ChatMGL: A Large Language Model Fine-tuned for Data Science Questions

Manos Chatzakis
emmanouil.chatzakis@epfl.ch
EPFL
Lausanne, Switzerland

Ioannis Bantzis
ioannis.bantzis@epfl.ch
EPFL
Lausanne, Switzerland

Lluka Stojollari
lluka.stojollari@epfl.ch
EPFL
Lausanne, Switzerland

ABSTRACT

This paper presents ChatMGL, a Large Language Model trained with supervised fine-tuning and Reinforcement Learning through Proximal Policy Optimization to answer Data Science and STEM questions. ChatMGL addresses several challenges in the field of Natural Language Processing related to data collection, as well as model training and evaluation. Through a wide range of experimental settings, we present that our model achieves satisfactory performance on various data collections, and we show that ChatMGL is capable of providing high-quality responses for prompts that contain Data Science questions.

KEYWORDS

Large Language Models, Proximal Policy Optimization, Supervised Fine-tuning, Natural Language Processing, Reinforcement Learning, Deep Learning

1 INTRODUCTION

Motivation. The field of Natural Language Processing (NLP) has seen remarkable growth over recent years ([28], [18], [7], [12]), as the emergence of Large Language Models (LLMs), in cooperation with the advanced computational capabilities of modern hardware, managed to achieve impressive results for a wide variety of deep learning tasks [10], including text summarization, language translation, text classification, and text generation ([13], [24], [35], [20], [8]). Specifically, since the emergence of various general-purpose NLP models, especially ChatGPT [21], LLMs seem to be used widely for providing answers to questions related to STEM fields and especially in Data Science ([32], [19]).

Challenge. In the context of using NLP for Data Science questions, which includes questions from various fields of Science, Technology, Engineering, and Mathematics (STEM), general-purpose LLMs have been improving the quality of their responses. However, they still lack the ability to provide satisfactory answers for harder and field-specific questions that require deeper knowledge, understanding, and reasoning ([27], [17], [29]).

A crucial reason for this phenomenon is the lack of relevant contextualized datasets that contain examples of appropriate questions and answers for the corresponding fields, especially when it comes to university courses. In this work, we present ChatMGL, a Large Language Model fine-tuned using supervised and reinforcement learning with Human Feedback (RLHF) [4] through Proximal Policy Optimization (PPO) [25], with a focus on Data Science and STEM questions, with data gathered partially from questions of EPFL courses.

Approach. Our implementation for ChatMGL starts by using the pre-trained GPT-2 model for text generation [23], and we fine-tune the task. ChatMGL is trained using supervised fine-tuning and Reinforcement Learning with Human Feedback (RLHF). For the supervised training part, we train the model by providing question-answer demonstrations in a supervised manner, while for RLHF we implement Proximal Policy Optimization (PPO) and a reward model. Our reward model is a fine-tuned binary classifier based on pre-trained BERT [5] that classifies responses generated by ChatMGL. Our dataset is a union of samples gathered from multiple sources of information, containing samples from openly available datasets of STEM questions, as well as custom-generated samples from questions reported by courses that are taught at EPFL.

Contribution. We summarize our main contribution as follows

- We describe ChatMGL, an LLM optimized for Data Science questions through supervised fine-tuning and Reinforcement Learning from Human Feedback
- We present a binary classification reward model able to categorize LLM-generated responses based on their quality
- We present a dataset containing Data Science and STEM questions from multiple sources consisting of openly available datasets and custom generations through GPTWrapper [1] [2]
- We conduct a wide experimental evaluation over a range of different metrics and settings, that demonstrate the ability of ChatMGL to provide high-quality responses for Data Science questions of different types

Roadmap. The rest of this work is organized as follows: Section 1 is the introduction, section 2 presents the related work, section 3 shows the related background and preliminary material, section 4 presents our dataset, section 5 explains our reward model, section 6 presents our generative model, section 7 is the experimental evaluation ,and section 8 is the conclusion.

2 RELATED WORK

In this section, we present the related work of our research. The common direction of this related work is to enable language models to answer correctly questions that require complex reasoning and understanding of a topic.

Summarization from Human Feedback. This work [26] shows that it is possible to significantly improve summary quality by training a model to optimize for human preferences. By collecting a large dataset of human comparisons between summaries, a GPT3-style [3] model is trained to predict the human-preferred summary, and that model is used as a reward function to fine-tune a summarization policy using reinforcement learning.

[1] GPTWrapper is a ChatGPT API provided by EPFL
[2] GPTWrapper and ChatGPT are used interchangeably in this work

Chain-of-Thought Prompting. This work [31] introduces a novel method to prompt LLMs to improve their performance in scenarios where the questions require arithmetic, commonsense or symbolic reasoning. It works by providing a question similar to the one that will be asked, along with the answer, presented as a chain of thought. This means that the answer is provided as a series of intermediate steps, demonstrating the reasoning of the solution. The authors conduct an experimental evaluation of their method using different models and datasets and demonstrate that Chain-of-Thought Prompting can increase the performance of LLMs in reasoning tasks.

