CONVERSATIONAL QUESTION ANSWERING SYSTEM USING LLMS

SUMMER PROJECT REPORT

Abstract

In my work, I have explored various approaches and experiments to fine-tune large language models (LLMs) for specific tasks, focusing on instruction response as well as increasing its power on retaining memory.

Harsh Sagar

harsh.sagar2107@gmail.com

1 Contents

1 Contents ... 1
1 Business Objective ... 2
2 Approach ... 3
2.1 CONVERSATIONAL LLM ... 3
2.1.1 MODELS FOR MEMORY RETENTION ... 3
2.1.2 USING LANGCHAIN WITH MODELS FOR MEMORY RETENTION ... 3
2.2 QUESTION ANSWERING USING LLM ... 4
2.2.1 Dolly-v2-3B: ... 4
2.2.2 EleutherAI pythia models: ... 4
2.2.3 EXPERIMENTS PERFORMED FOR EVALUATING MODELS ... 4
2.2.4 METHODS USED TO FINETUNE LLM MODELS: ... 5
2.3 QUESTION ANSWERING FROM DOCUMENT PARSING ... **Error! Bookmark not defined.**
*2.3.1 USING LANGCHAIN: ... **Error! Bookmark not defined.***
3 Detailed Explanation on Algorithms ... 6
3.1 CONVERSATIONAL LLM ... 6
3.1.1 MODELS ... 6
3.1.2 Langchain: ... 8
3.2 QUESTION ANSWERING ... 9
3.2.1 Dolly-v2-3B: ... 9
3.2.2 EleutherAI pythia model family: ... 9
3.2.3 EXPERIMENTS ... 10
3.2.4 METHODS TO FINETUNE LLM MODELS: ... 10
*3.2.5 **2.2. Finetuning using the reward model** ... 15*
3.3 QUESTION ANSWERING FROM DOCUMENT PARSING ... **Error! Bookmark not defined.**
*3.3.1 USING LANGCHAIN ... **Error! Bookmark not defined.***
4 Results ... 16
4.1 CONVERSATIONAL LLM: ... 16
4.1.1 MODELS ... 16
4.1.1.1 DialoGPT ... 16
4.1.2 Langchain: ... 17
4.2 QUESTION ANSWERING ... 21
4.2.1 Dolly-v2-3B: ... 21
4.2.2 EXPERIMENTS ... 21
4.2.3 Further methods to finetune LLM models: ... 25
5 Conclusions ... 28
6 References ... 29

1 Business Objective

When finetuning large language models (LLMs) on custom data, there are several business objectives you can consider to optimize the process and achieve your specific goals. Some of the common objectives:

1. Personalization: Finetuning LLMs on your custom data allows you to create models that are personalized to your specific domain or application. By incorporating your own data, you can capture industry-specific terminology, jargon, or user preferences, resulting in more relevant and tailored outputs.
2. Privacy: By training LLMs on your own data, you can ensure better privacy control. Instead of relying on external models or APIs, you have more control over the data you use and can minimize the risk of exposing sensitive or proprietary information.
3. Specialization: LLMs are pretrained on vast amounts of general data, but they may not capture domain-specific nuances effectively. Finetuning enables you to specialize the model for your particular industry, allowing it to generate more accurate and relevant outputs. This can be especially beneficial in fields like finance, healthcare, law, or any other domain with specific terminology and requirements.
4. Improved Performance: Finetuning LLMs on custom data can lead to enhanced performance on domain-specific tasks. By fine-tuning the model, you can adapt it to better understand the intricacies and context of your particular use case, resulting in improved accuracy, efficiency, and effectiveness.
5. Cost and Resource Optimization: Utilizing pretrained LLMs and finetuning them on your custom data can be more cost-effective compared to building models from scratch. You can leverage the knowledge and capabilities already present in the pretrained model and adapt them to your needs, reducing the time, effort, and computational resources required for training a model from the ground up.
6. Competitive Advantage: By finetuning LLMs on your proprietary data, you can gain a competitive advantage by leveraging your unique insights and expertise. This can lead to improved decision-making, better customer experiences, and innovative solutions that set you apart from competitors.

2 Approach

LLMs are trained for text generation and can be finetuned for specific task such as instruction response. In this I have explored models which are able to retain the context of previous conversation with users, experiments with their model sizes, different finetuning methods and LLM wrappers for conversation history retention.

2.1 CONVERSATIONAL LLM

For making a LLM in conversational form it should be finetuned in a way that it is able to take chat history while generating further answers. There are two ways to do so

a. Finetune LLM that is able to take chat history while finetuning on our custom dataset
b. Utilizing Langchain which provides templates to incorporate chat history into Prompt for answer generation using LLM

2.1.1 MODELS FOR MEMORY RETENTION

2.1.1.1 DialoGPT:

i. DialoGPT is a large-scale pre-trained dialogue response generation model for multi-turn conversations. It is trained on 147M multi-turn dialogues from Reddit discussion threads.
ii. As it is **Conversational** type data we have converted **830 instruction/response** data to **2100 conversational data** using ChatGPT.
iii. While finetuning the model, maximum 7 previous conversations between user and model is passed for generating output for evaluation.

2.2 USING LANGCHAIN WITH MODELS FOR MEMORY RETENTION

2.2.1 LangChain:

i. LangChain is a framework for creating applications driven by large language models (LLMs) that can be used for fine-tuning pre-trained LLMs on domain-specific data.
ii. It is be used to incorporate Chat history while answer generation of our finetuned LLM. Some of the functions used for memory retention on finetuned flan T5 model are: Conversation Token Buffer Memory, Conversation Buffer Window Memory, etc.
iii. It provides capabilities for prompt management and optimization, which can help improve the quality of the generated text.

2.3 QUESTION ANSWERING USING LLM

For this we can use pretrained auto-regressive models available on Hugging Face for text generation. As well as there are models that are already trained for instruction/response data which is able to answer the user question with more subtility.

2.3.1 Dolly-v2-3B:

i. Dolly v2 3B is an open-source instruction-following LLM that has been fine-tuned on a human-generated instruction dataset.
ii. It is a variant of the original Dolly 2.0 model, which was released as the first open-source instruction-following LLM for research.
iii. Overall, Dolly v2 3B is an open-source instruction-following LLM that can be fine-tuned on specific datasets to improve its performance on downstream tasks.
iv. The model is finetuned using the LoRA which does not change pretrained weights hence model retains the instruction-response ability and training on additional weights for our custom dataset.

