Mini-GPTA: Distilled ChatGPTeaching Assistant for EPFL Courses

Yiyang Feng* and **Haotian Wu*** and **Maciej Styczen***
CS-552 MHY Team, EPFL, Switzerland
{firstname.lastname}@epfl.ch

Abstract

In recent years, the development of autonomous AI tutors for educational purposes has attracted substantial attention. However, current methodologies either require large volumes of human-labeled data for training smaller models or pose challenges due to computational resources and closed-source issues when using large language models. This paper introduces Mini-GPTA: Mini-GPTeachingAssistant, a specialized chatbot targeting course content for EPFL engineering students, which is a GPT-2-medium model distilled from ChatGPT. Through interaction with ChatGPT, we obtain human-machine dialogue data and augment it with external datasets. These data then feed into our instruction tuning and Reinforcement Learning with Human Feedback (RLHF) process, resulting in a small yet efficient model. Mini-GPTA outperforms baseline methods by a considerable margin, achieving up to 10 times higher similarity with ground truth, demonstrating the effectiveness of our approach. This work contributes a potential pathway for achieving resource-efficient, specialized educational domain chatbots, sparking new opportunities for individual researchers and smaller institutions to create high-performing AI assistants [1].

1 Introduction

Chatbots find extensive application in a variety of domains, including customer support (Xu et al., 2017), finance (Okuda and Shoda, 2018), health (Nadarzynski et al., 2019), human resources (Drozdal et al., 2021), and particularly, education (Hiremath et al., 2018). In the educational context, there's a growing interest in developing sophisticated and knowledgeable AI tutors capable of autonomously answering course content questions, thereby alleviating the workload of Teaching Assistants (TAs) and ensuring meaningful learning experiences for students. With the advent of transformer-based models (Vaswani et al., 2017) like BERT (Devlin et al., 2018), GPT (Radford et al., 2018), and others, there has been a substantial improvement in language comprehension by these models. Recent development of Large Language Models (LLMs) (Brown et al., 2020), as well as advances in Instruction-tuning (Ouyang et al., 2022), have further propelled this progress, enabling language models to better align with human behavior and generate practical texts, despite the considerable computational resources required. These developments have made educational chatbots a possibility.

Under constrained computational budgets, current methodologies resort to training chatbots in two ways: (i) fine-tuning smaller language models (<1B parameters) or (ii) zero-shot inference or prompting to LLMs. Nevertheless, both approaches have their drawbacks. Fine-tuning small language models for a dialogue agent necessitates the collection of a large volume of human-labelled dialogue data, demanding substantial human effort. As for LLMs, not all large language models, such as ChatGPT and Bard, are open source. Even though open-source LLMs like LLaMa (Touvron et al., 2023) and Alpaca (Taori et al., 2023) do exist, training these models remains a daunting task for individual researchers, and their performance lags behind the state-of-the-art LLMs.

Our work draws inspiration from knowledge distillation (Hinton et al., 2015), where we attempt to compress an LLM into a smaller model for a specific task with minimal performance degradation. Specifically, we narrow our focus to training an intelligent chatbot specialized in answering course content (quizzes, final exam questions, etc.) for EPFL engineering students. Accordingly, we train a Mini-GPTA, a Mini version of the distilled ChatGPTeaching Assistant for EPFL

*Equal contribution. See Appendix A for the summary.

[1]Our data and code are available at https://github.com/CS-552/project-m3-mhy

courses. In this process, we first interact with the large-scale model, ChatGPT, to acquire human-machine dialogue data on EPFL course contents, label a confidence score, and further prompt external datasets MATH (Hendrycks et al., 2021) and OpenBookQA (Mihaylov et al., 2018) to supplement our dialogue data. We then apply instruction tuning and Reinforcement Learning with Human Feedback (RLHF) (Ouyang et al., 2022) on our collected human-machine dialogue data to train a smaller model, approximately 100M to 300M parameters, enabling our model to emulate the responses of ChatGPT to user queries. Consequently, we train a smaller Reward Model (RM), Supervised Fine-Tuning (SFT) model, and the SFT model further fine-tuned in an RLHF fashion with our RM and Proximal Policy Optimization (PPO).

We use GPT-2 (Radford et al., 2019) as our base for all models, and our Mini-GPTA significantly outperforms baseline methods, achieving up to 10 times higher syntactic similarity with ground truth, with only 2‰ parameters of ChatGPT (175B). Our reward model also demonstrates effectiveness by outperforming baseline methods by 14.6% accuracy improvement with a more significant positive correlation of confidence scores. We further analyze the generated text from a qualitative perspective, discussing the potential and limitations of Mini-GPTA. In summary, our contributions are as follows:

1. We prompt ChatGPT with EPFL course contents (quizzes, exam questions) and obtain a large dialogue data set, specifically with a confidence score for each interaction. We also employ two external datasets to prompt ChatGPT to acquire augmented educational dialogue data.

2. We use the educational dialogue data generated by ChatGPT and distill ChatGPT (175B) to a much smaller language model, GPT-2 (355M), through instruction tuning and RLHF. We eventually obtain our educational chatbot Mini-GPTA.

3. We prove that our Mini-GPTA outperforms multiple baseline methods with 10× higher syntactic similarity, and demonstrate the effectiveness of our reward model with 14.6% accuracy improvement (65.3% accuracy over 50.7% in baseline models). We also discuss the potential and the limitations of Mini-GPTA.

2 Chatbot Data Preparation

2.1 High-Level Tasks for Collected Data

We will use our collected educational chatbot data for the following tasks:

1. **Reward Model (RM)**: Given the whole dialogue sequence, predict a scalar indicating the reward of the dialogue.

2. **Supervised Fine-tuning (SFT)**: Given the information from previous dialogue turns, generate the answer for the last turn.

3. **Reinforcement Learning (PPO)**: Further fine-tune the SFT model with PPO policy.

