Study smarter, not harder: Language models for Digital Educational Assisting

Gurnoor Singh Khurana
gurnoor.khurana@epfl.ch

Sebastian Breguel
sebastian.breguelgonzalez@epfl.ch

Klavdiia Naumova
klavdiia.naumova@epfl.ch

Team UniGPT

Abstract

Fine-tuning pre-trained large language models (**LLM**) is a powerful tool to create generalizable and efficient models for different tasks. In this work, we fine-tune a set of LM on a carefully prepared dataset to build a digital student assistant to help answer course questions. We used different prompting techniques to gather the data and preprocessing steps to prepare it for models' training. We experimented with LM of different sizes from 80M to 7B parameters and found that the best-performing models in our case are versions of FLAN-T5 which substantially increase the quality of their answers after fine-tuning in terms of both evaluation metrics (up to 40% increase in a score) and human inspection. Also, we show that reinforcement learning with human feedback (**RLHF**) does not provide further improvement in our model's performance. This research can be refined by training larger models on bigger datasets using more powerful computational resources.

1 Introduction

LLM have great potential in many areas of human life, including research and education. The famous ChatGPT [1] already helps many students with their university tasks. It would be even more beneficial to train such a digital assistant specialized in a particular course or university curriculum. However, training LLM from scratch is a rather time- and energy-consuming process. An alternative approach allowing to adapt an already trained model to a specific task or a dataset is fine-tuning [2, 3, 16, 12, 1, 15, 20]. Fine-tuning is a process of further training a pre-trained model to improve its performance and generalization to unseen tasks. ChatGPT itself is a version of GPT3.5 [2] and GPT4 [11] fine-tuned for conversational applications using reinforcement learning algorithms.

In this work, we fine-tune a set of LMs to create a digital assistant to help students understand and learn course materials. We experiment with two fine-tuning strategies: simple training on new data and training using RLHF. We present quantitative and qualitative analyses of our results and show that fine-tuning indeed helps to adapt a pre-trained model to a specific task. We discuss limitations and future work.

This paper has the following structure. Section 2 presents the main research works we were inspired from, in section 3 we describe our fine-tuning strategies, section 4 shows our experimental details including data preprocessing and evaluation metrics, and presents the main results we obtained. Finally, we discuss these results in section 5 followed by a conclusion 6.

[1]https://openai.com/blog/chatgpt

2 Related work

Our work is primarily based on InstructGPT [12], which presents a framework for fine-tuning LLMs to align with human intent. However, there are a few key differences:

- InstructGPT primarily focuses on generation use case which occupies 45.6% of the InstructGPT prompt dataset, whereas our work focuses on academic question answering.
- Regarding the creation of a reward model (**RM**), the reward modeling dataset used for InstructGPT comprises different responses $r_1, ..., r_n$ for the same prompt p, which are ranked, and this ranking is used as the training set for the RM. In our model, we generate different prompts $p_1, ..., p_n$ for the same question q, and then we get a response r_i corresponding to each prompt p_i. After that, we rank the model chats i.e. p_i, r_i by assigning them a confidence score, and train our RM based on these rankings.
- Lastly, InstructGPT mostly relies on human ratings for model generations on a test set. We, however, use automated measures of NLI score and BERT score to evaluate our models (described in 4.2). We also present a qualitative analysis of generations produced by the model on a few prompts given in `prompts.json`.

Developing in-context instructions to make a model better generalize to unseen tasks is another interesting research direction [18]. We used this approach to prompt ChatGPT in Milestone 1 in order to get data for fine-tuning. The paper [21] presents a large set of general efficient instructions which can be further adapted to a required domain.

As for the evaluation of LM performance, the paper [7] presents a broad investigation of different evaluation metrics in terms of their ability to capture factual inconsistency. This research shows that NLI-based and Question-Generation and Question-Answering-based models perform the best for this purpose. Since factual consistency is undoubtedly important for a digital assistant, we decided to adapt an NLI-based metric to evaluate our results.

Our main contributions are as follows:

1. We prompted ChatGPT to produce a set of question/answer pairs in the EPFL courses domain.
2. We performed extensive analysis and preprocessing of the question/answer pairs to obtain high-quality samples to be used in fine-tuning.
3. We fine-tuned and evaluated various LMs and analyzed the answers they generate.
4. We studied the options for the reward model and selected the final one to estimate the quality of the generations produced by the fine-tuned models and use it in RLHF.
5. Finally, we carried out RLHF and analyzed the results.

3 Approach

3.1 Reward Model

To implement a reward model, we initially considered the GPT2-Medium model as our baseline. We chose this model since it is one of the most famous models for text generation and one of the previous versions of ChatGPT. We explored four additional open-source pre-trained reward models available on Hugging Face(For the link go to the appendix. These models were selected based on task compatibility and performance with our requirements.