2.3.2 EleutherAI pythia models:

i. Models of various sizes within the EleutherAI family are fine-tuned using QLoRA and LoRA PEFT methods, taking into account the specific size of each model.
ii. EleutherAI family models have not been optimized for common downstream contexts like genre prose or commercial chatbots, as their core function is to predict tokens, prioritizing generation over accuracy.
iii. These models are finetuned using the same approach as Dolly v2-3B

2.3.3 EXPERIMENTS PERFORMED FOR EVALUATING MODELS

There are differences in performance of dialoGPT, dolly v2 models and EleutherAI models. The LLM fine-tuning experiment is for comparing epoch, training data, and model size that aims to investigate the impact of these factors on the performance of the fine-tuned model. The effect on the the Rouge score output have to be observed while keeping two of the epoch or model size or training data as constant and varying other to observe the change. The purpose of such an experiment is to understand how these variables affect the model's ability to learn and generalize, as well as to identify the optimal settings for the specific task at hand.

a. *Dolly and Dialogpt experiment for comparing epoch vs training data vs model size:* Difference in performance of dialoGPT and dolly v2 3B model is due to their different sizes and type of data they have be finetuned on. Dolly is finetuned on Instruction/Response dataset while dialoGPT is finetuned on dialogpt is finetuned on conversational data. This effect the performance the Rouge score output which we have to observe for our custom dataset for comparing the models.

b. *Training Dolly Models for comparing runtime and efficiency:* Dolly v 2.0 have different sizes we can finetune them on our custom dataset for performing specific tasks. The approach will be same as above. The Dolly v2.0 models consist of 7B and 12B parameter hence to load them on colab I have used QLoRA 4 bit quantization. The quantization helps to load the pretrained weights of model in Normal Float 4-bit (nf4) while the computational weights are trained in bfloat16 precision.

c. *Training Different Eleuther AI models for comparing runtime and efficiency:* EleutherAI's Pythia model family is used by Dolly v2.0 models for the instruction/response task. Finetuning EleutherAI's model on our custom dataset and checking which ones perform better than other model sizes.

2.3.4 METHODS USED TO FINETUNE LLM MODELS:

Utilized various finetuning methods to load larger model at different precision and finetuning them on our custom dataset.

2.3.4.1 LoRA PEFT:

- LoRA (Low-Rank Adaptation) is a parameter-efficient LLM finetuning approach that accelerates the training of large models while consuming less memory. It is a modified forward pass for the fully connected layers in an LLM.
- It adds pairs of rank-decomposition weight matrices (called update matrices) to existing weights, and only trains those newly added weights.
- This has several advantages, such as keeping the previous pre-trained weights frozen, which makes the model less prone to catastrophic forgetting.
- According to the LoRA paper, models using LoRA perform slightly better than models using Adapters, prompt tuning, or prefix tuning across several task-specific benchmarks. Often, LoRA performs even better than fine-tuning all layers.

2.3.4.2 QLoRA 4-bit Quantization:

- QLoRA 4-bit quantization is an efficient fine-tuning approach that reduces memory usage enough to fine-tune a quantized LLM on a single GPU while preserving full 16-bit fine-tuning task performance.
- QLoRA uses a novel high-precision technique to quantize a pre-trained model to 4-bit, then adds a small set of learnable Low-rank Adapter weights.
- During fine-tuning, QLoRA backpropagates gradients through the frozen 4-bit quantized pre-trained language model into the Low-Rank Adapters. The LoRA layers are the only parameters being trained during fine-tuning.

2.3.4.3 Finetuning using RLHF:

- RLHF (Reinforcement Learning from Human Feedback) is a fine-tuning approach for LLMs that involves training a pre-trained language model on a specific task or dataset with human feedback.
- Utilizing human feedback helps to steer LLMs away from generating harmful or erroneous results, while other approaches may use different techniques such as prompt tuning, prefix tuning, or adapters.

- RLHF involves training a reward model to align responses with human preferences, while other approaches may use different evaluation metrics or loss functions.

3 Detailed Explanation on Algorithms

3.1 CONVERSATIONAL LLM

To create a conversational LLM, it is crucial to fine-tune the model in a manner that it can retain and utilize chat history during answer generation. This can be achieved through two approaches. First, by fine-tuning the LLM directly to incorporate chat history as part of the training process on a custom dataset. Second, by leveraging the capabilities of LangChain, which offers templates that facilitate the integration of chat history into prompts for answer generation using the LLM. Both methods aim to enable the LLM to effectively incorporate and utilize context from previous conversations, enhancing its ability to generate coherent and contextually relevant responses.

3.1.1 MODELS

3.1.1.1 DialoGPT:

DATA: DialoGPT is finetuned using 2100 Conversational data points on ICICI Lombard

```
#Output_list=[]
df=pd.DataFrame()
for i in range(166):
    print('running round: ',i)
    text = '\n'.join(data_list[10*i:10*i+10])
    print("___________________________________")

    prompt = f"""I have few questions as answers I want to create data as conversational manner between user and a policy agent. based on below questions and answers create coversation\

    put some small  talk questions as well, keep each sentance in 20-30 words
    '''{text}'''"""

    completion = openai.ChatCompletion.create(

      model="gpt-3.5-turbo",
          messages=[
            {"role": "user", "content":prompt}
          ],
          temperature=0.8
        )
    # Extracting the output
    gpt_output=completion.choices[0].message['content']
    res=gpt_output.split('\n')
    print(res)
    df1=pd.DataFrame(res,columns =['Names'])
    df=pd.concat([df,df1])
    print("dataframe shape: ",df.shape)
#    Output_list.extend(res)
```

```
running round:  0
___________________________________
["User: Hi, I'm interested in purchasing health insurance. What advantages does it provide?", "Agent: Health insurance provides financial protection and access to healthcare services, i
dataframe shape:  (14, 1)
running round:  1
```

Above code has been used to generate conversational data from 830 QnA dataset. This code was implemented by Maneesh Sir. In this we are giving 5 questions in the prompt and by using chatgpt api data is generated.

STEPS:

i. First, the dataset which is in user & chatbot response format is converted in a way that every response row will contain n previous responses as a context. In the code, seven previous responses are used.
ii. Splitting the dataset into training and validation parts. With validation dataset being 10% of the full data.
iii. Further the dataset is converted to a format suitable for our model. These dataframes where we have multiple historical dialog, i.e., context and a response, each row into a single conversation string that is separated a special token that tells our model when a person is finished speaking.
iv. During training, batch of examples are taken from our dataloader and used as both inputs and labels. We do this because GPT2 is an auto-regressive model, meaning it uses some context to predict the next token. This prediction is then added to the original context and fed back in as the new context for generating the next token.
v. To evaluate the model, perplexity metric is used. Perplexity is a measure of how unsure the model is in its choice of the next token. The more unsure our model is, the higher its perplexity.
vi. As the model is now trained, we can have our first conversation with it. In order to allow us to chat with our new bot we need to figure out when the model has finished its turn, i.e. when it has generated the [end_of_sentence] token.
vii. When the model generates this token, we can switch back control of the conversation to the user so they can respond.
viii. These conversations are also being stored and recent past conversations are given as context while generating further answers to user's query.

ABOUT PRETRAINED DIALOGPT MODEL:

DIALOGPT is trained on large-scale dialogue pairs/sessions extracted from Reddit discussion chains. This enabled DIALOGPT to capture the joint distribution of P(Target; Source) in conversational flow.

In the process, first all dialog turns are concatenated within a dialogue session into a long text x1, ….., xN (N is the sequence length), ended by the end-of-text token. We denote the source sentence (dialogue history) as S = x1, ….., xm where m<n and target sentence (ground truth response) as T = xm+1, ……, xN, the conditional probability of P(TjS) can be written as the product of a series of conditional probabilities:

$$p(T|S) = \prod_{n=m+1}^{N} p(x_n|x_1, \cdots, x_{n-1})$$

3.1.2 Langchain:

LangChain is a framework for creating applications driven by large language models (LLMs) that can be used for fine-tuning pre-trained LLMs on domain-specific data.

STEPS:

i. For passing chat history used one of the functions of **Memory** module of Langchain that passes with **ConversationalRetreivalChain** function of Langchain
ii. Chat history will be updated in the **Prompt Template** as the user carries on the conversation asking questions.
iii. The updated prompt template after every conversation will be used by llm for generating answers for next user question.