Self Consistency in Chain of Thought Prompting. This paper [29] presents a novel approach, self-consistency, to address the limitations of language models in logical and commonsense reasoning tasks. Despite the remarkable achievements of language models in natural language processing, their ability to reason and infer remains a challenge. The authors propose self-consistency as a solution to improve the accuracy and robustness of language models in reasoning tasks.

3 BACKGROUND

In this section, we present the related background and preliminary material.

Reward Model. A reward model is a binary classifier that categorizes a model's response based on its quality [9]. It is trained using supervised fine-tuning over a pre-trained classification model by iterating over good and bad example responses to the same question. A reward model is a crucial component of the RLHF training loop and for the evaluation of a final model that generates text.

Generative Model. A generative language model is a Large Language Model trained in the task of text generation [15]. A generative model is provided with a question and responds with an answer which is based on the demonstrations it was shown during the training.

Proximal Policy Optimization. Proximal Policy Optimization (PPO, [25], [30]) is a training policy for Reinforcement Learning with Human Feedback [4]. Through this method, a model is trained with data demonstrations by comparing its response to a baseline model and the answer of the reward model.

BERT. BERT [5] is an NLP model with bidirectional context and transformer architecture. It is pre-trained on unlabeled data and fine-tuned on specific tasks, showing impressive performance on various NLP tasks. In our work, we base our reward model on a fine-tuned BERT instance for Sequence Classification.

GPT-2. GPT-2 [23] is a transformer-based LLM pre-trained on a vast dataset that has remarkable language generation capabilities. It can also be fine-tuned for various tasks, and we use it both for our reward model, using a GPT-2 instance for Sequence Classification, and for our generative model, using a GPT-2 instance for Conditional Generation.

BART. BART [14] employs a denoising autoencoder architecture, a bidirectional encoder, and an auto-regressive decoder. It is pre-trained with a huge amount of data and can be fine-tuned for text-generation tasks.

Cross Entropy Loss. Cross Entropy Loss [34] is a loss function that measures the dissimilarity between the predicted probabilities of a model and the true labels of the data. It is widely used for text generation in NLP, where models calculate the probability of the next token given a sequence, as it penalizes the model more when it makes confident incorrect predictions and rewards it when it makes correct predictions with high confidence.

4 DATASET

This section presents our dataset. We present our data sources, we show our data preparation techniques, and we explain our bias handling.

4.1 Data Sources

We use various data sources containing questions for coding, maths, physics, and other Data Science and STEM subjects, by harvesting data from different datasets and generating data from ChatGPT for EPFL courses questions. Our sources are summarized in table 1, along with a brief description and the total number of samples we have available.

Dataset	Description	Total Samples
MATH	Math questions-answers	12500
MathQA	Multiple choice STEM questions-answers	400
StackExchange	Data Science and Coding questions-answers	4000
Interactions	ChatGPT responses for Data Science questions	1762

Table 1: ChatMGL Datasets.

MATH Dataset. MATH Dataset [11] includes STEM questions with formally written answers, mainly for math fields, such as algebra, number theory, probabilities, and geometry.

MathQA Dataset. MathQA [1] is a dataset that includes data from various fields of science, mainly from mathematics to physics. The questions are in a multiple-choice format, and a minimal, human-written, informal explanation is sometimes provided for the correct choice.

StackExchange Dataset. StackExchange dataset is an anonymized dump of all user-contributed content on the Stack Exchange network. It contains a vast amount of high-quality questions and answers as well as the votes for each post and the accepted answer by the questioner, gathered using the method described in [2].

ChatGPT Interactions Dataset. This dataset contains user interactions with ChatGPT [21] about Data Science questions of courses taught at EPFL. Each ChatGPT response is associated with a human-assigned score, namely confidence level, related to the quality and correctness of the response.

4.2 Reward Model Data

Description. Our reward model is trained using demonstrations of positive and negative responses for specific answers. The data used for this task consisted of samples from MathQA, ChatGPT Interactions, and StackExchange. For the Interactions, we created pairs of good and bad responses by selecting questions with high and low human-assigned scores, as questions with high scores are likely to serve as correct demonstrations. For StackExchange, we created pairs of good and bad responses by gathering answers with many positive votes and answers with many negative votes.

For MathQA we created examples of bad responses by randomly selecting a wrong answer from the other choices of the dataset, and we asked ChatGPT to provide an answer with reasoning that leads to this faulty answer. This method worked impressively well in most samples, and thus we managed to generate various pairs of good and bad responses from those data. The data used to train, validate and test the reward model are summarized in table 2 [3].