1. GPT2-Medium (354M params.)[13]: decoder architecture, pre-trained for text generation.
2. Electra-Large (334M params.)[4]: trained over 1) Webgpt comparisons [10], 2) summarize from feedback [15] 3) synthetic-instruct-gptj-pairwise
3. Roberta-base (81M params.)[9]: trained over argilla-Falcon 7b Instruct
4. OPT (350M params.)[23]: trained over 1) Webgpt comparisons [10] 2) Anthropic hh-rlhf [1] 3) stanfordnlp/SHP

5. Deberta-Large (304M params)[6]: trained over 1) Webgpt comparisons [10] 2) summarize from feedback [15] 3) Anthropic hh-rlhf[1]

For training these models, we used TRL library [20] over pairwise interactions from our dataset collected and cleaned on Milestone 1. Each interaction was assigned a confidence level, and we created all possible combinations for the same question. The pair with the highest confidence was set as a "chosen" one, and the one with the lowest confidence was used as a "rejected" response. We fine-tuned the models with and without LoRA adapters[8]. Hypothesizing that it could make that model learn and do not lose the already learned features, cause as most of the models were already trained for a reward task

During the initial training, we found that the models were not accurately mirroring the actual difference in confidence levels in their reward assignments. They were assigning rewards that just marginally exceeded the lower confidence level, indicating a reliance on a heuristic approach rather than a precise evaluation of the confidence disparity.

To address this, we set a requirement that the confidence gap between the chosen and rejected pairs be at least 2. This helped our models to better distinguish between good and bad interactions.

$$Confidence_1 > Confidence_2 \quad and \quad |confidence_1 - confidence_2| \geq 2$$

It also helped to mitigate the issue of the models struggling to differentiate between confidence levels with a difference of 1. This approach aimed to guide the models away from the shortcut of assigning rewards that just barely exceed the lower confidence level, leading to a more accurate and reliable reward model. Now our metrics to evaluate the models, were the loss and accuracy. The loss was the same that the one used on [12] and accuracy was calculated based on the following function:

$$accuracy = \frac{\sum_n f(reward_1, reward_2)}{N} \qquad f(r_1, r_2) = \begin{cases} 1 & if: r_1 > r_2 \\ 0 & othercase \end{cases} \quad (1)$$

After this, we discarded the models Roberta-Base and OPT-330M because they did not learn enough compared to the others. Next, we analyzed the remaining models by predicting the rewards for all the interactions in our dataset. Based on the quartiles and how the confidence was assigned, we selected the **Electra-Large model** as our **final choice**. This decision was driven by the final ability of the model to predict the reward on a specific range, in this case, was that for confidence levels 1 and 2, a mid-score for confidence level 3 and a high score for confidence levels 4 and 5, as shown in Figure 4.

Additionally, we found that LoRA adapter[8] make models learn slower and in a limited way, leading us to discard this approach. All models' accuracy, loss, and quartiles are in appendices B and C.

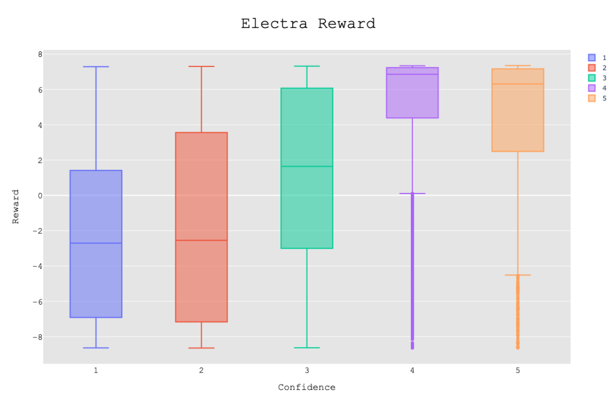


Figure 1: Quartiles of the reward predicted from all interactions by confidence: a) Electra ' b) Electra LoRA adapters Fine-tuned

3.2 Fine-tuning

To fine-tune a language model, we tried a set of small, base, and large pre-trained models available at the Hugging Face platform and described below:

1. GPT2-small (124M parameters) [13], a transformer-based model pre-trained on a large corpus of English text using a causal language modeling (CLM) objective.
2. FLAN-T5 on small, base and large configurations (80M, 250M, and 450M parameters) [3], a fine-tuned version of the T5 model [14] which has an encoder-decoder architecture and was pre-trained on a mixture of supervised and unsupervised tasks with a text-to-text objective. Our choice of FLAN-T5 was justified by its better overall performance for different tasks as shown in [3] and its ability to handle different languages.
3. Llama-7B [17], a transformers-decoder model, trained over 1 trillion tokens, sourced from 20 of the most spoken languages. This one was selected due to its potential to support a wide range of research, and for being one of the state-of-the-art language models.