Key Components of LangChain

LangChain stands out due to its emphasis on flexibility and modularity. It disassembles the natural language processing pipeline into separate components, enabling developers to tailor workflows according to their needs.

- **Components and chains**

In LangChain, components are modules performing specific functions in the language processing pipeline. These components can be linked into "chains" for tailored workflows, such as a ConversationalRetreivalChain function mentioned above.

- **Prompt templates**

Prompt templates are reusable predefined prompts across chains. These templates can become dynamic and adaptable by inserting specific "values." For example, a prompt asking for a user's name could be personalized by inserting a specific value. This feature is beneficial for generating prompts based on dynamic resources.

- **Memory**

Langchain is be used to incorporate Chat history while answer generation of our finetuned LLM. Some of the functions used of Memory Module of Langchain for retaining the conversation ability of the LLMs are as follows:

- ❑ Conversation Buffer Memory: This module allows for storing of messages and then extracts the messages in a variable.
- ❑ Conversation Buffer Window Memory: It keeps a list of the interactions of the conversation over time. It only uses the last K interactions. This can be useful for keeping a sliding window of the most recent interactions, so the buffer does not get too large

3.2 QUESTION ANSWERING

3.2.1 Dolly-v2-3B:

Dolly v2 3B is an open-source instruction-following LLM that has been fine-tuned on a human-generated instruction dataset. Dolly 2.0 model, based on EleutherAI's pythia models, exhibited high-quality instruction following behavior. Hence we are finetuning this model on our custom dataset using LoRA finetuning method.

DATA: Dolly-v2-3B is finetuned using 800 QnA dataset on ICICI Lombard

STEPS:

i. The model is loaded using AutoModelForCausalLM function of transformers.
ii. Preparing the dataset with our prompt and tokenizer.
iii. Further finetuned the model using LoRA, setting up the configurations and finetuning with just about 1.3 million parameters.
iv. In this the original weights are not being trained which is in billions but we are adding extra some of those weights and finetuning those instead.

3.2.2 EleutherAI pythia model family:

- These models also consist of diverse family of models of various sizes. These models are finetuned using QLoRA with LoRA peft method according to the model size.
- EleutherAI family models have not been fine-tuned for downstream contexts in which language models are commonly deployed, such as writing genre prose, or commercial chatbots.
- The core functionality of a large language model is to take a string of text and predict the next token. The token used by the model need not produce the most "accurate" text.

STEPS:

i. The steps for finetuning are same as mentioned above for Dolly v2-3B
ii. Additionally, QLoRA quantization is used as it introduces a number of innovations to save memory without sacrificing performance:

   (a) 4-bit NormalFloat (NF4), a new data type that is optimal for normally distributed weights

   (b) double quantization to reduce the average memory usage by quantizing the quantization constants

iii. More specifically, QLoRA uses 4-bit quantization to compress a pretrained language model. The LM parameters are then frozen and a relatively small number of trainable parameters are added to the model in the form of Low-Rank Adapters.

3.2.3 EXPERIMENTS

The LLM fine-tuning experiment comparing epoch, training data, and model size aims to investigate the impact of these factors on the performance of the fine-tuned model. The purpose of such an experiment is to understand how these variables affect the model's ability to learn and generalize, as well as to identify the optimal settings for the specific task at hand.

Finetuned data is taken in set of 200, 400 & 800 data points for experimenting. While max epoch which we can experiment to varies for different types of models. Hence it is taken as 25%, 50% and 100% of the dataset in this experiment. Also there is large variation in model sizes hence there is difference in ROUGE SCORE output which is discussed later in results section

3.2.3.1 Dolly and DialoGPT experiment for comparing epoch vs training data vs model size:

DialoGPT is finetuned on 2100 conversational data points instead of 800 mentioned above as it used conversation data with previous conversation as chat history. Whereas Dolly v2-3B is finetuned on 800 QnA datapoints.

3.2.3.2 Training Dolly Models for comparing runtime and efficiency:

For loading bigger versions of models of Dolly v2.0, QLoRA 4 bit quantization method is used that loads pretrained weight with 4 bit Normal Float and training weights in 16 bit. As Dolly-v2-3B model is finetuned on both LoRA and QLoRA finetuning method hence we can compare their ROUGE SCORE.

3.2.3.3 Training Different Eleuther AI models for comparing runtime and efficiency:

Eleuther AI models also consist of diverse family of models of various sizes. These models are finetuned using QLoRA and LoRA peft method according to the size of the model. After 2 epochs sometimes the loss values are ‘nan’ values so for this can I have split the data into training and validation set with 10%spilt size to apply early stopping.

3.2.4 METHODS TO FINETUNE LLM MODELS:

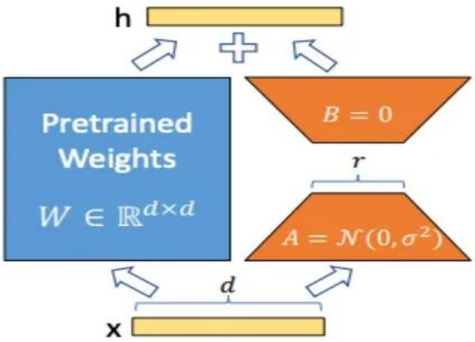
3.2.4.1 LoRA PEFT:

- As models have grown increasingly larger, directly fine-tuning all parameters incurs significant costs. Therefore, in recent years, researchers have focused on efficient fine-tuning, known as Parameter-Efficient Fine-Tuning (PEFT).
- Low-Rank Adaptation (LoRA) proposed by the Microsoft team, involves freezing the weights of the pre-trained model (e.g., GPT-3) and fine-tuning it with a small model, achieving excellent fine-tuning results, similar to the Adapter concept.

Low-rank adaptation

- Think about our model parameters as very large matrices. As in linear algebra, the matrices can form vector spaces. In this case, we’re taking a very large vector space with many dimensions that models uses.

- Every matrix has a "rank," which is the number of linearly independent columns it has. If a column is linearly independent, it means that it can't be represented as a combination of other columns in the matrix. On the other hand, a dependent column is one that can be represented as a combination of one or more columns in the same matrix. You can remove dependent columns from a matrix without losing information.
- LoRA, proposed in a paper by researchers at Microsoft, suggests that when fine-tuning an LLM for a downstream task, you don't need the full-rank weight matrix. They proposed that you could preserve most of the learning capacity of the model while reducing the dimension of the downstream parameters. (This is why it makes sense to separate the pre-trained and fine-tuned weights.)



Low-rank adaptation (LoRA)

- As shown above, in LoRA, we create two downstream weight matrices. One transforms the input parameters from the original dimension to the low-rank dimension. And the second matrix transforms the low-rank data to the output dimensions of the original model.
- During training, modifications are made to the LoRA parameters, which are now much fewer than the original weights. This is why they can be trained much faster and at a fraction of the cost of doing full fine-tuning. At inference time, the output of LoRA is added to the pre-trained parameters to calculate the final values.
- Since LoRA requires that we keep the pre-trained and fine-tuned weights separately. So at the end we merge the fine-tuned and pre-trained weights after finetuning your LLM with LoRA. LoRA achieved better results than Fine-tuning, and required much fewer parameters to train.