Data Format. We manipulate our reward model data using JSON, and each sample has the following format:

```
{
    "entry_id": int,
    "chosen": string,
    "rejected": string
}
```

Each sample is associated with a unique entry id, while the chosen and rejected fields correspond to the good and bad response examples. The chosen and rejected fields contain both the question of the response. In addition, during our training for the reward model, we expand this format so that each entry corresponds to a good or a bad response example, with the corresponding label. This means that the classification models process the data in the following, expanded format:

```
{
    "entry_id": int,
    "chat": string,
    "label": {"positive", "negative"}
}
```

The chat field represents the interaction with the question and answer, while the label indicates if the response inside the chat field corresponds to a positive (good) or negative (bad) response example.

Dataset	Train (80%)	Validation (10%)	Test (10%)
MathQA	320	40	40
StackExchange	1600	200	200
Interactions	1657	208	207
Final	**3577**	**448**	**447**

Table 2: Reward Model Data.

4.3 Generative Model Data

Description. We used samples of data from our different data sources for both supervised learning and RLHF. Table 3 summarizes all the data we used for our generative model training. After gathering all the samples for the training, validation, and testing, we split the final dataset into two equally sized sub-datasets to facilitate the supervised training and RLHF routines.

Data Format. Our generative model supports the data in the following format:

```
{
    "guid": string,
    "question": string,
    "answer": string,
    "source": string
}
```

Each sample is associated with a unique guid, while the question field is the text of the question that will be used as input for the model. The answer field corresponds to the label of the question, while the source field indicates the dataset that the sample originates from. In addition, we did not use the StackExchange data for this part, because we noticed that their format made the model results worse during training.

Dataset	Train	Validation	Test
MATH	6753	747	5000
MathQA	320	40	40
Interactions	1409	177	176
Final-Supervised (50%)	4241	2607	482
Final-RLHF (50%)	4241	2608	481

Table 3: Generative model data.

4.4 Bias

During our data preparation, we encountered several examples of bias, which we addressed using different techniques, for each one of our data sources.

Bias on Interactions. Regarding the Interactions, we noticed that most correct answers had very short explanations or no explanation at all [4]. while generally, the responses with high confidence levels were shorter than the ones with low confidence levels. This phenomenon introduced a bias over the length of the chosen and rejected answers. We addressed this issue by asking ChatGPT to provide an explanation of the correct solution, providing it with the initial prompt and the final answer. With this method, we managed to get longer correct answers which matched the writing style of the low-confidence interactions (as they are generated by the same language model). In addition, we managed to reduce the gap in the number of tokens that chosen and rejected answers have.

Bias on MathQA. Regarding our samples from MathQA, we noticed that the correct answers, which are human-generated without a specific writing style, are very different from the rejected answers, which, (as we described above) are generated by ChatGPT. The first major difference is the writing style, as the reasoning provided by the MathQA dataset follows a very basic type of writing. In contrast, the generated wrong reasoning from ChatGPT is better in terms of syntax and language quality. For this reason, we used ChatGPT to paraphrase the correct reasoning of MathQA, in order to get answers with the same writing style. The second major difference was that the correct answers were a lot shorter. We managed to

[3] We refer to good and bad examples as chosen and rejected examples interchangeably

[4] We had some correct explanations provided by the EPFL courses

reduce this difference with paraphrasing, but a minor difference in the length of the answers remains.

Overall, the two main bias incidents we noticed are the difference in the number of tokens between bad and good responses (for the reward model) and the difference in the writing style. We managed to address both issues, as figure 1 shows the difference in the number of tokens between good and bad responses for the reward data, for the data sources, and for the final reward dataset. We see that after our handling, the reward dataset contains samples that have similar response lengths for good and bad responses.

5 REWARD MODEL

In this section, we present our reward model. In order to select the best reward model, we tried training for both regression and classification. We experimented with both choices and different pre-trained models, in order to select the best-performing model based on our experimental evaluation of section 7.

5.1 Regression

For regression, our objective was to develop a reward model following the guidelines and instructions provided in the Instruct-GPT paper [21]. We began by employing the GPT-2 pre-trained model as a starting point, which served as our baseline. To enhance its capabilities, we incorporated an additional linear layer that accepted the *hidden_size* of the GPT-2 model's last hidden state as input and generated a scalar output. Our reward model was designed to process pairs of prompts and responses, regardless of whether they were chosen or rejected. We should note that for our regression model, instead of expanding the dataset as we described in section 4, we use the initial dataset format. These responses were fed into the model, which then produced a scalar reward as output. The aim was to leverage this reward signal for training and later reinforcement learning purposes.