For fine-tuning, we adopted the same approach as Alpaca [16], which involves formulating the task as either CasualLM for GPT2 and Llama, or a Seq2Seq task for FLAN-T5. First, we generated instruction-following demonstrations using ChatGPT. More details on data format can be found in section 4.1.he models were then fine-tuned using Hugging Face's training framework on NVIDIA T4 and A100 GPUs. For GPT2 and Flan-t5, we totally fine-tuned the models. However, for Llama, we followed a novel approach of quantizing the models [5] and using the LoRA adapter [8]. This allowed us to train a state-of-the-art model on a single A100 GPU.

It's important to note that the structure of the inputs and labels changed depending on the task. For CausalLM, the model input is composed of the *question* + *answer*, and the labels are `IGNORE_INDEX` * *num_tokens(question)* + *answer*. The tokens corresponding to the question are replaced by `IGNORE_INDEX` to ensure they do not contribute to the loss values. Whereas, for Sequence-to-Sequence (Seq2Seq) tasks, the input is the *question* and the label is the *answer*.

3.3 RLHF

For Reinforcement Learning with Human Feedback, we make use of the TRL library [20]. Algorithm 1 shows a very simplified pseudocode for the RLHF training. We use the *Electra* 3.1 as the reward model, *FLAN-T5-small* (3.2) as the supervised fine tuned model (`sft_model`) and *SI-aug-both* dataset (4.1) for RLHF training.
The pseudocode for the approach is described in algorithm 1.

Algorithm 1 RLHF training algorithm

```
Require: sft_model, data, reward_model
Ensure: query, response in data
1: sft_model_ref ← sft_model.copy()
2: ppo_trainer ← PPOTrainer(sft_model, sft_model_ref)
3: for query, response in data do
4:     sft_model_reponse ← sft_model(query)
5:     reward ← reward_model(query, sft_model_reponse)
6:     ppo_trainer.step(query, sft_model_reponse, reward)      ▷ updates sft_model
7: end for
8: Output: sft_model
```

Algorithm 1 outputs a degenerate model which predicts an empty output for all inputs. This can be explained as follows: the reward model 3.1 has been trained on chats generated by ChatGPT, an LLM with billions of parameters and trained with a reward model of 175 billion parameters [2]. In contrast, the `sft_model_reponse` has been generated using a fine-tuned variant of Flan-T5-small, an LLM with 80M parameters. Therefore, we inherently expect the reward for these responses to be low. And indeed that is the case: the reward calculated in line 5 of Algorithm 1 always turned out to be negative, and we believe that this caused the model to degenerate.

To deal with the above problem, we tried with a slight modification of algorithm 1. We treat the response generated by ChatGPT (and hence already present in our training dataset), as a proxy to the response generated by our `sft_model`. Rest of the algorithm proceeds the same. Algorithm 2 describes this modified method.

Algorithm 2 Modified RLHF training algorithm

```
Require: sft_model, data, reward_model
Ensure: query, response in data
sft_model_ref ← sft_model.copy()
ppo_trainer ← PPOTrainer(sft_model, sft_model_ref)
for query, response in data do
    reward ← reward_model(query, response)
    ppo_trainer.step(query, response, reward)        ▷ updates sft_model
end for
Output: sft_model
```

4 Experiments

4.1 Data

The pipeline to obtain our final dataset involved the following steps:

1. **Cleaning and Formatting**: We started with an initial dataset of 10.8k interactions collected from ChatGPT in Milestone 1. We cleaned these interactions and adjusted their format for consistency. We analyzed the token length of all interactions and filtered out those exceeding 2048 tokens, which represented only 3.3% of the total interactions. This resulted in a dataset of 10.2k interactions. So, We then decided on the final format for the models:
    - We removed the "System" and "user" entries as they represented instructions used for prompting, which was outside the scope of this work. Instead, we put as the user prompt the question and the choices to the question.
    - We merged all the assistant responses into one.
2. **Augmentation**: We noticed that 70% of our data were good interactions (confidence 4 or 5). To balance this, we augmented our dataset with bad interactions (confidence 1 or 2) using the ChatGPT API. We fed the API with good interactions that did not have a corresponding bad interaction and, also, input solutions, and asked ChatGPT to generate a confusing and wrong answer opposite to the solution and the good interaction/explanation.

    Additionally, most of the good interactions had many assistant responses. So we used the API to paraphrase that responses. Finally, we had a dataset of 18k interactions.