2.2 Data Source

We mainly use our Educational Dialogue Data (EDD) in milestone 1, which will be the questions and corresponding answers. One question may correspond to multiple answers, one of which is the ground truth provided by the teachers, and students annotate the other answers with the help of ChatGPT. We also use two extra public datasets to augment the training set for fine-tuning our final assistant model. One is MATH Dataset (Hendrycks et al., 2021), which is a new dataset of 12,500 challenging competition mathematics problems. Each problem in MATH has a full step-by-step solution that can be used to teach models to generate answer derivations and explanations. The other is OpenBookQA Dataset (Mihaylov et al., 2018), which is a question-answering dataset modeled after open-book exams for assessing human understanding of a subject. It consists of 5,957 multiple-choice elementary-level science questions (4,957 train, 500 dev, 500 test). We will use the demonstration data for all three models and two external datasets only for SFT and PPO models.

2.3 Preliminary Data Analysis and Cleanup

Educational Dialogue Data: We begin by analyzing the provided dataset of questions and interactions. Starting with some general numbers, we have a `solutions.json` file with 4,450 questions and ground-truth answers, and a `interactions.json` file with 10,835 student-generated interactions. Each interaction is related to one of the questions from the solutions file, and

is accompanied by the annotator's perceived confidence in the correctness of the answer on a 1-5 scale.

To prepare the dataset for training, we perform a number of data cleanup steps: (i) We remove 131 interactions with confidence scores outside the [1,5] range. (ii) Some questions are duplicated - we merge questions that have the same body, choices, and answer into a single question. This reduces the number of questions from 4,450 to 4,217. (iii) We remove 37 questions that refer to figures - they are not possible to answer in this setting, and it is not meaningful to have them included in the dataset. (iv) Since most of the example has a format that repeats the pattern "User: {...} Assistant: {...}", we filter all examples without this format. (v) We remove all examples with tokenized lengths greater than 1024 (the max context length of GPT-2) for the SFT model and 512 for the PPO model.

After the cleanup, we are left with 4,180 questions, out of which 2,822 are multiple-choice, and 1,358 are open. 3,855 of the questions have an answer (325 don't), and 1,132 of the questions have an explanation (3,048 don't). On average, per question (i.e. per `sol_id`), we have 2.54 interactions.

MATH Dataset: The original MATH Dataset consists of open-ended math questions of different levels of difficulty. We use the MATH data to augment our open-ended question part of EDD. It has a train-test split and is further split into question domains. We merge all of the data together, filter all duplicated examples, and remove examples with anomaly levels. We then sampled 1,000 examples from 12,500 of them. The distribution is shown in Appendix B.1.

OpenBookQA Dataset: The OpenBookQA Dataset includes 5,957 scientific multiple-choice questions for elementary school students. We use this dataset to obtain more multiple-choice questions. We merge all data from different splits, remove duplicated examples, and sample 1,000 examples from the original dataset.

2.4 Data Augmentation

Educational Dialogue Data: The answers provided in the `solutions.json` file are typically concise, often limited to indicating the correct responses to multiple choice questions with an optional brief explanation. These responses starkly differ from the expectations for an interactive assistant experience. To bridge this gap, we augment the initial dataset with ground-truth positive examples, which are generated using ChatGPT based on the information found in the `solutions.json` file. We supply ChatGPT with the correct answer and, if present, the explanation, and instruct it to create a justification for the answer. Given that these interactions represent ground truth examples, we assign them a confidence score of 6, indicative of high-quality answers. Although we experimented with generating negative examples, as demonstrated in Appendix B.2, this approach did not yield satisfactory results for our reward model. Thus, we solely use ground-truth examples for data augmentation.

Further, we augment the original dialogue dataset by breaking it down into multiple examples based on the number of dialogue turns. Specifically, each original example is divided into multiple examples that comprise the current dialogue turn and all previous information. This approach is employed exclusively for the SFT and PPO data.

MATH Dataset: As the original MATH Dataset is not formatted for dialogue question-answering, we use ChatGPT to enhance the ground-truth data in a similar way to the Educational Dialogue Data augmentation. We supply the question and correct answer to ChatGPT, which in turn repeats the answer and provides a justification. We then use the question as the user query and the ChatGPT response as the answer, concatenating these elements in the same format as our EDD.[2]

OpenBookQA Dataset: The OpenBookQA Dataset is presented in a multiple-choice format. It only includes an answer key, rather than text generated by a chatbot. Moreover, the question and the choices together form a complete sentence. These individual components are incomplete and cannot be directly inputted into our language model. Therefore, we initially prompt ChatGPT with the original question and choices to generate complete sentences of questions and choices. Following this, we input the generated questions, choices, and the correct answer key into ChatGPT to obtain a response that repeats the answer with justification. The generated query and response are then concatenated in the same format as our EDD.[3]

[2] We filter all examples with tokenized lengths greater than 1024 and retain 992 examples.

[3] All 1000 examples have tokenized lengths less than 1024 so we keep all of them.

2.5 Dataset Structure and Splits

Educational Dialogue Data: To convert the interactions into dataset entries, from each interaction, we construct a JSON object containing the `sol_id` identifying the question to which the interaction relates, the `confidence` containing the confidence score provided by the annotator, and `interaction` - the concatenated interaction string, constructed according to the project description.

We split the dataset into three parts:

1. Dataset for the Reward Model, 47.5% of the data, split into train/validation subsets into 90:10 proportions.

2. Dataset for the Supervised Fine Tuning, 47.5% of the data, also split into train/validation subsets into 90:10 proportions.

3. Test set, 5% of the data, which we will use for the evaluation of finetuned reward model and the final assistant model.

Note: the split is performed at question level (i.e. `sol_id` level), meaning that all interactions corresponding to a single question will be in the same split. This way we make sure that the splits are completely distinct.