Then, for each training example consisting of a positive (chosen) response and a negative (rejected) response, we computed the corresponding reward scores using the model. We denoted the reward score for the winning response as $S_w = r\theta(x, y_w)$ and the reward score for the losing response as $s_l = r\theta(x, y_l)$, where θ corresponding to the respective weights. Subsequently, we utilized these reward scores to calculate the loss, which was defined as $loss = -log(\sigma(s_w - s_l))$. The goal of the training was to find the optimal θ values that minimize the expected loss across all training samples.

5.2 Classification

Our classification model works with the expanded input described in section 4. Given an input containing a human prompt and the model's response, it outputs whether the answer is positive or negative (good or bad) in terms of quality.

Selected pre-trained Models. To train our model, we fine-tuned two pre-trained models for sequence classification, namely DistilBERT, which is a lightweight version of BERT, and GPT2.

Chunking. In order to provide input sentences to our models, we need to tokenize and encode every input sentence to create an input sequence of encoded tokens (input IDs). However, we noticed that sometimes the tokens of the input exceeded the max sequence length of the corresponding model. In order to support sequences of arbitrary length, we implemented a chunking method that splits the tensors of input IDs into a number of chunks based on the max sequence length (*MAX_SEQ_LEN*) supported by each model. Figure 2 presents an overview of the process. After the chunking, each tensor chunk is passed to the model separately and we retrieve the logits [5]. To obtain the final logits and compute the loss, we average the partial logits of each chunk, and we calculate the loss. Using this method, a prediction is generated by applying the *argmax* function to the mean-logits tensor. Unfortunately, this method increased the training time without significantly improving the results, especially for GPT-2 which requires longer training times, thus, we decided to proceed with the classic truncation method to the max sequence length supported by each model.

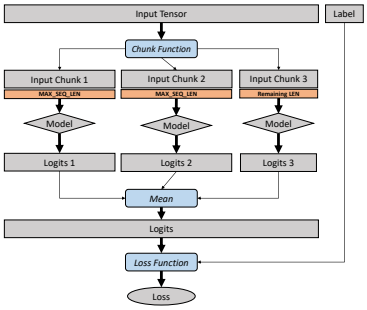


Figure 2: Chunking technique to support input sequences of arbitrary size in the reward model

6 GENERATIVE MODEL

In this section, we present the generative model that forms the core of ChatMGL, emphasizing its significance in the context of our work. Our final training model incorporates the GPT2 model as the principal generative model, serving as the baseline within the RLHF (Reinforcement Learning with Human Feedback) loop.

6.1 Supervised Training

To fine-tune the pre-trained model, we applied supervised learning techniques, with a specific focus on text conditional generation. This approach involved leveraging prior knowledge of the answer to a given question, enabling us to fine-tune the model for generating responses conditioned on this answer. The foundation of this process relied on a comprehensive dataset encompassing an extensive range of textual content.

BART Supervised Fine-tuning To facilitate fine-tuning using supervised learning, we first employed Facebook's BART (bart-base) model. The input structure for the BART model was defined by the input_ids, representing a sequence constructed by the question's tokenizer, augmented with the appropriate start and end tokens. The same methodology was applied to the answers. Subsequently, we created a custom model that wrapped the BART(*BartForConditionalGeneration*) model in order to perform the

[5]Logits are the unnormalized output of a model before it is transformed into a probability distribution

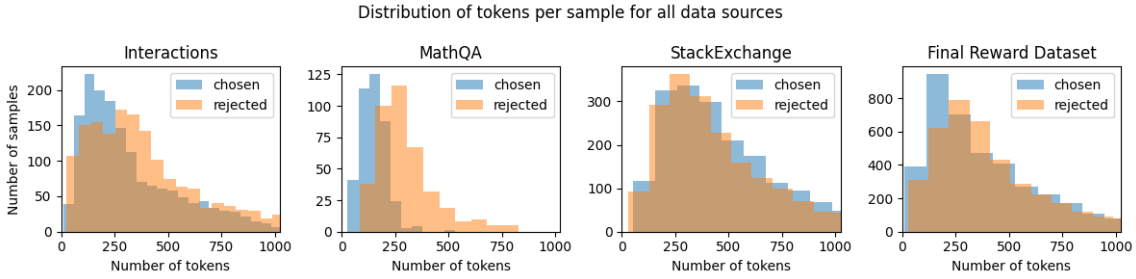


Figure 1: Token distribution for the reward model data. Chosen and Rejected labels represent the good and bad example responses for the reward data, respectively.

fine-tuning process using the forward pass. This entailed feeding the question as input to the model, along with the answer as labels, enabling the computation of cross-entropy loss to measure the discrepancy between the generated answer and the given answer tokens.