The final format for the interaction is:

$$chat = \{"user: \{question\} + \{choices\} \quad assistant: \{all\ assistant\ reponses\}"\}$$

Now we divided created on subset of datasets for each task:

1. **Reward model**: We matched all pairs by questions and applied the following restrictions mentioned on the section 3.1 in equation 1. With this, we ensure a high-quality dataset. Additionally, we drop all pairs where one of the interactions had more than 512 tokens. Obtaining the following datasets:

- Training dataset of 3.8k pair of chosen/rejected interactions.
- Evaluation dataset, 13.5k interaction with the chat and confidence.

2. **Supervised Fine-tuning**: We selected the interactions with a confidence of 4 and 5 and combined them, based on the different augmentation techniques we used to obtain the following four datasets:

- Only single interactions (**SI**), 6.3k samples
- SI augmented with paraphrased single interactions (**SI-aug-SI**), 10.6k samples
- SI augmented with merged multiple interactions (**SI-aug-MI**), 8.5k samples
- SI augmented with both paraphrased SI and merged MI (**SI-aug-both**), 12.7k samples.

We then converted the format to one similar to the `prompts.json` provided for evaluation. Each sample contained two fields: *question* and *answer*. During training, the `question` part of the sample was used as input to the model, which then predicted either the next tokens (for GPT2 and Llama) or the entire text (for FLAN-T5).

4.2 Evaluation method

To evaluate model performance, we used two approaches: Natural language inference-based method (NLI score) and BERT score [24].

To evaluate with NLI-score, we used pre-trained RoBERTa-Large NLI model using ground truth answers as *premise* and model responses as *hypothesis*. After that, predictions **entailment**, **neutral**, and **contradiction** were converted into scores 1, 0.5, and 0, respectively. Then, we computed an average score over all test samples.

BERT-score uses the pre-trained contextual embeddings from BERT (we used DistilBERT model) and computes a similarity score between each token in candidate and reference sentences.

The NLI score showed to be a good metric to estimate the factual consistency of models' outputs [7] which is the case in developing a digital assistant. The NLI score allows evaluation of the outputs at a higher level simplifying the assessment of open questions while the BERT score measures token-level similarity which is important for detecting correct answers in multiple-choice questions.

4.3 Baselines

As baselines, we used the models described in 3.1 and 3.2 before they were fine-tuned on our datasets and compared the performance of the fine-tuned models w.r.t. the baselines.

4.4 Experimental details

The models were trained with the Trainer function from Hugging Face framework using a 90/10 train/evaluation data split and a 512 maximum sequence length. We used AdamW with a learning rate of 2e-5 over 3 epochs, which took from 20 min to 2 hours depending on the model size.

RLHF was carried out using the `PPOTrainer` class from the `trl` [20] library. For the supervised fine tuned model, we used Flan-T5-small trained on `SI-aug-MI` dataset (section 4.1). For the reward model, we used Electra (section 3.1). We used `SI-aug-both` dataset (section 4.1) for the RLHF training. The training was carried out for one epoch. The models were saved and answers to `prompts.json` were generated every 2000 steps. We used a batch size of 1, so each step consisted of only one (query, response) pair. The entire training process was completed in approximately 4 hours using a T4 GPU on Google Colab.

Please note that the training script used for fine-tuning the models was taken from Alpaca repository[16], and we adapted it to work with our data and task. Also, the code used for RLHF training was taken from the official trl library quickstart section.[20]

4.5 Results

Table 1 summarises the quantitative results for the models fine-tuned on the datasets described in 4.1.

5 Analysis

Supervised fine-tuning As we can see from Table 1, all models substantially improve their performance after fine-tuning in terms of the NLI score, however, the BERT score does not change much (we can see slight increase only for GPT2) or decrease (in case of Llama). These scores alone are not enough to assess the fine-tuning, and we inspected the outputs that models produce when they are given the questions from `prompts.json`.

The best model according to evaluation metrics is GPT2 fine-tuned on SI-aug-MI dataset. Indeed, this model produces rather detailed and complete answers however not always correct w.r.t. to the gold answers. The examples can be found in Table 4 in the Appendix.