Augmented External Data: We only use the MATH and OpenBookQA data for SFT and PPO models. Specifically, for the SFT model, we only augment the training data from EDD with all 1,992 augmented examples from MATH and OpenBookQA. We further sample 2,000 from the SFT training data as our PPO training data.

2.6 Potential Issues with the Data

Subjectivity of Confidence Scores: During the annotation process, there was no formal guideline on how the confidence scores should be determined, and there will definitely be a large degree of variance across users. Therefore, it might not be meaningful to compare two interactions for the same question based on their confidence scores, since the confidence scores are assigned by two different annotators who might have different standards. Unfortunately, the subjective confidence score is the only measure of the quality of the interactions available to us, so we will make use of it during the training, but we will keep in mind that this might be a serious limiting factor for the quality of our final assistant. One way of improving this would be introducing a per-annotator normalization. Subtracting the annotator's mean confidence rating from all their confidence scores could help to mitigate the issue of harsh/lenient annotators. However, this is not possible for us since we were not given the annotator ids along with the interactions, so it is not possible to group the interactions by their annotator.

Figure 1: Distribution of confidence scores.

Skewed Distribution: As you can see in Figure 1, there is a strong bias towards the positive scores, with almost half of the interactions being labeled with a maximum score of 5. This will make quantifying the quality and comparing two different demonstrations harder.

Ethical Issues and Hallucinations: Moreover, we use ChatGPT to generate augmented ground-truth examples, therefore our ground-truth data can itself contain false and misleading information. We accept this risk as we cannot use the provided answers as training examples directly, but we are aware that this might have a side effect of inheriting ChatGPT's biases and potentially learning from its hallucinations.

3 Pipeline of Mini-GPTA Training

Before introducing our Mini-GPTA, we define our problem and notations. A dialogue z consists of an instruction x and a demonstration y, where x is the previous dialogue information and the current user query, and y is the generated answer. We use the information (z, x, y) in each example during training and only use the instruction x during the evaluation to generate the answer y.

In Section 2.3, the assistant's response begins with "Assistant: ". We split the dialogue text at this string, using the left part with the string as the instruction and the right part as the demonstration.

We utilize multiple GPT-2 models (Brown et al., 2020) to train our Mini-GPTA. These models are chosen for their suitability within our resource constraints. Besides, they are pretrained on a large corpus and showcase impressive few-shot and zero-shot capabilities. Our training process incorporates three techniques derived from InstructGPT (Ouyang et al., 2022). We implement these models with PyTorch (Paszke et al., 2017) and Huggingface (Wolf et al., 2019) libraries.

3.1 Reward Modeling (RM)

We use the GPT-2-small model for our reward model, which has 117M parameters, instead of larger models because they suffer from unstable training problems (Ouyang et al., 2022). Our setup is almost identical to that of InstructGPT (Ouyang et al., 2022), with some minor differences: (i) We set the max length of the GPT-2 tokenizer to its maximum, 1024 to read the entire conversation interaction. (ii) Since our GPU memory is limited and the context length of the reward model is long, we found it impossible to train the reward model in a batch fashion that updates the model after all comparisons from the same question. As a result, we only treat one comparison as a single batch element. Our training batch size is set to 1 due to computational constraints.

Specifically, our modified loss function for the reward model is:

$$\mathcal{L}_{\text{RM}}(\theta) = -\log\left(\sigma(r_\theta(z_w) - r_\theta(z_l))\right) \quad (1)$$

where $r_\theta(z)$ is the output of the reward model (a scalar) for the input dialogue z. z_w is preferred over z_l and they are from the question in our student annotated dataset.

3.2 Supervised Fine-Tuning (SFT)

After training the reward model, we proceed with further training our Mini-GPTA. Specifically, we fine-tune the pretrained GPT-2-medium model, which consists of 335 million parameters. We opted to continue using the GPT-2 series for our SFT model as it aligns with our reward model, ensuring better compatibility between the two. However, we refrained from employing a larger pretrained GPT-2 model as our SFT model due to two primary reasons. Firstly, our computational resources are limited to a single NVIDIA T4 GPU with 12GB of memory, making it unsuitable for loading larger models, even with reduced input max length. We also attempted to acquire a more powerful GPU, such as the A100 40GB, but our request was unsuccessful due to high demand. Secondly, based on our understanding, PPO training consumes more GPU memory compared to training a standalone pretrained model. Hence, we decided against employing a significantly larger pretrained model.

Regarding training specifics, we set the maximum length of the tokenizer to 1024 and utilized the Adam optimizer with a learning rate of `1e-5` along with a Cosine scheduler. The loss function for the SFT model is defined as follows:

$$\mathcal{L}_{\text{SFT}}(\theta) = -\sum_{T} \log P(y_t|\{y_{<t}\};\{x\}) \quad (2)$$

Our model represents the instruction as x and the demonstration as y. When utilizing the `cross_entropy` function to compute the loss, parallel computing becomes challenging due to varying demonstration lengths across the data. We set the batch size to 1 to address this issue, ensuring each data piece is processed individually. Our experiments showed that the pretrained GPT-2-medium model typically converges within 2 to 3 epochs. We save the model that exhibits the best performance on the validation set.

3.3 Reinforcement Learning with PPO

Once again, following Ouyang et al. (2022), we further fine-tune our SFT model in a Reinforcement Learning (RL) fashion with PPO policy. We treat instructions x as states, demonstrations y as actions, and rewards as $r_\theta(z) = r_\theta(x, y)$. We treat our language models for generating texts as policies. Specifically, we have a baseline policy $\pi_{\text{SFT}}(y|x)$ from our SFT model, and we will obtain an RL policy $\pi_{\text{PPO}}(y|x)$ from the baseline policy via PPO policy learning.

Practically, we add a KL divergence term (Csiszár, 1975) to the reward function as a penalty: $R(x, y) = r_\theta(x, y) - \beta \log \frac{\pi_{\text{PPO}}(y|x)}{\pi_{\text{SFT}}(y|x)}$ to avoid the optimized policy going too far away from the original policy that has high reward but low text generation quality.