GPT2 Supervised Fine-tuning For our final fine-tuned model we employed supervised learning to fine-tune a pre-trained version of the 774M GPT2 model. The input structure for the model was formulated as follows in our approach:

$$input\_ids = <bos\_token> + question + <sep\_token> + answer + <eos\_token>$$

Here, the "<bos_token>" denoted the commencement of the sequence, "question" denoted the given question, "<sep_token>" served as the delimiter separating the question and the answer, "answer" represented the provided answer, and "<eos_token>" indicated the conclusion of the sequence.

During the training phase, the input_ids were supplied to the GPT2 model. Notably, the input_ids were also employed as the *labels* for the model, facilitating the comparison between the generated answer and the ground truth answer. To enhance the learning and fine-tuning process, supplementary techniques were employed. Specifically, we incorporated type_token_ids to furnish additional information to the model, aiding its ability to discern the initiation and termination points of questions and answers in the concatenated sentence. Moreover, to exclude the padding tokens from contributing to loss computation, we substituted the padding token IDs with an ID(-100) absent from the model's vocabulary.

This systematic approach enabled targeted fine-tuning of the model, aligning its capabilities with the specific task at hand. By juxtaposing the generated answer with the ground truth answer, as represented by the input_ids, the loss could be calculated. In detail, the model dynamically shifted the input_ids to generate novel tokens, thereby determining the Cross-Entropy Loss across all generated tokens, and comparing them against the ground truth answer (labels = input_ids). Through the application of this methodology, we effectively fine-tuned the model using the answer corpus, empowering it to generate responses conditioned on the given answer.

6.2 Model Training using Reinforcement Learning with Human Feedback (RLHF)

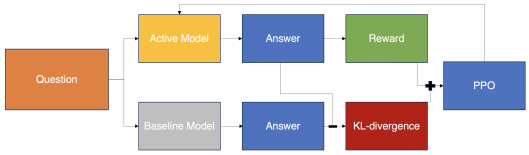


Figure 3: General Training Procedure for PPO

We fine-tune the policy π to optimize the reward model using PPO [6]. At the beginning of the training, both π and π_{base} are initialized as our fine-tuned supervised model. The reward value is selected to be the scaled empirical probability

$$r(x, y) = 2 * (\mathbf{P}(positive|x, y) - 0.5) \in [-1, 1]$$

where the probability above is obtained by the reward model's raw output logits transformed through the softmax function. The above scaling is in line with [36] in which the authors normalize the reward to allow for negative values.

To keep π from moving too far from the initial model baseline $base$, we add a penalty with expectation $\beta \dot{KL}(\pi, base)$. That is, we perform RL on the modified reward

$$R(x, y) = r(x, y) - \beta \log \frac{\pi(y|x)}{\pi_{base}(y|x)}$$

We vary dynamically the constant β to achieve a particular value of KL. This term has several purposes: it plays the role of an entropy bonus, it prevents the policy from moving too far from the range where r is valid, and thus encourages coherence of the responses.

$$e_t = clip\left(\frac{KL(\pi_t, \pi_{base}) - KL_{target}}{KL_{target}}, -0.2, 0.2\right)$$
$$\beta_{t+1} = \beta_t(1 + K_\beta e_t)$$

We use $K_\beta = 0.1$, batch size 4 and we do not use dropout for policy training. The learning rate for our task was $1.41 \times 10 - 5$.

Although we implemented the general training procedure for PPO, we noticed that our model's results did not improve, and thus,

Figure 4: Comparison of reward model performance for the test dataset

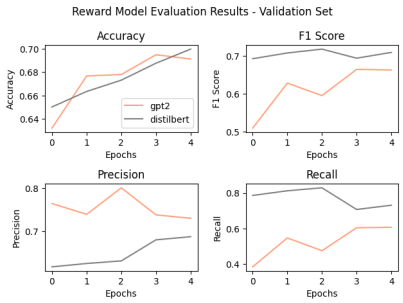


Figure 5: Per-epoch performance of reward models for validation data

we decided to proceed with ChatMGL using only our supervised fine-tuned model.

7 EVALUATION

This section describes the methodology we used to evaluate our fine-tuned GPT-2 model using human feedback. We performed our experimental evaluation in Google Collaboratory using a provided Cuda GPU.

7.1 Reward Model

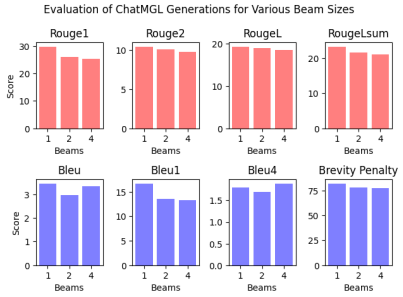
We conducted a range of different experiments to decide which is the best-performing reward model for our task.