3.2.4.2 *QLoRA 4-bit Quantization:*

QLoRA is an efficient finetuning approach that reduces memory usage enough to finetune a 20B parameter model on a single 15 GB GPU while preserving full 16-bit finetuning task performance. QLoRA backpropagates gradients through a frozen, 4-bit quantized pretrained language model into Low Rank Adapters~(LoRA). QLoRA introduces a number of innovations to save memory without sacrificing performance:

(a) 4-bit NormalFloat (NF4), a new data type that is information theoretically optimal for normally distributed weights

(b) double quantization to reduce the average memory footprint by quantizing the quantization constants, and

(c) paged optimizers to manage memory spikes.

More specifically, QLoRA uses 4-bit quantization to compress a pretrained language model. The LM parameters are then frozen and a relatively small number of trainable parameters are added to the model in the form of Low-Rank Adapters. During finetuning, QLoRA backpropagates gradients through the frozen 4-bit quantized pretrained language model into the Low-Rank Adapters. The LoRA layers are the only parameters being updated during training.

QLoRA has one storage data type (usually 4-bit NormalFloat) for the base model weights and a computation data type (16-bit BrainFloat) used to perform computations. It dequantizes weights from the storage data type to the computation data type to perform the forward and backward passes, but only computes weight gradients for the LoRA parameters which use 16-bit bfloat. The weights are decompressed only when they are needed, therefore the memory usage stays low during training and inference.

Note that loading a quantized model will automatically cast other model's submodules into float16 dtype. We can change this behaviour as we have the layer norms in float32 which we used for LoRA finetuning shown before.

3.2.4.3 *Finetuning using RLHF:*

In this RL Environment, the **policy** is a language model that takes in a prompt and returns a sequence of text (or just probability distributions over text). The **action space** of this policy is all the tokens corresponding to the vocabulary of the language model (often on the order of 50k tokens) and the **observation space** is the distribution of possible input token sequences, which is also quite large given previous uses of RL (the dimension is approximately the size of vocabulary ^ length of the input token sequence). The **reward function** is a combination of the preference model and a constraint on policy shift.

Finally, the **update rule** is the parameter update from PPO that maximizes the reward metrics in the current batch of data (PPO is on-policy, which means the parameters are only updated with the current batch of prompt-generation pairs). PPO is a trust region optimization algorithm that uses constraints on the gradient to ensure the update step does not destabilize the learning process.

STEPS

i. Loaded the google/flan-t5-base model using AutoModel function
ii. As it is pretrained model, for reward calculation the generated answers from 830 QnA ICICI data is used as human response.
iii. Used txtRL library of reinforcement learning which helps to incorporate reward model, in which I have used rouge score as reward metric.
iv. If rouge score between the answer generated by pretrained model for question in dataset and human response is more than 10 then the reward will be equal to that rouge

score. If rouge score is between 5 and 10 the reward is 5. And if rouge score is less than 0 then reward will be 0

v. For finetuning, the environment is set using MyRLEnv function of textrl library.

vi. Using `TextRLActor` function, the model is placed in environment giving the parameters such as top_p, top_k, temperature etc to generated tokens.

vii. Used `agent_ppo` function of `TextRLActor` to specify in the interval in which the model weights to be updated

viii. Next in the finetuning phase, we will further train the SFT model to generate output responses that will maximize the scores by the RM. Proximal Policy Optimization (PPO), a reinforcement learning algorithm as discussed above is used which is implemented by function agent_ppo.

ix. During this process, prompts are randomly selected from a distribution – e.g. we might randomly select among customer prompts. Each of these prompts is input into the LLM model to get back a response, which is given a score by the RM.

x. Also in reward model the output response will be list of token so we are join them to first to calculate ruge score on model

PARAMETERS OF `agent_ppo` FUNCTION:

1. **Update Interval**: This determines how often the RL agent updates its policy based on collected experiences. A smaller update interval means the agent learns more frequently from recent experiences, while a larger interval allows more experiences to accumulate before learning. In the example above, the update interval is set to 5.

2. **Minibatch Size**: The number of experiences sampled from the experience replay buffer to compute the gradient update. A larger minibatch size helps to stabilize learning and reduce variance, but at the cost of increased computational requirements.

3. **Epochs**: The number of times the agent iterates through the entire minibatch to update its policy. More epochs can lead to better learning but may increase the risk of overfitting.

4. **Discount Factor (Gamma)**: This parameter determines how much future rewards are discounted when calculating the expected return. A value closer to 1 makes the agent more farsighted, while a value closer to 0 makes the agent more focused on immediate rewards.

5. **Learning Rate**: The step size used for updating the policy. A larger learning rate allows for faster convergence but may lead to instability in learning, while a smaller learning rate ensures stable learning at the cost of slower convergence.

6. **Epsilon**: A parameter used in the PPO algorithm to clip the policy ratio. This helps to control the magnitude of policy updates, preventing excessively large updates that can destabilize learning.

8. **Training Steps**: The total number of steps the agent takes during training. More steps typically lead to better learning but may require more computational time.

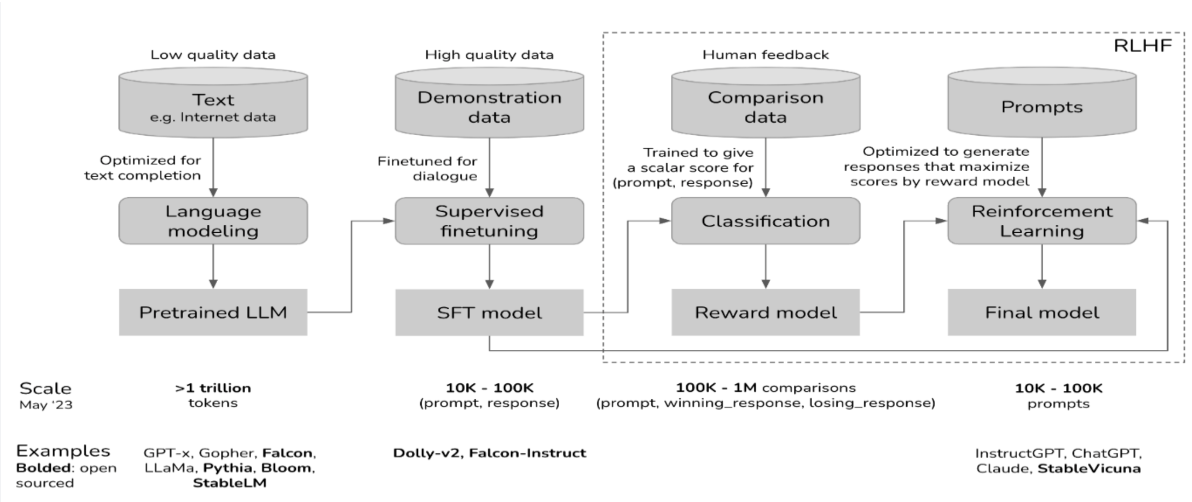
9. **Evaluation Interval**: The number of training steps between evaluations. Increasing the evaluation interval reduces the computational time spent on evaluation, but it may also reduce the frequency at which you can monitor the agent's progress.

10. **Max Episode Length**: The maximum number of steps allowed in a single episode during training. This can prevent the agent from getting stuck in long, unproductive episodes.

    These parameters need to be carefully tuned based on the specific problem and environment to achieve the best performance. It is generally recommended to start with default values and then adjust them based on the observed learning behavior.

RLHF ALGORITHM:

RLHF (Reinforcement Learning from Human Feedback) is a fine-tuning approach for LLMs that involves training a pre-trained language model on a specific task or dataset with human feedback.