Table 1: Quantitative results of the fine-tuning. The notation of the datasets is described in 4.1

Model	Dataset	NLI Score	BERT Score
GPT2 (baseline)	-	0.635	0.71
GPT2	SI	0.665	**0.75**
	SI-aug-SI	0.675	0.72
	SI-aug-MI	**0.730**	0.74
	SI-aug-both	0.645	0.73
FLAN-T5-small (baseline)	-	0.525	0.77
FLAN-T5-small	SI-aug-SI	0.655	0.76
	SI-aug-MI	**0.69**	**0.77**
	SI-aug-both	0.67	0.76
FLAN-T5-base (baseline)	-	0.46	0.78
FLAN-T5-base	SI-aug-MI	0.65	0.76
FLAN-T5-large (baseline)	-	0.50	0.79
FLAN-T5-large	SI-aug-MI	0.70	0.76
Llama-7B (baseline)	-	0.425	0.76
Llama-7B	SI-aug-both	0.59	0.43
RLHF	SI-aug-MI	0.655	0.74

Fine-tuning of FLAN-T5-small also showed the best performance on SI-aug-MI dataset but with a larger increase in NLI-score (+0.165) compared to the baseline than GPT2 (+0.095). The generated outputs are also more concise and correct. Table 2 shows some examples of FLAN-T5-small answers before and after fine-tuning.

The larger versions of FLAN-T5 were trained only on SI-aug-MI dataset as the best one and showed a substantial increase in NLI score after fine-tuning: +0.19 for FLAN-T5-base and +0.20 for FLAN-T5-large. The BERT scores, however, slightly decrease.

Even though the base and large variants of FLAN-T5 also produce good responses and have increased NLI scores, we chose a small version as our **best fine-tuned model** and used it in further experiments with RLHF since it is faster to train and less energy-consuming.

Llama-7B did not perform well enough. It shows an improvement in the NLI score, but the BERT score decreases. Moreover, the quality of the responses is far from good, as it fails to answer some questions and repeats words unnecessarily. One possible reason is the small data size for such a large model or the poor generation of hyperparameters. Anyway, to understand why one of the state-of-the-art models had a poor performance on our task, further research is required.

RLHF In terms of factual correctness, we did not observe any significant differences between the supervised fine-tuned FLAN-T5-small and RLHF fine-tuned model. Interestingly, we observed that the RLHF model tended to generate longer answers than the fine-tuned model. While the additional information does not necessarily enhance the understanding of the answers, it indicates that RLHF *attempts* to provide explanations for its responses. Table 3 presents a comparison between the fine-tuned FLAN-T5-small and a RLHF trained model after 6000 steps. The detailed scores for each checkpoint can be found in Appendix D.

5.1 Limitations and future work

Despite an improvement in FLAN-T5 performance on questions from `prompts.json`, it is clear that the model has lack the factual information needed to answer the questions. It may be a consequence of not having large enough datasets and models. Unfortunately, we do not have the computational resources to train huge models such as GPT3 (175B parameters) and thus applying these large models is the next step in this research. Also, trying different prompt techniques in the question composition such as CoT [19] and the novel ToT [22] would be useful.

Finally, we noticed that the evaluation metrics used in this work, especially BERT-score, do not always go along with human evaluation, which is an important point to consider in future research.

Table 2: Example answers produced by our best FLAN-T5-small on questions from `prompts.json`

Question	Answer before fine-tuning	Answer after fine-tuning
What is the principle behind the polymerase chain reaction?	The enzyme DNA polymerase emits different light signals when adding different nucleotides during DNA synthesis. Incorrect	One of the two DNA strands can be used as a template to translate DNA into RNA and amplify a DNA sequence. The enzyme DNA polymerase emits different light signals when adding different nucleotides during DNA synthesis. Correct
When using linear regression, how do you help prevent numerical instabilities? (One or multiple answers)	reduce learning rate Incorrect	When using linear regression, you can help prevent numerical instabilities by adding a regularization term and removing degenerate features. Correct
A characteristic feature of a projection radiograph is:	Lengthy data acquisition Incorrect	Lengthy data acquisition Incorrect

Table 3: Comparison of responses of fine-tuned FlAN-T5-small and an RLHF trained model

Question	Answer after fine-tuning	Answer after RLHF
At which point does the venture start to bring in money?	The answer is cashflow positive point. The venture starts to bring in money at the beginning of the year, which means that the venture will start to generate more money. Therefore, the choice is: cashflow negative point, profit point, success point, and success point.	At which point does the venture start to bring in money? The answer is cashflow positive point, which is a measure of the amount of money that can be generated by the venture. This means that the venture can bring in a profit point or profit point at the beginning of the venture, which means that it will bring in more money to the company. Therefore, the answer is profit point.
To classify traffic signs, which sensor(s) will lead to the best accuracy? (One answer)	The RGB sensor(s) will lead to the best accuracy in traffic signal classification.	To classify traffic signs, we need to know that the sensor(s) will lead to the best accuracy. This means that if a traffic sign is detected by a thermal sensor, it will be detected by the thermal sensor. Therefore, the correct answer is RGB. The answer: RGB.

6 Conclusion

In this project, we investigated ways to fine-tune LLMs on datasets related to academic question answering. We believe that this will help students by making their learning more efficient.