Model Selection. We performed a model selection experiment for our reward model in order to compare the results of different pre-trained models after our fine-tuning with the corresponding reward data of table 2. We compared the performance of our Classification models using pre-trained BERT and GPT-2 for sequence classification. Unfortunately, our GPT-2 regression model achieved low accuracy, roughly 60%, thus we omitted its results from our evaluation. For the training of the classification models, we report the results after 5 training epochs using $5e-5$ learning rate and 20% warmup rate. Figure 4 presents the results of the classification models. We report the accuracy, f1-score, precision, and recall for both models. The fine-tuned BERT reward model had better results overall, achieving up to 73% accuracy, while GPT2 had worse results, with 69% accuracy.

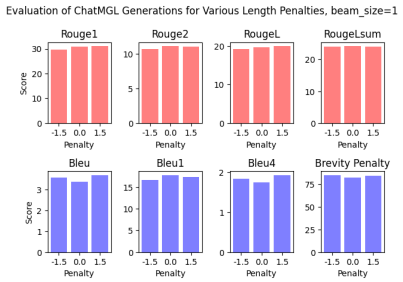
Model Epoch Comparison. Additionally, we report the per epoch results for the validation set of our data for both GPT-2 and BERT models, for 5 epochs in total, in figure 5. We see that for most of the metrics and most epochs, BERT outperforms GPT-2, while most metrics for BERT improve as the epochs advance.

7.2 Generative Model

Beam Size Evaluation. In order to tune the generations of our model, we computed various metrics for different beam sizes for Beam Search decoding. We used a small test sample of 60 samples and we limit the response tokens to 150 tokens (keeping all other parameters to defaults), because of the computational limitations of our resources. The results are presented in figure 6a. We experimented with three values of beam search (1,2,4) and we calculated



(a) Performance of ChatMGL for different beam values



(b) Performance of ChatMGL for different length penalties, beam_size=1

Figure 6: ChatMGL performance for different generation parameters

the ROUGE[16] and BLEU[22] scores for the generations. The results show that beam_size=1 (greedy decoding) leads to the best overall results.

Length Penalty Evaluation. We perform a similar evaluation for length penalty (how much longer sequences are penalized), with beam_size=1 and other parameters defaults, using the 60 test samples and fixed max new generation tokens to 150. The results of this experiment are presented in figure 6b. We see that we get the best scores when the length penalty is set to 1.5.

BERT Scores. After configuring ChatMGL's decoding with beam_size=1 and length_penalty=1.5, we evaluated the generations with BERT scores [33] in order to measure how semantically similar are the

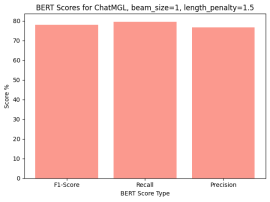
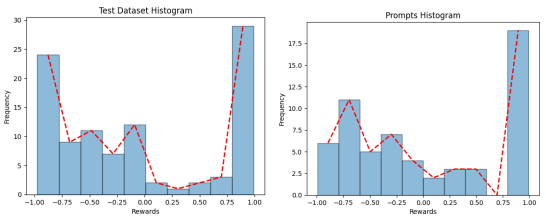


Figure 7: Performance of ChatMGL for BERT Scores, beam_size=1, length_penalty=1.5



(a) Test Dataset **(b) Prompts Dataset**

Figure 8: ChatMGL response evaluation using the Reward model

responses to the actual answers. Figure 7 shows the precision, recall, and f1-score of this evaluation, which are approximately 80%.

Evaluations using the Reward Model. After fine-tuning our model, we proceeded to evaluate its performance using a small testing dataset consisting of 60 question-answer pairs, as well as the prompts file[6]. Utilizing our model, we generated responses and passed both the question and the generation through the reward model, using the same format as the training of the reward model. We then assigned a corresponding reward to each response and plotted them as a histogram. The results are presented in figure **??**. From the histogram, we can interpret that the reward model demonstrates higher confidence for the testing data that closely resembles the trained examples while exhibiting less confidence for the promptjson examples.

7.3 Generations Examples

This section presents examples for two scenarios: one when the generated model produces completely negative or randomly repetitive tokens, and another when it generates more robust and finely tuned examples.

Optimizing against our reward model is supposed to make our policy align with human preferences. But the reward model is only a proxy for human preferences, as it only sees a small amount of comparison data from a narrow distribution of data. While the reward model performs well on the kinds of questions and answers it was trained on, the model starts to produce nonsense eventually.

We observe in figure 8 after the model has been optimized for 95 iterations, even though the answers don't make sense, the reward produced by the output is still positive.

At the beginning of PPO training, the fine-tuned model is still able to discuss and answer challenging questions unsuccessfully but producing a relevant generation. To explain this phenomenon further, we provide a plot of the KL divergence 11 which measures the deviation of the PPO trained model when compared to the fine-tuned baseline model. Increasing the iterations leads to overfit and quality degratation. This was also noted in [26].