Apart from the KL divergence penalty, we further use the PPO-Clip objective to update our policy to maximize the reward but constrain our policy not going too far away. The objective is calculated as follows:

$$\mathcal{L}_{\text{PPO}}(\theta) = -\min(\frac{\pi_{\text{PPO}}(y|x)}{\pi_{\text{SFT}}(y|x)}A, g(\epsilon, A)), \quad (3)$$

where

$$g(\epsilon, A) = \begin{cases} (1+\epsilon)A & A \geq 0 \\ (1-\epsilon)A & A < 0 \end{cases}, \quad (4)$$

and A is the advantage, which can be viewed as a reward value minus the average of rewards: $A(x, y) := R(x, y) - R_{\text{mean}}$. To calculate the average rewards, we apply an exponential smoothing that for each iteration, $R_{\text{mean}} := \lambda R + (1-\lambda)R_{\text{mean}}$, where λ is the smoothing ratio set to 0.5.

Since we have a limited budget for GPU memory, we only do an online update to each example. We train our model for 2000 episodes and do a validation per 200 episodes. We reuse 2000 examples with tokenized lengths less than 512, and we set the max context length of the PPO model to 512 in order to reduce the memory load. We set the KL coefficient β to 0.5, the clip ratio ϵ to 0.2, and the smoothing ratio λ to 0.5. We use the Adam optimizer with a learning rate of `9e-6`.

4 Evaluation

4.1 Evaluation for Reward Models

For the reward model evaluation part, we mainly do two kinds of evaluation analysis on the test dataset. First, we calculate the correlation and corresponding p-value between the scores given by the reward models and the confidence in the annotator's work. Second, we define an evaluation metric for computing accuracy and test it with different reward models.

Correlation Analysis Figure 2 illustrates the relationship between scores and confidence using the reward model trained with augmented ground truth examples. Despite that our model has learned some biased patterns illustrated in Appendix C.1, our observations indicate that our model has successfully acquired knowledge. Initially, for the pretrained reward model, the scatter points exhibit a random distribution pattern. However, after training, the scatter points demonstrate a clear upward trend: the higher the confidence, the higher the score. Additionally, the correlation coefficient increases significantly from an initial value close to zero to 0.09 and 0.17 with high confidence (p-value < 0.05). Especially our fine-tuned model augmented with ground-truth examples significantly outperforms the model only fine-tuned with original examples by having a larger correlation coefficient ($0.17 > 0.09$) and a lower p-value ($0.00 < 0.05$). It becomes apparent that our reward model trained with data augmentation attains a more advanced ability in distinguishing good and bad interactions.

Accuracy Metric Table 1 presents the accuracy results for three different models: a pretrained model, a reward model fine-tuned with the original data, and a reward model fine-tuned with both the original and augmented ground-truth data. We assess the models' accuracy based on their ability to distinguish high-quality interactions (associated with higher confidence scores) from low-quality ones (linked to lower confidence scores).

	GPT-2-base	w. EDD	w. aug-EDD
Acc (%)	50.7	60.2	**65.3**

Table 1: Comparison of accuracy. GPT-2-base is our non-finetuned model. w. EDD denotes the model fine-tuned with our Educational Dialogue Dataset, and the prefix "aug-" implies further augmentation with ground-truth examples.

The results clearly illustrate that our fine-tuned models significantly outperform the baseline GPT-2-base model, demonstrating accuracy improvements of 9.5% and 14.6%, respectively. Furthermore, our reward model, which was fine-tuned using augmented ground-truth examples, displayed superior performance, achieving an accuracy of 65.3% compared to the 60.2% accuracy of the model fine-tuned solely with the EDD. Based on these results, we conclude that the reward model fine-tuned with positive data augmentation represents the optimal choice for simulating human feedback in our final assistant model. A detailed analysis of these findings can be found in Appendix C.2.

4.2 Evaluation for Educational Chatbots

We compare performance of four models:

1. **NFT**: Non-fine tuned `GPT-2-medium` (baseline)
2. **SFT I**: `GPT-2-medium` after supervised fine tuning with the original dataset
3. **SFT II**: `GPT-2-medium` after supervised fine tuning with the augmented dataset

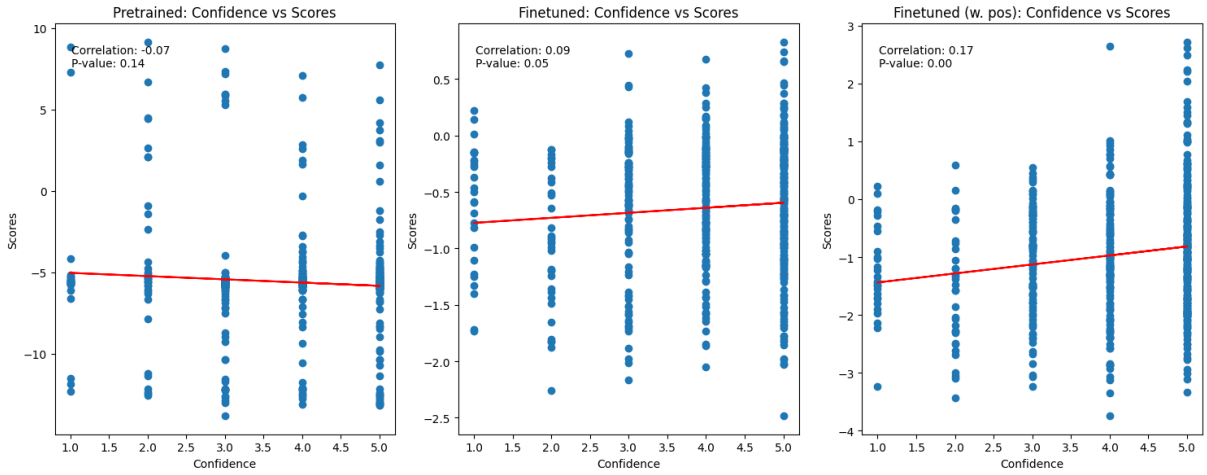

Figure 2: Model comparison of correlation between confidence scores (Confidence) and reward values (Scores) on the test set (without positive generated examples).