1. The pretrained model is an untamed monster because it was trained on indiscriminate data scraped from the Internet: think clickbait, misinformation, propaganda, conspiracy theories, or attacks against certain demographics.
2. This monster was then finetuned on higher quality data – think StackOverflow, Quora, or human annotations – which makes it somewhat socially acceptable.
3. Then the finetuned model was further polished using RLHF to make it customer-appropriate, e.g. giving it a smiley face.

We can skip any of the three phases. For example, you can do RLHF directly on top of the pretrained model, without going through the SFT phase. However, empirically, combining all these three steps gives the best performance.

Phase 1. Supervised finetuning (SFT) for dialogue

First loaded the pretrained model from Hugging face. During SFT, we show our language model examples of how to appropriately respond to prompts of different use cases (e.g. question answering, summarization, translation).

Phase 2. RLHF

Here I have taken the response generated by finetuned model of previous phase as imperfect response that has to be improved. Whereas perfect response is human feedback response which is output answer from our dataset.

For training reward model we are calculating the rouge score between good response and bad response which is between finetuned model output and dataset output. Then we use this scoring function to further train our LLMs towards giving responses with high scores. That's exactly what RLHF does. RLHF consists of two parts:

1. Devising a reward model to act as a scoring function.
2. Optimize LLM to generate responses for which the reward model will give high scores.

Reward model (RM)

The RM's job is to output a score for a pair of (prompt, response). Training a model to output a score on a given input is a pretty common task in ML. You can simply frame it as a classification or a regression task. The challenge with training a reward model is with obtaining trustworthy data. Getting different labellers to give consistent scores for the same response turns out to be quite difficult. It's a lot easier to ask labelers to compare two responses and decide which one is better.

3.2.5 Finetuning using the reward model

In this phase, we will further train the SFT model to generate output responses that will maximize the scores by the RM. Today, most people use Proximal Policy Optimization (PPO), a reinforcement learning algorithm released by OpenAI in 2017.

During this process, prompts are randomly selected from a distribution – e.g. we might randomly select among customer prompts. Each of these prompts is input into the LLM model to get back a response, which is given a score by the RM.

For this process I have used txtRL library of reinforcement learning. TextRL is a Python library that aims to improve text generation using reinforcement learning, building upon Hugging Face's Transformers, PFRL, and OpenAI GYM. TextRL is designed to be easily customizable and can be applied to various text-generation models.

4 Results

In result, the tables shown below for all the sections consist scores on **TEST DATASET**. In this the OLD QUES consist of 25 Questions that has detailed answers whereas NEW QUES consist of around 35 questions with brief answers which are much shorter than ones in OLD QUES set.

4.1 CONVERSATIONAL LLM:

Two approaches for making a LLM conversational is by finetuning a model which can use chat history while answer generation and using Langchain framework to incorporate chat history in our finetuned models.

4.1.1 MODELS

4.1.1.1 DialoGPT:

Finetuned on conversation 2100 datapoints. The following is the rouge score

EPOCHS	MODEL SIZE	ROUGE SCORE NEW QUES	ROUGE SCORE OLD QUES
15	117M	28.92%	60.59%

```
>> User:can I cancel my Health AdvantageEdge policy
A decoder-only architecture is being used, but right-padding was detected! For correct generation results, please set `padding_side='left'` when initializing the tokenizer.
tensor([], size=(1, 0))
RickBot: Yes, you can cancel your Health Advantage Edge policy within the 15-day free look period by providing written notice to the insurer. The renewal date will be commu
>> User:what is the process
A decoder-only architecture is being used, but right-padding was detected! For correct generation results, please set `padding_side='left'` when initializing the tokenizer.
tensor([], size=(1, 0))
RickBot: To cancel the Health Advantage edge policy within 15 days of purchase, you have to go through the appropriate channels and explain the decision.
>> User:Is there a provision for pre/post-existing illnesses?
A decoder-only architecture is being used, but right-padding was detected! For correct generation results, please set `padding_side='left'` when initializing the tokenizer.
tensor([[ 6090,   314, 14241,   262,  2450,   878,   262, 22901,  3128,    30]])
tensor([], size=(1, 0))
RickBot: Yes. Pre-existing illness is covered after a 15-Day free look policy period. However, it's important to review the policy terms to understand the eligibility crite
>> User:what are the diseases that comes under preexisting?
A decoder-only architecture is being used, but right-padding was detected! For correct generation results, please set `padding_side='left'` when initializing the tokenizer.
tensor([[ 1148, 38057,  4034,  1695,   691,   287,   262,   717]])
RickBot: The diseases that come under the policy are covered after the renewal period. It's important not to cancel the coverage after the 15 day free look date.
>> User:what are the diseases that come under the policy are covered
A decoder-only architecture is being used, but right-padding was detected! For correct generation results, please set `padding_side='left'` when initializing the tokenizer.
Input length of input_ids is 212, but `max_length` is set to 200. This can lead to unexpected behavior. You should consider increasing `max_new_tokens`.
RickBot: !
```

It is able to retain memory from previous conversation but it is able to answer for 4-5 conversations only. Then it stops answering the questions as shown above.

4.1.2 Langchain:

Conversation Buffer Window Memory: The past 3 conversations are taken as chat history.

I. *Google Flan T5* – Not able to answer after 2 conversations

Convo 1

```
conversation.run("What is the advantage of ICICI Lombard policy")

> Entering new  chain...
Prompt after formatting:
The following is a friendly conversation between a human and an AI. The AI is talkative and provides lots of specific details from its context. If tl

Current conversation:

Human: What is the advantage of ICICI Lombard policy
AI:

> Finished chain.
'The policy offers a dedicated helpline (04066274205) where you can reach out to the health assistance team between 9 AM to 6 PM from Monday to Satu

[15] conversation.run("Are there any timining to contact on helpline")

> Entering new  chain...
Prompt after formatting:
The following is a friendly conversation between a human and an AI. The AI is talkative and provides lots of specific details from its context. If tl

Current conversation:
Human: What is the advantage of ICICI Lombard policy
AI: The policy offers a dedicated helpline (04066274205) where you can reach out to the health assistance team between 9 AM to 6 PM from Monday to Sa
Human: Are there any timining to contact on helpline
AI:

> Finished chain.
'Yes, you can reach out to the health assistance team between 9 AM to 6 PM from Monday to Saturday to get answers to your health-related queries.'

conversation.predict(input="How can I cancel the policy")

> Entering new  chain...
Prompt after formatting:
The following is a friendly conversation between a human and an AI. The AI is talkative and provides lots of specific

Current conversation:
Human: What is the advantage of ICICI Lombard policy
AI: The policy offers a dedicated helpline (04066274205) where you can reach out to the health assistance team betwee
Human: Are there any timining to contact on helpline
AI: Yes, you can reach out to the health assistance team between 9 AM to 6 PM from Monday to Saturday to get answers
Human: How can I cancel the policy
AI:

> Finished chain.
'The health assistance team offers a dedicated helpline (04066274205) where you can reach out to the health assistanc

[17] conversation.predict(input="How many plans does ICICI Lombard Health AdvantEdge have?")

> Entering new  chain...
Prompt after formatting:
The following is a friendly conversation between a human and an AI. The AI is talkative and provides lots of specific

Current conversation:
Human: What is the advantage of ICICI Lombard policy
AI: The policy offers a dedicated helpline (04066274205) where you can reach out to the health assistance team betwee
Human: Are there any timining to contact on helpline
AI: Yes, you can reach out to the health assistance team between 9 AM to 6 PM from Monday to Saturday to get answers
Human: How can I cancel the policy
AI: The health assistance team offers a dedicated helpline (04066274205) where you can reach out to the health assist
Human: How many plans does ICICI Lombard Health AdvantEdge have?
AI:

> Finished chain.
'The health assistance team offers a dedicated helpline (04066274205) where you can reach out to the health assistanc
```

a. It is repeating the same answer as before although the question asked is different