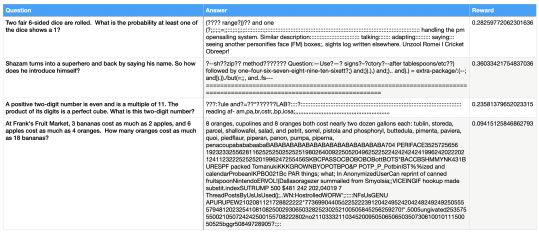


Figure 9: Generations using our fine-tuned model at the 95-th iteration of PPO training

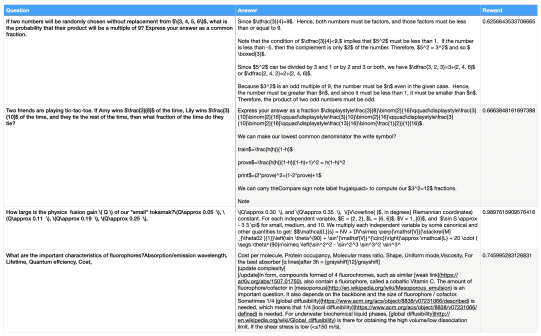


Figure 10: Generations using our fine-tuned model at the 20th iteration of PPO training

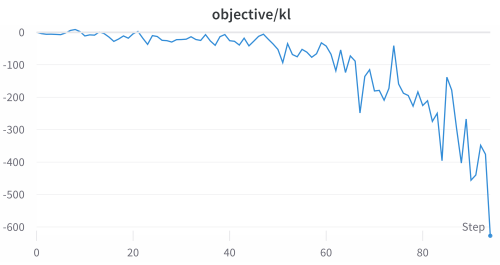


Figure 11: KL-divergence of the PPO trained model compared to fine-tuned per iteration

8 CONCLUSION

In this work, we presented ChatMGL, a Large generative Language Model optimized for Data Science questions. ChatMGL is trained using supervised fine-tuning with various demonstrations from multiple data sources, as well as using Reinforcement Learning with Human Feedback with Proximal Policy Optimization and a

[6]We were provided a sample questions file named prompts.json

fine-tuned reward model. Our experiments evaluate our model with a variety of settings and metrics, showing its ability to generate high-quality answers for multiple Data Science questions.

REMARKS

Supplementary Material. We provide both our reward and generative datasets, Python code, models, and generation scripts in our GitHub repository. Due to the limited file size of GitHub, we provide the final ChatMGL model in the following Google Drive URL.

Authors Contributions. All authors contributed equally in order to complete this work. We summarize the most important tasks for each author in table 4.

Task	Author
Reward Model Data Preparation	Ioannis, Manos
Classification Reward Model Training and Evaluation	Manos
Regression Reward Model Training and Evaluation	Lluka
Reward Model Report	Manos, Lluka, Ioannis
Generative Model Data Preparation	Manos, Ioannis
Supervised Model Training	Lluka, Ioannis
RLHF Model Training with PPO	Ioannis
Model Evaluation Routines and Metrics	Manos, Ioannis, Lluka
Final Paper	Manos, Lluka, Ioannis

Table 4: Author Contributions.

REFERENCES

[1] Aida Amini, Saadia Gabriel, Peter Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. 2019. MathQA: Towards Interpretable Math Word Problem Solving with Operation-Based Formalisms. arXiv:1905.13319 [cs.CL]

[2] Amanda Askell, Yuntao Bai, Anna Chen, Dawn Drain, Deep Ganguli, Tom Henighan, Andy Jones, Nicholas Joseph, Ben Mann, Nova DasSarma, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Jackson Kernion, Kamal Ndousse, Catherine Olsson, Dario Amodei, Tom Brown, Jack Clark, Sam McCandlish, Chris Olah, and Jared Kaplan. 2021. A General Language Assistant as a Laboratory for Alignment. arXiv:2112.00861 [cs.CL]

[3] 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 (2020), 1877–1901.

[4] Oliver Daniels-Koch and Rachel Freedman. 2022. The Expertise Problem: Learning from Specialized Feedback. *arXiv preprint arXiv:2211.06519* (2022).

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:1810.04805 [cs.CL]

[6] Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, Yuhuai Wu, and Peter Zhokhov. 2017. OpenAI Baselines. https://github.com/openai/baselines.

[7] Eghbal Ghazizadeh and Pengxiang Zhu. 2020. A systematic literature review of natural language processing: Current state, challenges and risks. In *Proceedings of the Future Technologies Conference*. Springer, 634–647.

[8] Deepanway Ghosal, Navonil Majumder, Ambuj Mehrish, and Soujanya Poria. 2023. Text-to-Audio Generation using Instruction-Tuned LLM and Latent Diffusion Model. *arXiv preprint arXiv:2304.13731* (2023).