4. **PPO I**: `GPT-2-medium` after supervised fine tuning with the augmented dataset and secondary fine tuning with PPO policy[4]

First, we perform quantitative evaluation using syntactic similarity metrics (BLEU and ROUGE) and scores output by the reward model.

Then, we give some qualitative analysis of individual answers produced by our models.

Syntactic Similarity Similarly to what's discussed in Section 2.4, we generate augmented ground-truth examples using ChatGPT based on the answers and explanation provided in the test dataset.

We perform syntactic similarity evaluation using two metrics: BLEU (Papineni et al., 2002) & ROUGE (Lin, 2004). For each model, we calculate the average similarity between the answers it produced and the ground-truth answers.

Results of this evaluation are presented in Table 2. We notice that for both BLEU and ROUGE, SFT provides a significant improvement compared to the non-fine tuned model. Moreover, the model trained on augmented dataset performs better than the model trained only on the original dataset.

Although syntactic similarity scores do not necessarily correspond to semantic similarity, the consistent improvements of scores for both metrics indicate a positive impact of dataset augmentation and fine tuning on the question-answering performance. However, the similarity scores do not show a definitive improvement from the PPO fine-tuning: it performs worse than SFT in terms of BLEU score, but it performs the best in terms of ROUGE score.

Evaluation with the Reward Model Second evaluation strategy we employ is evaluation using the trained reward model. As shown previously, the scores output by the reward model have a positive correlation with human produced confidence scores, so they can be provide a valuable estimate of the quality of the answers produced by our models.

The results are presented in Table 2. We don't notice a significant difference between SFT-I and NFT, but SFT-II obtains a significantly better average score, indicating a positive effect of dataset augmentation on performance. PPO achieves the highest average reward score. However, one should keep in mind that the reward model was used in PPO's training procedure, so it cannot be considered as an unbiased performance metric for PPO.

Qualitative Evaluation We manually inspect some of the generated answers to observe the improvements that our models make over the non fine-tuned `gpt-2-medium` and the limitations we are still facing.

The non-fine-tuned `gpt-2-medium` model often fails completely to answer a question. For example for question with `guid=2fabe332-48ef-433b-9b1c-b4e486180637` (a multiple choice question about matrices), it outputs *"If you have any questions, please feel free to contact me."*, which is a non-answer and is completely unrelated to the question. SFT-I, SFT-II and PPO all do better - they build an answer that isrelated to the question and at least attempt at picking the correct

[4]Practically, we use the model fine-tuned after 200 episodes because we observe our model overfitting with high reward and poor text quality after a long training period.

	BLEU	ROUGE	Reward
NFT	.0013	.133	-0.67
SFT I (orig. data)	.0046	.174	-0.71
SFT II (aug. data)	**.0121**	.260	-0.09
PPO	.0079	**.294**	**0.15**[a]

Table 2: Model performance comparison: average BLEU and ROUGE syntactic similarity scores (compared against augmented ground truth) and average reward score.

[a] Reward model is used in PPO training process, so it can be a biased estimate of model performance in this case.

answers, for example SFT-II replies with: *"Choice (a), (b), (d), and(g)) are incorrect because they do not take into account the fact that $|D|$ must be invertible, which is not the case in this case."*.

One issue we observe with the models is that it they have a bias towards choosing the first answer - they repeatedly claim that "Option (A)" or "Option (1)" is the correct one. Interestingly, this bias is not present in the answers produced by the original non-fine tuned `gpt-2-medium` model, but it appears after fine tuning and is particularly strong for the PPO model.

Our general observation is that while their produced answers are often not correct on a conceptual level, all of our fine tuned models do significantly better at understanding what is expected from an educational assistant and when asked a multiple choice question they try to generate an answer along with a justification. This is not always the case for the non-fine-tuned model.

5 Related Work

5.1 Training Language Models to Follow Instructions

Our work focuses on the research of cross task generalization in language models. In this context, language models are fine-tuned using a wide variety of public NLP datasets, typically accompanied by specific instructions, and subsequently evaluated on different NLP tasks. This area of study has seen various approaches that vary in terms of training and evaluation data (Yi et al., 2019; Mishra et al., 2021), instruction formats (Wei et al., 2021), pretrained model sizes (Aribandi et al., 2021), and other experimental details. A recurring observation in these studies is that fine-tuning language models on diverse NLP tasks, while utilizing instructions, enhances their performance on unseen tasks, both in zero-shot and few-shot scenarios.

5.2 Learning from Human Feedback

We build upon existing techniques that aim to align models with human intentions, specifically focusing on reinforcement learning from human feedback (RLHF). Originally developed for training simple robots in simulated environments and Atari games (Christiano et al., 2017), RLHF has more recently been employed for fine-tuning language models in text summarization tasks (Wu et al., 2021), dialogue task (Yi et al., 2019), translation task (Kreutzer et al., 2018) and story generation task (Zhou and Xu, 2020).

5.3 Modifying the Behavior of Language Models

A goal of modifying the behavior of language models is to mitigate the harms of these models when they're deployed in the real world (Tamkin et al., 2021). These risks have been extensively documented. There are many ways to change the generation behavior of language models, such as using a small and value-targeted dataset to fine-tune LMs (Solaiman and Dennison, 2021) which can improve the models' ability to adhere to these values on a question-answering task, or using word embedding regularization for the LMs (Huang et al., 2019) in order to reduce the generated bias.