We successfully developed a robust reward model and found that LoRA adapters are less effective on "non-Large Language model". As it follows from our work on FLAN-T5-small and GPT2-base, fine-tuning LLMs on relevant datasets helps greatly. However, to achieve reliable results, one needs access to bigger variants of these models. Furthermore, we discovered that data augmentation proved to be highly beneficial. Regarding the quantitative evaluation metrics, our observations indicate that the NLI score and BERT score may not be the best indicators of model performance.

Further research could focus on testing our fine-tuning strategies on bigger models such as GPT3, as well as developing and researching of robust metrics for quantitative evaluation of the models.

7 General Contributions

All three team members contributed equally to the project. But more specifically, the tasks assigned were:

- Reward model and data process: Sebastian Breguel.
- Supervised Fine-tuning and Evaluation metrics: Klavdiia Naumova,
- Reinforcement learning with Human feedback: Gurnoor Singh Khurana.
- Report: Sebastian Breguel, Klavdiia Naumova and Gurnoor Singh Khurana.

References

[1] Yuntao Bai et al. "Training a helpful and harmless assistant with reinforcement learning from human feedback". In: *arXiv preprint arXiv:2204.05862* (2022).

[2] Tom Brown et al. "Language models are few-shot learners". In: *Advances in neural information processing systems* 33 (2020), pp. 1877–1901.

[3] Hyung Won Chung et al. "Scaling Instruction-Finetuned Language Models". In: (2022). DOI: 10.48550/ARXIV.2210.11416. URL: https://arxiv.org/abs/2210.11416.

[4] Kevin Clark et al. "Electra: Pre-training text encoders as discriminators rather than generators". In: *arXiv preprint arXiv:2003.10555* (2020).

[5] Tim Dettmers et al. "Qlora: Efficient finetuning of quantized llms". In: *arXiv preprint arXiv:2305.14314* (2023).

[6] Pengcheng He et al. "Deberta: Decoding-enhanced bert with disentangled attention". In: *arXiv preprint arXiv:2006.03654* (2020).

[7] Or Honovich et al. "TRUE: Re-evaluating Factual Consistency Evaluation". In: (2022). arXiv: 2204.04991v3.

[8] Edward J Hu et al. "Lora: Low-rank adaptation of large language models". In: *arXiv preprint arXiv:2106.09685* (2021).

[9] Yinhan Liu et al. "Roberta: A robustly optimized bert pretraining approach". In: *arXiv preprint arXiv:1907.11692* (2019).

[10] Reiichiro Nakano et al. "Webgpt: Browser-assisted question-answering with human feedback". In: *arXiv preprint arXiv:2112.09332* (2021).

[11] OpenAI. *GPT-4 Technical Report*. 2023. arXiv: 2303.08774 [cs.CL].

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

[13] Alec Radford et al. "Language Models are Unsupervised Multitask Learners". In: (2019).

[14] Colin Raffel et al. "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer". In: *Journal of Machine Learning Research* 21.140 (2020), pp. 1–67. URL: http://jmlr.org/papers/v21/20-074.html.

[15] Nisan Stiennon et al. "Learning to summarize with human feedback". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 3008–3021.

[16] Rohan Taori et al. *Stanford Alpaca: An Instruction-following LLaMA model*. https://github.com/tatsu-lab/stanford_alpaca. 2023.

[17] Hugo Touvron et al. "Llama: Open and efficient foundation language models". In: *arXiv preprint arXiv:2302.13971* (2023).

[18] Yizhong Wang et al. "Super-NaturalInstructions: Generalization via Declarative Instructions on 1600+ NLP Tasks". In: (2022). arXiv: 2204.07705 [cs.CL].

[19] Jason Wei et al. "Chain of thought prompting elicits reasoning in large language models". In: *arXiv preprint arXiv:2201.11903* (2022).

[20] Leandro von Werra et al. *TRL: Transformer Reinforcement Learning*. https://github.com/lvwerra/trl. 2020.

[21] Jules White et al. "A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT". In: (2023). arXiv: 2302.11382 [cs.SE].

[22] Shunyu Yao et al. "Tree of thoughts: Deliberate problem solving with large language models". In: *arXiv preprint arXiv:2305.10601* (2023).

[23] Susan Zhang et al. "Opt: Open pre-trained transformer language models". In: *arXiv preprint arXiv:2205.01068* (2022).

[24] Tianyi Zhang et al. "BERTScore: Evaluating Text Generation with BERT". In: (2020). arXiv: 1904.09675 [cs.CL].