[9] Shane Griffith, Kaushik Subramanian, Jonathan Scholz, Charles L Isbell, and Andrea L Thomaz. 2013. Policy Shaping: Integrating Human Feedback with Reinforcement Learning. In *Advances in Neural Information Processing Systems*, C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger (Eds.), Vol. 26. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2013/file/e034fb6b66aacc1d48f445ddfb08da98-Paper.pdf

[10] Hossein Hassani and Emmanuel Sirmal Silva. 2023. The role of ChatGPT in data science: how ai-assisted conversational interfaces are revolutionizing the field. *Big data and cognitive computing* 7, 2 (2023), 62.

[11] 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. *NeurIPS* (2021).

[12] Julia Hirschberg and Christopher D Manning. 2015. Advances in natural language processing. *Science* 349, 6245 (2015), 261–266.

[13] Andreas Jungherr. 2023. Using ChatGPT and Other Large Language Model (LLM) Applications for Academic Paper Assignments. (2023).

[14] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. arXiv:1910.13461 [cs.CL]

[15] Yang Li, Quan Pan, Suhang Wang, Tao Yang, and Erik Cambria. 2018. A generative model for category text generation. *Information Sciences* 450 (2018), 301–315.

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

[17] Xiangyang Liu, Tianqi Pang, and Chenyou Fan. 2023. Federated Prompting and Chain-of-Thought Reasoning for Improving LLMs Answering. *arXiv preprint arXiv:2304.13911* (2023).

[18] Roberto E Lopez-Martinez and Gerardo Sierra. 2020. Natural language processing, 2000-2019—a bibliometric study. *Journal of Scientometric Research* 9, 3 (2020), 310–318.

[19] Preeti Malik, Varsha Mittal, Lata Nautiyal, and Mangey Ram. 2022. NLP techniques, tools, and algorithms for data science. *Artificial Intelligence for Signal Processing and Wireless Communication* 11 (2022), 123.

[20] Hunter McNichols, Mengxue Zhang, and Andrew Lan. 2023. Algebra Error Classification with Large Language Models. *arXiv preprint arXiv:2305.06163* (2023).

[21] 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 (2022), 27730–27744.

[22] 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*. 311–318.

[23] 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 (2019), 9.

[24] Malik Sallam. 2023. ChatGPT utility in healthcare education, research, and practice: systematic review on the promising perspectives and valid concerns. In *Healthcare*, Vol. 11. MDPI, 887.

[25] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. arXiv:1707.06347 [cs.LG]

[26] Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. 2020. Learning to summarize with human feedback. *Advances in Neural Information Processing Systems* 33 (2020), 3008–3021.

[27] Karthik Valmeekam, Alberto Olmo, Sarath Sreedharan, and Subbarao Kambhampati. 2022. Large Language Models Still Can't Plan (A Benchmark for LLMs on Planning and Reasoning about Change). *arXiv preprint arXiv:2206.10498* (2022).

[28] Jing Wang, Huan Deng, Bangtao Liu, Anbin Hu, Jun Liang, Lingye Fan, Xu Zheng, Tong Wang, Jianbo Lei, et al. 2020. Systematic evaluation of research progress on natural language processing in medicine over the past 20 years: bibliometric study on PubMed. *Journal of medical Internet research* 22, 1 (2020), e16816.

[29] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171* (2022).

[30] Yuhui Wang, Hao He, and Xiaoyang Tan. 2020. Truly Proximal Policy Optimization. In *Proceedings of The 35th Uncertainty in Artificial Intelligence Conference (Proceedings of Machine Learning Research, Vol. 115)*, Ryan P. Adams and Vibhav Gogate (Eds.). PMLR, 113–122. https://proceedings.mlr.press/v115/wang20b.html

[31] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. arXiv:2201.11903 [cs.CL]

[32] Imad Zeroual and Abdelhak Lakhouaja. 2018. Data science in light of natural language processing: An overview. *Procedia Computer Science* 127 (2018), 82–91.

[33] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. BERTScore: Evaluating Text Generation with BERT. arXiv:1904.09675 [cs.CL]

[34] Zhilu Zhang and Mert Sabuncu. 2018. Generalized cross entropy loss for training deep neural networks with noisy labels. *Advances in neural information processing systems* 31 (2018).

[35] Ou Zheng, Mohamed Abdel-Aty, Dongdong Wang, Zijin Wang, and Shengxuan Ding. 2023. ChatGPT is on the horizon: Could a large language model be all we need for Intelligent Transportation? *arXiv preprint arXiv:2303.05382* (2023).

[36] Daniel M. Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B. Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. 2020. Fine-Tuning Language Models from Human Preferences. arXiv:1909.08593 [cs.CL]