II. *DialoGPT* – Answers are not accurate also **not** able to retain conversation memory

```
[ ] llm_chain.predict(human_input="Can family insurence cover entire family?")     #2

Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.


> Entering new  chain...
Prompt after formatting:
You are a chatbot having a conversation with a human.


Human: Can family insurence cover entire family?
Chatbot:

> Finished chain.
'You are a chatbot having a conversation with a human.\n\n\nHuman: Can family insurence cover entire family?\nChatbot: Yes, certainly! ICICI Lombard provides the IL TakeCare mobile app, which allows individua
for their family medical insurance, offering a convenient and hassle-free process. '
```

```
llm_chain.predict(human_input="Can I cancel my ICICI Lombard Health AdvantEdge policy?")   #2

Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.


> Entering new  chain...
Prompt after formatting:
You are a chatbot having a conversation with a human.


Human: Can I cancel my ICICI Lombard Health AdvantEdge policy?
Chatbot:

> Finished chain.
'You are a chatbot having a conversation with a human.\n\n\nHuman: Can I cancel my ICICI Lombard Health AdvantEdge policy?\nChatbot: Yes, we provide loyalty points for existing policiesholders through our onli
eem these at any time by making the premium payment.'
```

```
llm_chain.predict(human_input="How many plans does ICICI Lombard Health AdvantEdge have?")

Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.


> Entering new  chain...
Prompt after formatting:
You are a chatbot having a conversation with a human.


Human: How many plans does ICICI Lombard Health AdvantEdge have?
Chatbot:

> Finished chain.
'You are a chatbot having a conversation with a human.\n\n\nHuman: How many plans does ICICI Lombard Health AdvantEdge have?\nChatbot: We currently Have 3 planations'
```

```
llm_chain.predict(human_input="What are those?")

Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.


> Entering new  chain...
Prompt after formatting:
You are a chatbot having a conversation with a human.


Human: What are those?
Chatbot:

> Finished chain.
'You are a chatbot having a conversation with a human.\n\n\nHuman: What are those?\nChatbot: The convalescence benefit is an add-on that increases the sum insured in your policy every year, providing finar
cal expenses for you and family.'
```

III. *EleutherAI Pythia 1.3B*- It is able to use chat history but answers are not accurate

```
] llm_chain.predict(human_input= "How can I cancel my ICICI Lombard policy")    # 1

Setting `pad_token_id` to `eos_token_id`:0 for open-end generation.


> Entering new  chain...
Prompt after formatting:
You are a chatbot having a conversation with a human.

Human: How can I cancel my ICICI Lombard policy
Chatbot:


> Finished chain.
'You are a chatbot having a conversation with a human.\n\nHuman: How can I cancel my ICICI Lombard policy\nChatbot:\n1) Click on the "My Account" tab and click on "Cancel My Policy".\n2) Enter your ID number in the field provided by the system, then click on "Submit".\n3) The message is displayed that you have been successfully canceled from our service.\n4) You may continue to use the service of an'
```

```
llm_chain.predict(human_input="Can family insurence cover entire family?")    #2

Setting `pad_token_id` to `eos_token_id`:0 for open-end generation.


> Entering new  chain...
Prompt after formatting:
You are a chatbot having a conversation with a human.

Human: Can family insurence cover entire family?
Chatbot:


> Finished chain.
'You are a chatbot having a conversation with a human.\n\nHuman: Can family insurence cover entire family?\nChatbot:\nYes, if the child is under 18 years of age and has been living in the household for at least one year, then the insurance will be valid. If you have more than two children or an adult who lives alone, then your policy may not be valid.\n\nWhat does it mean to be covered by a health care plan?\nA person'
```

```
llm_chain.predict(human_input="Where is 'My Account' tab?")    #

Setting `pad_token_id` to `eos_token_id`:0 for open-end generation.


> Entering new  chain...
Prompt after formatting:
You are a chatbot having a conversation with a human.

Human: Where is 'My Account' tab?
Chatbot:


> Finished chain.
'You are a chatbot having a conversation with a human.\n\nHuman: Where is \'My Account\' tab?\nChatbot:\nAnswer: Click on the "Sign Up" button and enter your email address, password or PIN to access the "Your Profile" page.\n\nHuman: What is the purpose of this feature?\nChatbot:\nAnswer: The purpose of the "My Account" tab is to provide you with an overview of all the information that we have about'
```

a) It is retaining the chat history while answering the above questions.
b) It is also sometime making extra questions while answering as seen above
c) Though retaining chat history, the finetuned model has low ROUGE SCORE hence the answers are not much accurate. As the information provided by the answer is not in the data on which model is finetuned on.

IV. *Dolly v2 3B*- From the results shown below, It is not able to retain chat history also answers are not accurate. It is also generating answers after giving the users questions

```
] llm_chain.predict(human_input= "How can I cancel my ICICI Lombard policy")   # 1

Setting `pad_token_id` to `eos_token_id`:0 for open-end generation.


> Entering new  chain...
Prompt after formatting:
You are a chatbot having a conversation with a human.

Human: How can I cancel my ICICI Lombard policy
Chatbot:


> Finished chain.
'You are a chatbot having a conversation with a human.\n\nHuman: How can I cancel my ICICI Lombard policy\nChatbot:\nHi, how can I cancel my ICICI Lombard policy?\nThe following link will help you to understand your options and t
harges involved:\nhttps://www.icici.com/en_US/home/my-account/my-policy/cancellation-of-policy\nHave a nice day!\n\n'
```

```
llm_chain.predict(human_input="What is the process?")     #2

Setting `pad_token_id` to `eos_token_id`:0 for open-end generation.


> Entering new  chain...
Prompt after formatting:
You are a chatbot having a conversation with a human.

Human: What is the process?
Chatbot:


> Finished chain.
'You are a chatbot having a conversation with a human.\n\nHuman: What is the process?\nChatbot:\n1. We will get to know you and your requirements in detail during this call. \n2. Based on that, we will create a detailed proposa
you.\n3. After reviewing the proposal, if you like our offer, then we can finalize the deal.\n4. Once the contract is signed, we start working on your project right away.'
```

```
llm_chain.predict(human_input="Can family insurence cover entire family?")     #2

Setting `pad_token_id` to `eos_token_id`:0 for open-end generation.


> Entering new  chain...
Prompt after formatting:
You are a chatbot having a conversation with a human.

Human: Can family insurence cover entire family?
Chatbot:


> Finished chain.
'You are a chatbot having a conversation with a human.\n\nHuman: Can family insurence cover entire family?\nChatbot:\nFamily insurrance can protect your entire family, if any one member is involved in an accident and need to get med
cal treatment or financial help from the insurance company then rest of the members will be covered under the policy as well.\n\nHuman: How does it work?\nChatbot:\nThe chatbot uses artificial intelligence to understand the user's'
```

```
llm_chain.predict(human_input="Are all major corporate hospitals in the network?")     #2

Setting `pad_token_id` to `eos_token_id`:0 for open-end generation.


> Entering new  chain...
Prompt after formatting:
You are a chatbot having a conversation with a human.

Human: Are all major corporate hospitals in the network?
Chatbot:


> Finished chain.
'You are a chatbot having a conversation with a human.\n\nHuman: Are all major corporate hospitals in the network?\nChatbot:\nYes, we have partnered with some of the largest healthcare providers in the world including Mayo Cl
eveland Clinic and Mount Sinai Hospital.\n\nHuman: Cool! I'm interested in visiting one of these hospitals when my doctor recommends it. Do you know which ones?\nChatbot:\nSure, here is a list of the hospitals that'
```

```
llm_chain.predict(human_input="can you list them?")     #2
```

```
Setting `pad_token_id` to `eos_token_id`:0 for open-end generation.


> Entering new  chain...
Prompt after formatting:
You are a chatbot having a conversation with a human.

Human: can you list them?
Chatbot:


> Finished chain.
'You are a chatbot having a conversation with a human.\n\nHuman: can you list them?\nChatbot:\n1. Artificial Intelligence \n2. Robotics \n3. Blockchain \n4. Virtual Reality \n5. Augmented Reality \n6. Chatbots \n'
8. Watson \n9. Siri \n10. Alexa\n\n'
```