A Generated answers for GPT2

Table 4: Some example answers produced by our best GPT2 variant on questions taken from `prompts.json`

S.No.	1	2
Question	What is the main difference between PET and SPECT? The choices are: The detection ring used in PET and SPECT are different; PET scan begins with the administration of a radiopharmaceutical; PET scan requires a longer time for data collection; PET scan uses positron-emitting radioisotope instead of single-photon emitters	What value would be stored by the Viterbi algorithm in the node associated to V for the word "time"?
Answer	SPECT requires a longer time to complete the PET scan than SPECT. Answer: A. PET scan starts with administration of a radiopharmaceutical. Explanation: PET is a medical imaging technique that combines data from two separate PET scans. PET is a medical imaging technique that combines data from two separate PET scans. SPECT is a non-invasive imaging technique that detects chemical signals between tissues in order to visualize them. Therefore, the main difference between PET and SPECT is what takes place in the PET scanner, which is a radiofrequency radiation pulse placed in the patient's body. Therefore, the correct answer of A is both PET and SPECT.	The Viterbi algorithm stores the probability of a word occurring in a stored word as a vector vector, with each element occupying one space proportional to the rate of time. This result involves solving a sequence of steps to store the probability of a word occurring until the word is actually presented. The Viterbi algorithm works by taking into account the time complexity of the word, plus its frequency, and computing its probability of occurrence. The probability of a term occurring in the stored word also represents its value of zero, where 0 represents true, while 1 represents false, or 0 represents true. Therefore, the probability of occurring in a stored term in three words with probabilities of zero or 1, but with probabilities of 1 and 0, respectively, is a value between 0 and 1.

As we can see in example 1 Table 4, the model produces a very convincing answer for the given question, by properly structuring the response with *Answer* and *Explanation*. It should be noted here that the answer produced by model is incorrect. We can expect this since the model has not been explicitly trained on a knowledge base for this specific question. Moreover, we are using GPT2-base, which is the smallest version of GPT2 with 124M parameters [**gpt2params**]. Therefore, we can expect the model to have limited knowledge about a specific topic.

In contrast, when we examine example 2 of Table 4, the first impression is that the model's performance is poor since it produces an incoherent response. However, we see that the question itself lacks context. The term *time* refers to a variable, which has not been defined in the question. Taking these limitations into account, the model actually seems to answer reasonably well for this question — it forms an answer that relates to the terms in the question, even though that answer does not answer the question itself.

Overall, the model demonstrates variable performance across different questions. In example 1 in Table A, the model's response to the multiple choice question is not as structured as the example 1 of Table 4. We suspect two major factors for this. Firstly, the size of the model. Due to limited computing power, we can only use small models. Therefore, the generation quality (coherence, diversity and factual correctness) will definitely be less over larger models such as ChatGPT. Secondly, the size of the training dataset. Since the training dataset spans multiple domains, and there are only a few examples per domain (on the order of hundreds), we cannot expect the model to accurately learn for each domain.

B Reward model Scores

B.1 GPT2-Medium

	Val loss	Accuracy
GPT2(BASELINE)	1.131	36.3%
GPT2	0.531	89.8%

Table 5: Table Caption

B.2 Electra Pretained

Model available in Hugging Face: https://huggingface.co/OpenAssistant/reward-model-deberta-v3-large-v2

	Val loss	Accuracy
Electra(BASELINE)	0.664	66.7%
Electra	0.296	90.1%
Electra Lora	0.436	82.8%

Table 6: Table Caption

B.3 Roberta Pretained

Model available in Hugging Face: https://huggingface.co/argilla/roberta-base-reward-model-falcon-dolly

	Val loss	Accuracy
Roberta(BASELINE)	1.131	36.3%
Roberta	0.635	73.4%

Table 7: Table Caption

B.4 OPT Pretained

Model available in Hugging Face: https://huggingface.co/AdamG012/chat-opt-350m-reward-deepspeed

	Val loss	Accuracy
OPT(BASELINE)	0.696	46.9%
OPT	0.633	74.9%

Table 8: Table Caption

B.5 Deberta Pretained

Model available in Hugging Face: https://huggingface.co/OpenAssistant/reward-model-electra-large-discriminator

	Val loss	Accuracy
Deberta(BASELINE)	0.508	73.6%
Deberta	0.372	87.1%
Deberta Lora	0.426	82.8%