6 Conclusion

In this work, we utilize knowledge distillation to train Mini-GPTA, a small yet intelligent chatbot focused on answering EPFL course content queries. We interact with ChatGPT to acquire knowledge for our chatbot. We then use instruction tuning and RLHF to compress ChatGPT's knowledge into a much smaller model, GPT-2-medium. The results show that Mini-GPTA outperforms multiple baselines, and our reward model exhibits a substantial accuracy improvement. We also qualitatively analyze our model's generated responses, discussing its potential and limitations in an educational setting. Our study proves the feasibility of training a smaller, task-specific AI model by leveraging larger language models.

References

Vamsi Aribandi, Yi Tay, Tal Schuster, Jinfeng Rao, Huaixiu Steven Zheng, Sanket Vaibhav Mehta, Honglei Zhuang, Vinh Q Tran, Dara Bahri, Jianmo

Ni, et al. 2021. Ext5: Towards extreme multi-task scaling for transfer learning. *arXiv preprint arXiv:2111.10952*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30.

Imre Csiszár. 1975. I-divergence geometry of probability distributions and minimization problems. *The annals of probability*, pages 146–158.

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

Jaimie Drozdal, Albert Chang, Will Fahey, Nikhilas Murthy, Lehar Mogilisetty, Jody Sunray, Curtis Powell, and Hui Su. 2021. The design and evaluation of a chatbot for human resources. In *HCI International 2021-Late Breaking Posters: 23rd HCI International Conference, HCII 2021, Virtual Event, July 24–29, 2021, Proceedings, Part I*, pages 239–248. Springer.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the MATH dataset. *arXiv preprint arXiv:2103.03874*.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Guruswami Hiremath, Aishwarya Hajare, Priyanka Bhosale, Rasika Nanaware, and KS Wagh. 2018. Chatbot for education system. *International Journal of Advance Research, Ideas and Innovations in Technology*, 4(3):37–43.

Po-Sen Huang, Huan Zhang, Ray Jiang, Robert Stanforth, Johannes Welbl, Jack Rae, Vishal Maini, Dani Yogatama, and Pushmeet Kohli. 2019. Reducing sentiment bias in language models via counterfactual evaluation. *arXiv preprint arXiv:1911.03064*.

Julia Kreutzer, Shahram Khadivi, Evgeny Matusov, and Stefan Riezler. 2018. Can neural machine translation be improved with user feedback? *arXiv preprint arXiv:1804.05958*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Conference on Empirical Methods in Natural Language Processing*.

Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. 2021. Natural instructions: Benchmarking generalization to new tasks from natural language instructions. *arXiv preprint arXiv:2104.08773*.

Tom Nadarzynski, Oliver Miles, Aimee Cowie, and Damien Ridge. 2019. Acceptability of artificial intelligence (ai)-led chatbot services in healthcare: A mixed-methods study. *Digital health*, 5:2055207619871808.

Takuma Okuda and Sanae Shoda. 2018. Ai-based chatbot service for financial industry. *Fujitsu Scientific and Technical Journal*, 54(2):4–8.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Irene Solaiman and Christy Dennison. 2021. Process for adapting language models to society (palms) with values-targeted datasets. *Advances in Neural Information Processing Systems*, 34:5861–5873.

Alex Tamkin, Miles Brundage, Jack Clark, and Deep Ganguli. 2021. Understanding the capabilities, limitations, and societal impact of large language models. *arXiv preprint arXiv:2102.02503*.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. 2023. Alpaca: A strong, replicable instruction-following model. *Stanford Center for Research on Foundation Models. https://crfm. stanford. edu/2023/03/13/alpaca. html*, 3(6):7.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2021. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Jeff Wu, Long Ouyang, Daniel M Ziegler, Nisan Stiennon, Ryan Lowe, Jan Leike, and Paul Christiano. 2021. Recursively summarizing books with human feedback. *arXiv preprint arXiv:2109.10862*.

Anbang Xu, Zhe Liu, Yufan Guo, Vibha Sinha, and Rama Akkiraju. 2017. A new chatbot for customer service on social media. In *Proceedings of the 2017 CHI conference on human factors in computing systems*, pages 3506–3510.

Sanghyun Yi, Rahul Goel, Chandra Khatri, Tagyoung Chung, Behnam Hedayatnia, Anu Venkatesh, Raefer Gabriel, and Dilek Hakkani-Tür. 2019. Towards coherent and engaging spoken dialog response generation using automatic conversation evaluators. *arXiv preprint arXiv:1904.13015*.

Wangchunshu Zhou and Ke Xu. 2020. Learning to compare for better training and evaluation of open domain natural language generation models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 9717–9724.

A Contribution

Our teammates make roughly equal contributions to the project. Here is a brief summary of what each team member did for the project:

- Yiyang: demonstration data collection, reward model training and evaluation scripts, external data augmentation for the SFT and PPO models, PPO model training scripts, and the report writing of abstract, introduction, conclusion, external data augmentation, reward modeling, and PPO modeling.
- Haotian: demonstration data collection, reward model validation script, analyzing the effect of reward model, SFT model training and validation scripts, and the report writing of data source, supervised fine-tuning, evaluation for reward models and the related work.
- Maciej: demonstration data collection, researching data external data sources, dataset exploratory analysis and preparation, dataset augmentation: generating augmented ground-truth examples using ChatGPT, scripts for prediction generation, final evaluation (qualitative and quantitative) of the chat assistant, report writing - mostly the Dataset and Evaluation parts

B Detailed Data Processing

B.1 Data Distribution

We show the distribution of question domain and difficulty level of selected MATH dataset for our training in Figure 3 and Figure 4.

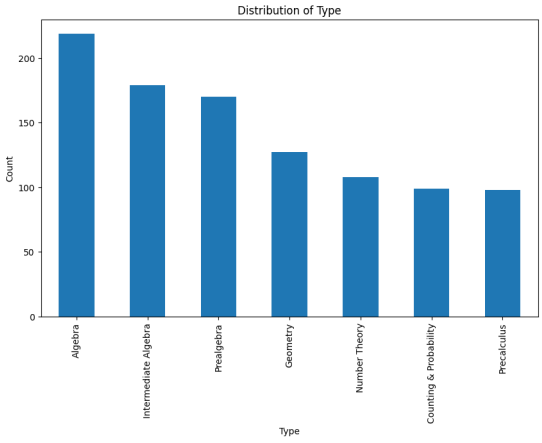

Figure 3: Distribution of question domains.

