answer before explaining your reasoning."

Answer pre-prompt: "You must give your reasoning and answer given that the correct answer is: " + ANSWER

Explanation pre-prompt: "Here is the explanation of the solution: " + EXPLANATION

B. Examples of generations

1) Questions:

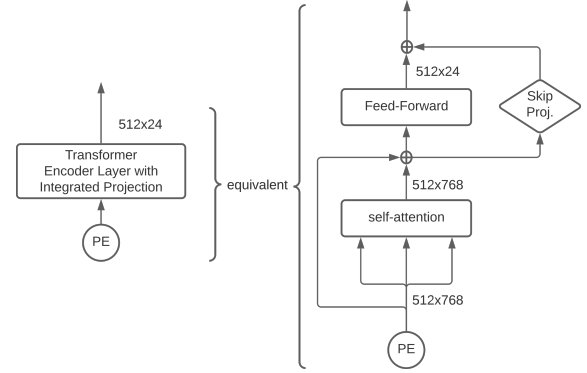


Fig. 3. Transformer Encoder Layer with Integrated Projection

Question	Choices	Correct Answer
You are doing your ML project. It is a regression task under a square loss. Your neighbor uses linear regression and least squares. You are smarter. You are using a neural net with 10 layers and activations functions $f(x) = 3x$. You have a powerful laptop but not a supercomputer. You are betting your neighbor a beer at Satellite who will have a substantially better scores. However, at the end it will essentially be a tie, so we decide to have two beers and both pay. What is the reason for the outcome of this bet?	1) Because we use exactly the same scheme. 2) Because it is almost impossible to train a network with 10 layers without a supercomputer. 3) Because I should have used more layers. 4) Because I should have used only one layer.	1) Because we use exactly the same scheme.
Sometimes, in Cl plasma etching, a corrosion phenomenon is observed in Al etching under the form of chlorine-containing residues remaining on the film side-walls. Which of the following is a correct approach to avoid this problem?	1) Immersing the wafer in a PGMEA developer 2) Gently blowing the wafer surface with nitrogen gun to create AlN gas 3) Dipping before etching the wafer in diluted acetone solution 4) Exposing the etched structure to a fluorine plasma immediately after the Cl plasma	4) Exposing the etched structure to a fluorine plasma immediately after the Cl plasma
Assume that you are part of a team developing a mobile app using Scrum. One of your colleagues suggests that your team should organize daily Scrum meetings to discuss the progress of the tasks and how to implement complex features. He especially wants to discuss the implementation of a feature that will allow users to scan a QR code to get a discount, and would like some input from the team. What are your thoughts on this?	-	The purpose of the daily Scrum meeting is to synchronize the team's work and to identify any impediments that might prevent the team from delivering the work. Therefore, the team should not be discussing the implementation details, of features during the meeting. Instead, the meeting should be solely dedicated to reporting the progress of the tasks.

2) Question 1 answers per model:

Model	Answer
FLAN-T5-large (SFT)	The correct answer is 2) Because it is almost impossible to train a network with 10 layers without a supercomputer. In this case, the training process for a neural net with 10 layers is very challenging because the activations functions are not optimized for the number of layers, and the training time is very long. Therefore, the best solution is to use more layers to improve the performance of the model.
FLAN-T5-base (SFT)	The answer is 1) Because we use exactly the same scheme. This bet is a regression task under a square loss. The neighbor uses linear regression and least squares to learn the neural network. The neighbor's score is determined by the activations functions of the neural net, which are the numbers that are given in the model. The neighbor will have a significantly better score if they use the same algorithm. The neighbor must have an optimal performance for the task, so the neighbor must have a superior score if the task is not a tie. The neighbor should be able to perform the task without a supercomputer, so the bet is not based on the assumption that the neighbor will have an optimal score.
FLAN-T5-base (SFT/PPO)	The answer is: 1). Because we use exactly the same scheme. 2. Because it is almost impossible to train a network with 10 layers without a supercomputer.

3) Question 2 answers per model:

Model	Answer
FLAN-T5-large (SFT)	The correct approach to avoid corrosion phenomenon in Al etching is 1) Immersing the wafer in a PGMEA developer. This method involves immersing the film in a solution of PGMEAA, which can be used for etching in Cl plasma. However, it may not be suitable for Al etch because it may leave residues on the film sidewalls. 2) Gently blowing the wafer surface with nitrogen gun to create AlN gas. This method does not provide any protection against corrosion, and it may cause a corrosion reaction. 3) Dipping before etching the wafer in diluted acetone solution. This method is not recommended for Al etched films, as it may result in a corrosion problem. 4) Exposing the etched structure to a fluorine plasma immediately after the Cl plasma. This method can help prevent corrosion, but it may not completely eliminate the corrosion problem.

FLAN-T5-base (SFT)	To avoid corrosion, we should first isolate the etching process and then apply a fluorine solution to the surface of the wafer. This will allow the etcher to create a fluoride solution that can be applied directly to the surface. Then, we can use a fluorosin solution to remove the chlorine-containing residues on the film sidewalls. The solution will be diluted to produce a fluoresine solution and then etched into the wafer to allow it to react with the etch material.
FLAN-T5-base (SFT/PPO)	To avoid this problem, we can use a PGMEA developer.

4) *Question 3 answers per model:*

Model	Answer
FLAN-T5-large (SFT)	I would suggest that the team should organize daily Scrum meetings to discuss the progress of the tasks and how to implement complex features. This will help to ensure that the team is on track and that they are not stalling on the task or feature that they are working on.
FLAN-T5-base (SFT)	I would suggest that the team should organize daily Scrum meetings to discuss the progress of the tasks and how to implement complex features. This will help them understand the process and the goals of the project, and will also provide feedback on the implementation of the feature.
FLAN-T5-base (SFT/PPO)	I would like to discuss the implementation of a feature that will allow users to scan a QR code to get a discount.

C. Multi-Round interaction

Question: Assume that you are part of a team developing a mobile app using Scrum. One of your colleagues suggests that your team should organize daily Scrum meetings to discuss the progress of the tasks and how to implement complex features. He especially wants to discuss the implementation of a feature that will allow users to scan a QR code to get a discount, and would like some input from the team. What are your thoughts on this?

Answer: I would suggest that the team should organize daily Scrum meetings to discuss the progress of the tasks and how to implement complex features. This will help to ensure that the team is on track and that they are not stalling on the task or feature that they are working on.

Question: Are you sure that complex features should be discussed here ?

Answer: I am sure that the team should discuss complex features in daily Scrum meetings. This will help to ensure that the team is on track and that they are not stalling on the task or feature that they are working on.

Question: What is Scrum ?

Answer: Scrum is a process for managing and coordinating the work of a team. It is a method used by agile software development methodologies to manage the flow of work and ensure that each team member contributes to the overall project.