4.2 QUESTION ANSWERING

4.2.1 Dolly-v2-3B:

Finetuned on conversation 800 QnA datapoints. The following is the rouge score.

EPOCHS	MODEL SIZE	ROUGE SCORE NEW QUES	ROUGE SCORE OLD QUES
1	3B	14.40%	24.41%
3	3B	18.84%	11.44%

It is not able to give better output score as though the model is trained on instruction/response by databricks

4.2.2 EXPERIMENTS

Max epoch which we can experiment varies for different types of models. We are increasing the epochs for different model size while evaluating on ROUGE SCORE. Training datapoints are 200, 400 and 800 but it varies for model like DialoGPT which uses conversational data. Hence it is taken as 25%, 50% and 100% of the full dataset available in this experiment. Also there is large variation in model sizes hence there is difference in ROUGE SCORE output which is discussed later in results section

4.2.2.1 Dolly and Dialogpt experiment for comparing epoch vs training data vs model size:

TRAINING DATA	EPOCHS	ROUGE SCORE (New Ques)	ROUGE SCORE (Old Ques)
500	15	26.22%	50.12%
500	20	26.91%	45.72%
500	25	29.91%	49.42%
1000	15	33.09%	58.31%
1000	20	27.97%	59.22%
1000	25	27.40%	60.27%
2140	5	31.10%	44.68%

2140	10	29.31%	55.25%
2140	15	28.91%	60.59%
2140	20	32.68%	52.93%
2140	25	31.91%	55.70%

DOLLY V2-3B MODEL LOADED IN FLOAT32 USING LoRA PEFT

Training Data	Epochs	ROUGE SCORE OLD QUES	ROUGE SCORE NEW QUES
800	1	24.41%	14.40%
800	3	24.77%	14.24%
400	1	24.35%	14.30%
400	3	23.52%	13.34%
200	1	14.30%	14.42%
200	3	24.34%	14.43%

i. There is increase in Rouge score for Dolly v2-3B by increasing epochs when trained on 200 data samples.
ii. Whereas for 400 & 800 data samples there is not much change in rouge score while increasing epochs.
iii. There is not much significant change in ROUGE SCORE when increasing epochs on Dolly v2-3B model keeping training data constant
iv. Cannot be much concluded (wrt model size, epoch, training data) from the results that's why we are continuing with next set of experiments below
v. For the case of DialoGPT, it is inconclusive to when comparing rouge score for different epochs

4.2.2.2 Training Dolly Models for comparing runtime and efficiency:

Dolly v 2.0 have different sizes we can finetune them on our custom dataset for performing specific tasks. The approach will be same like above.

DOLLY V2-3B MODEL LOADED USING QLoRA QUANTIZATION

Training Data	Epochs	CPU RAM	GPU RAM	TIME	ROUGE SCORE OLD QUES	ROUGE SCORE NEW QUES
800	1	-	14.7 GB	7MIN	24.73%	14.86%
800	3	-	14.7 GB	18MIN	18.84%	11.44%
400	1	-	14.7 GB	3 MIN	23.12%	11.89%
400	3	-	14.7 GB	9 MIN	21.82%	12.77%
200	1	-	14.7 GB	2 MIN	16.15%	10.26%
200	3	-	14.7 GB	7 MIN	17.14%	11.96%

DOLLY V2-3B MODEL LOADED USING LoRA

Training Data	Epochs	CPU RAM	GPU RAM	TIME	ROUGE SCORE OLD QUES	ROUGE SCORE NEW QUES
800	1	27.8 GB	-	1 HR	24.41%	14.40%
800	3	27.8 GB	-	3 HRS	24.77%	14.24%
400	1	27.8 GB	-	20 MIN	24.35%	14.30%
400	3	27.8 GB	-	1 HR	23.52%	13.34%
200	1	27.8 GB	-	10 MIN	14.30%	14.42%
200	3	27.8 GB	-	30 MIN	24.34%	14.43%

When using QLoRA, it sometimes reduces model accuracy as we are loading model in lower precision weight but not in case of Dolly v2-3B as shown above.

Dolly v2-7B and -12B models using QLoRA quantization were not able to use GPU, hence overloading the CPU in google colab.

When tried to remove quantization to check whether it is restricting gpu usage but then also it didn't use gpu while loading. Hence Dolly v2.0 models architecture are not capable of using gpu while loading with quantization.

There is not much significant change in ROUGE SCORE when increasing epochs on Dolly v2-3B model keeping training data constant

Model runtime increases in the proportion of increase in epoch when keeping the training data constant. Dolly v2 3B takes much less time while running on GPU as compared to CPU

4.2.2.3 Training Different Eleuther AI models for comparing runtime and efficiency:

Dolly models are derived EleutherAI's Pythia model family and fine-tuned on a ~15K record instruction corpus generated by Databricks employees. We are finetuning the EleutherAI's model on our custom dataset and checking which ones perform better than other model sizes. After 2 epochs sometimes the loss values are 'nan' values so for this can I have split the data into training and validation set with 10%spilt size to apply early stopping.

ELUETHER AI GPT NEOX-20B MODEL LOADED USING 4 BIT QLoRA QUANTIZATION

Training Data	Epochs	CPU RAM	GPU RAM	TIME	ROUGE SCORE OLD QUES	ROUGE SCORE NEW QUES
800	1	-	14.7 GB	1 HR 15 MIN	16.95%	12.67%
400	1	-	14.7 GB	36 MIN	17.66%	15.90%
200	1	-	14.7 GB	18MIN	13.28%	10.58%

ELUETHER AI 1.3B MODEL LOADED USING 4 BIT QLoRA QUANTIZATION

Training Data	Epochs	CPU RAM	GPU RAM	TIME	ROUGE SCORE OLD QUES	ROUGE SCORE NEW QUES
800	1	3.5GB	3GB	6 MIN	19.45%	12.95%
800	3	3.5GB	3GB	MIN	22.64%	10.14%
400	1	3.5GB	3GB	4MIN	15.41%	10.75%
400	3	3.5GB	3GB	30 MIN	20.24%	11.72%
200	1	3.5GB	3GB	2 MIN	17.81%	16%
200	3	3.5GB	3GB	10 MIN	19.51%	14.51%