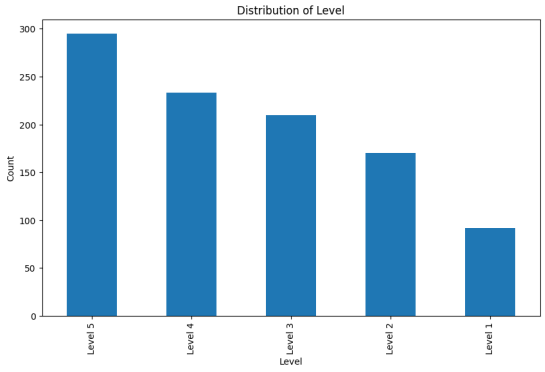

Figure 4: Distribution of question difficulty.

B.2 Negative Data Augmentation

For the negative data generation, the overall approach is similar to positive examples, but we instruct ChatGPT to select an actual incorrect answer as the "correct" answer and then generate a justification for this "correct" answer. During our experiments, we observed that ChatGPT often begins its response with the phrase "Incorrect answer:" to indicate that the answer is incorrect. To address this, we replace the word *Incorrect* with *Correct* and the term *incorrect* with *correct*.

This way, we will obtain more elaborate positive and negative examples, more resemble to the actual interactions conducted by students.

Examples of this process can be found in the `data_preparation.ipynb` notebook on our GitHub.

C Detailed Evaluation

C.1 Reward Model Evaluation of Correlation

This section provides a detailed evaluation of different reward models. We augment our Educational Dialogue Data with negative examples in the way illustrated in Appendix B.2. Figure 5 illustrates the relationship between scores and confidence using the reward model trained with augmented examples. In Figure 6, the same relationship is depicted for the reward model trained without augmented examples. Our observations indicate that our model has successfully acquired knowledge. Initially, for the pretrained reward model, the scatter points exhibit a random distribution pattern. However, after training, the scatter points demonstrate a clear upward trend: the higher the confidence, the higher the score. Additionally, the correlation coefficient increases significantly from an initial value close to zero.

It is worth noting that in the lower right corner of Figure 6, the correlation coefficient of the scatter points is negative. This arises because our model was exclusively trained on the training set with positive data augmentation. However, the test set comprises both positive and negative examples, leading the model to develop a bias that assigns higher scores to the generated examples.

Furthermore, the bottom two images in Figure 6 indicate that the inclusion of negative examples partially mitigates the bias but does not entirely eliminate it in the comparison between original and augmented examples. Comparing Figure 5 and Figure 6, it becomes apparent that the reward model trained with data augmentation attains a more advanced ability, particularly in distinguishing between positive and negative instances.

C.2 Reward Model Evaluation of Accuracy

Algorithm 1 shows how we use an accuracy computation algorithm to quantitatively evaluate the performance of the models. We utilize this algorithm to assess three fine-tuned models and one pretrained model. These findings are presented in Figure 7.

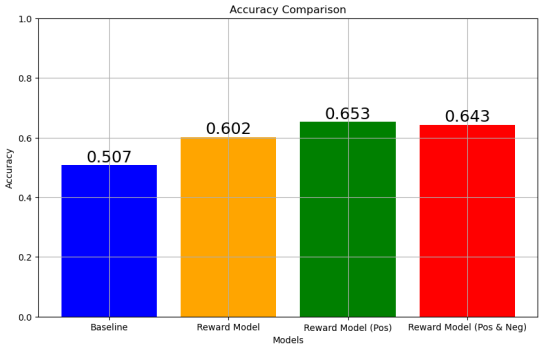

Figure 7: Accuracy comparison result

Among the four models' performance in Figure 7, the reward model trained with additional positive examples demonstrates the highest accuracy of 65.3%. This is followed by the reward model trained with additional positive and negative examples, which achieves an accuracy of 64.3%. The accuracy of reward model with original data is just 60.2%, lower than any data augmentation methods. In comparison, the baseline GPT-2-base model only achieves an accuracy of 50.7%.

These accuracy values indicate that all reward models outperform the baseline model in terms of accuracy, and the one with positive data augmentation performs 9% better than one without any data augmentation, which also proves that the original dataset with data augmentation can produce a better performance reward model. We think the reason why the performance of reward model only with positive data augmentation is better than one with positive and negative data augmentation is that maybe the negative examples are not necessarily worse than some of the student examples rated with a higher confidence score, but the positive examples are actually better than student examples. So only positive data augmentation makes the quality of training set higher.

Algorithm 1 Computing Accuracy

```
Input: Groups with sol_id, scores, and confidence in the dataset
Output: Accuracy
Initialize correctNumber = 0
Initialize totalComparison = 0
for each group with the same sol_id do
    Initialize isConsistent = true
    for each pair of scores and confidence within the group do
        if rank relationship between scores and confidence is inconsistent then
            isConsistent = false
            break
        end if
    end for
    if isConsistent then
        correctNumber = correctNumber + 1
    end if
    totalComparison = totalComparison + 1
end for
Accuracy = correctNumber / totalComparison
return Accuracy
```