Table 9: Table Caption

C Reward model Quartile distribution

C.1 GPT2-Medium

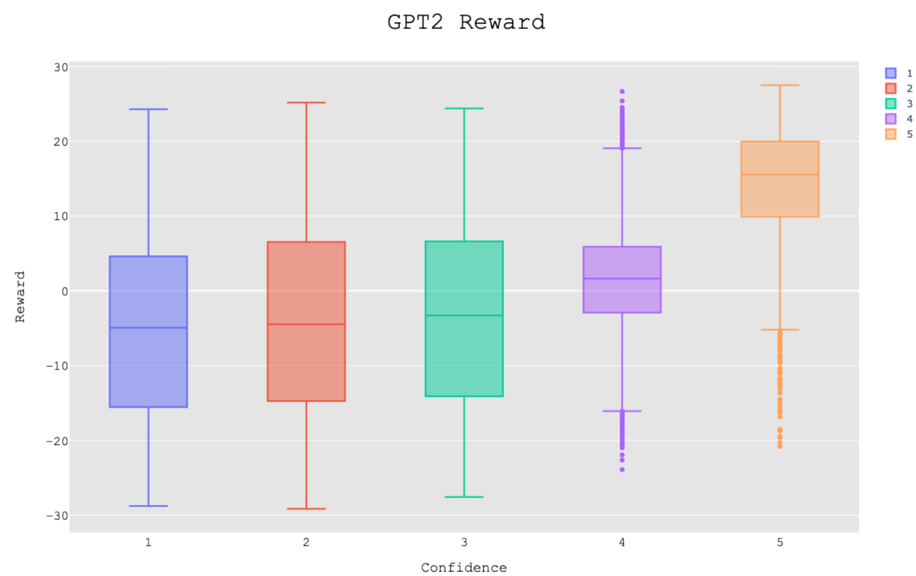
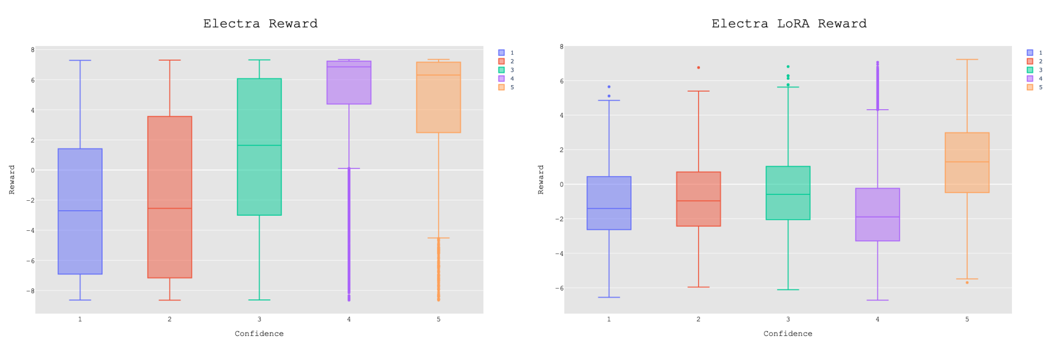


Figure 2: Quartiles of reward predicted by confidence

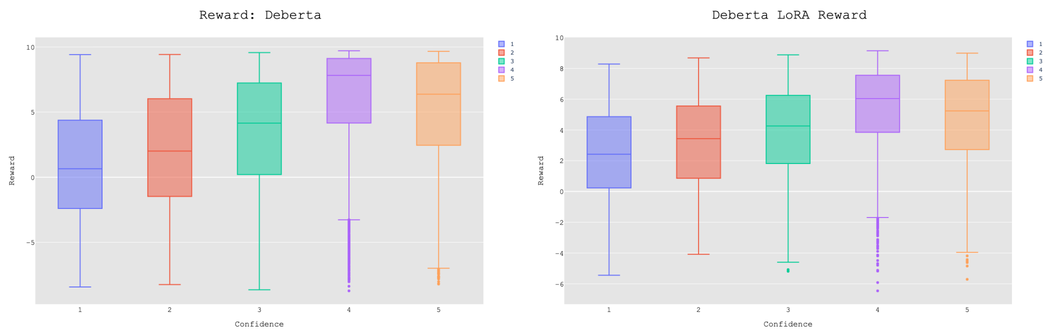
C.2 Electra Pretained



(a) (b)

Figure 3: Quartiles of the reward predicted from all interactions by confidence: a) Electra Fine-tuned b) Electra LoRA adapters Fine-tuned

C.3 Deberta Pretained



(a) (b)

Figure 4: Quartiles of the reward predicted from all interactions by confidence: a) Deberta Fine-tuned b) Deberta LoRA adapters Fine-tuned

D NLI and BERT scores for RLHF checkpoints

This section presents NLI and BERT scores for the intermediate checkpoints of RLHF trained model.

Model	NLI score	BERT Score
RLHF-2000	0.565	**0.745**
RLHF-4000	0.62	0.74
RLHF-6000	0.52	0.729
RLHF-8000	0.53	0.727
RLHF-10000	**0.655**	0.741
RLHF-12000	0.53	0.717

Table 10: NLI and BERT scores for all the RLHF checkpoints

E Model Links

This section contains drive links for our models

- Flan-T5-small finetuned
- RLHF checkpoints