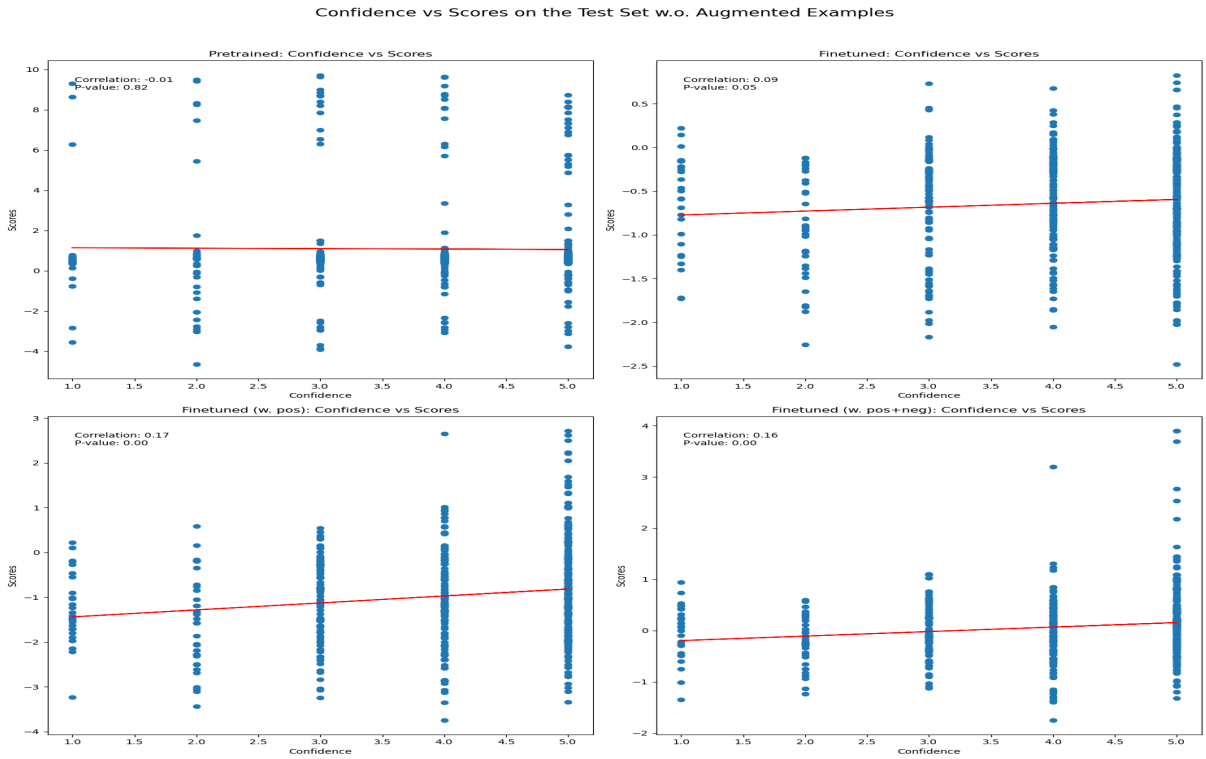

Figure 5: Confidence vs Scores on the test set without positive generated examples.

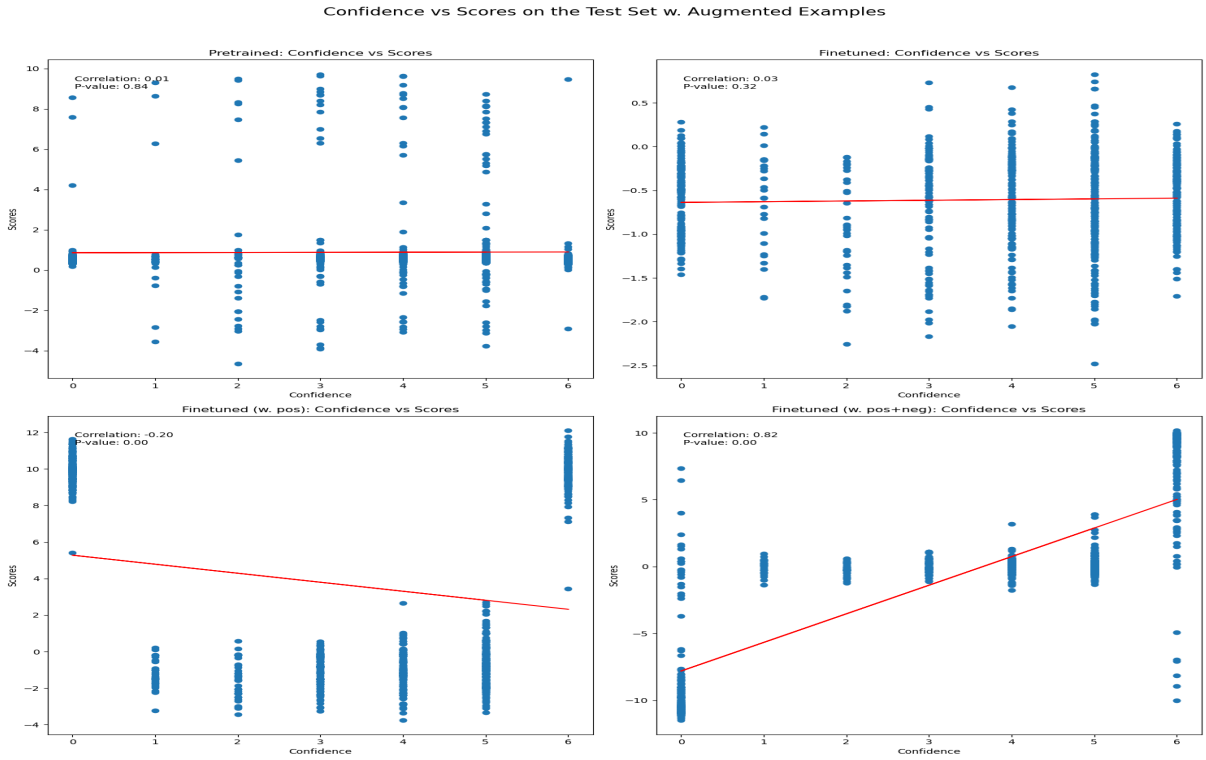

Figure 6: Confidence vs Scores on the test set with positive generated examples.