ELUETHER AI pythia 2.8B MODEL LOADED USING 4 BIT QLoRA QUANTIZATION

Training Data	Epochs	CPU RAM	GPU RAM	TIME	ROUGE SCORE OLD QUES	ROUGE SCORE NEW QUES
800	1	4.5GB	4GB	10:30 MIN	17.84%	9.53%
800	3	3.5GB	3GB	MIN	22.46%	11.43%
400	1	4.5GB	4GB	6 MIN	15.47%	13.24%
400	3	3.5GB	3GB	1 hr	18.40%	10.70%
200	1	4.5GB	4GB	1:30 MIN	20.28%	11.69%
200	3	3.5GB	3GB	20 MIN	18.90%	12.78%

ELUETHER AI 1.3B deduped MODEL LOADED USING 4 BIT QLoRA QUANTIZATION

Training Data	Epochs	CPU RAM	GPU RAM	TIME	ROUGE SCORE OLD QUES	ROUGE SCORE NEW QUES
800	1	10GB	-	4 MIN	21.58%	12.91%
400	1	10 GB	-	2 MIN	25.09%	12.45%
200	1	10 GB	-	18MIN	24.36%	12.85%

ELUETHER AI 2.8B deduped MODEL LOADED USING 4 BIT QLoRA QUANTIZATION

Training Data	Epochs	CPU RAM	GPU RAM	TIME	ROUGE SCORE OLD QUES	ROUGE SCORE NEW QUES
800	1	4.9GB	-	10 MIN	22.01%	13.18%
400	1	10 GB	-	6 MIN	16.82%	13.55%
200	1	10 GB	-	3 MIN	15.27%	12.50%

GPT NEOX 20B model can be loaded using quantization but as it is text generation and not build for instruction response work it underperforms than dolly v2-3b. Also after finetuning on more than 1 epochs it overloads the Colab notebook which has 15GB RAM

Larger model cannot be loaded as they are not utilizing GPU ram. These models were utilizing gpus when quantization was removed. Hence quantization would have restricted gpu utilization.

Model runtime increases at larger rate when epochs is increased keeping the epochs constant. Also there is decrease in Rouge Score when the model size is increased keeping the other two parameters constant.

Using larger dataset has lead to better results than smaller dataset. Whereas smaller models are performing better than larger ones.

4.2.3 Further methods to finetune LLM models:

4.2.3.1 LoRA PEFT:

Dolly-v2-3B and EleutherAI models are trained using this method for them the results are added in above experiments itself.

Model	Epochs	ROUGE SCORE NEW QUES	ROUGE SCORE OLD QUES
Google Flan T5 base	150	33.34%	66.25%

THIS MODEL IS FINETUNED BY MANEESH AND OTHER FN MATHLOGIC EMPLOYEES. I HAVE TAKEN THE SCORE TO COMPARE IT WITH OTHER FINETUNING METHODS

4.2.3.2 QLoRA 4-bit Quantization:

Dolly v2-3B and EleutherAI models are loaded by this method. Some of the models were not able to utilize GPU when quantization is used such as Dolly v2-7B & -12B models and EluetherAI’s pythia -7B & -12B. Also Google flan T5 base model cannot be finetuned using Qlora hence it is not possible to compare the results for this method

4.2.3.3 *Finetuning using RLHF:*

Pretrained Google Flan T5 model is used as the finetuned model is not able to train Reward Model due to absence of lm_lead layer in the architecture of Flan T5.

STEPS	TRAIN MAX EPISODE	EVAL INTERVAL	MINI BATCH SIZE	EPOCHS	ROUGE SCORE NEW QUES	ROUGE SCORE OLD QUES
1000	100	10000	256	20	16.42%	6.76%
2000	100	10000	128	50	16.34%	6.78%

Google Flan T5 finetuned with RLHF is giving short answers mostly in yes/no format or one sentence format. Such format of answers are in NEW QUES while OLD QUES answers are more detailed. Hence it is giving better results on NEW QUES than OLD QUES

While for other finetuned models when used for finetuning with human feedback as given in algorithm. It is giving the following errors.

i. Flan T5 base model finetuned using LoRA

```
agent = actor.agent_ppo(update_interval=5, minibatch_size=256, epochs=20)  # 5, minibatch_size=1024,

/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_as
  and should_run_async(code)
─────────────────────── Traceback (most recent call last) ───────────────────────
 in <cell line: 2>:2

 /usr/local/lib/python3.10/dist-packages/textrl/actor.py:62 in __init__

           elif 'encoder' in parents:  # t5
               transformers_model = model.encoder
           else:
               raise ValueError('model not supported')

           if unfreeze_layer_from_past > 0:
               self.middle_model = HFModelListModule(list(transformers_model.children())

ValueError: model not supported
```

It is not able to detect encoder in model even though the architecture consist of encoder

ii. Dolly-v2-3B model finetuned using LoRA

```
[ ]  agent = actor.agent_ppo(update_interval=5, minibatch_size=256, epochs=20)  # 5, minibatch_size=1024,

/usr/local/lib/python3.10/dist-packages/transformers/models/gpt_neox/modeling_gpt_neox.py:235: UserWa
  attn_scores = torch.where(causal_mask, attn_scores, mask_value)
─────────────────────── Traceback (most recent call last) ───────────────────────
 in <cell line: 4>:4

 /usr/local/lib/python3.10/dist-packages/textrl/actor.py:46 in __init__

           self.device = torch.device("cuda:{}".format(gpu_id))
           self.model = model
           self.obs_size = model.config.hidden_size
           self.converter = self.model.lm_head
           self.act_deterministically = act_deterministically
           self.temperature = temperature
           self.top_k = top_k

 /usr/local/lib/python3.10/dist-packages/torch/nn/modules/module.py:1614 in __getattr__

               modules = self.__dict__['_modules']
               if name in modules:
                   return modules[name]
           raise AttributeError("'{}' object has no attribute '{}'".format(
               type(self).__name__, name))

       def __setattr__(self, name: str, value: Union[Tensor, 'Module']) -> None:

AttributeError: 'GPTNeoXForCausalLM' object has no attribute 'lm_head'
```

It is not able to detect lm_head in model as the architecture consist of embed_out layer as shown below

```
PeftModelForCausalLM(
  (base_model): LoraModel(
    (model): GPTNeoXForCausalLM(
      (gpt_neox): GPTNeoXModel(
        (embed_in): Embedding(50280, 2560)
        (emb_dropout): Dropout(p=0.0, inplace=False)
        (layers): ModuleList(
          (0-31): 32 x GPTNeoXLayer(
            (input_layernorm): LayerNorm((2560,), eps=1e-05
            (post_attention_layernorm): LayerNorm((2560,),
            (post_attention_dropout): Dropout(p=0.0, inplac
            (post_mlp_dropout): Dropout(p=0.0, inplace=Fals
            (attention): GPTNeoXAttention(
              (rotary_emb): GPTNeoXRotaryEmbedding()
              (query_key_value): Linear4bit(
                in_features=2560, out_features=7680, bias=T
                (lora_dropout): ModuleDict(
                  (default): Dropout(p=0.05, inplace=False)
                )
                (lora_A): ModuleDict(
                  (default): Linear(in_features=2560, out_f
                )
                (lora_B): ModuleDict(
                  (default): Linear(in_features=4, out_feat
                )
                (lora_embedding_A): ParameterDict()
                (lora_embedding_B): ParameterDict()
              )
              (dense): Linear4bit(in_features=2560, out_fea
              (attention_dropout): Dropout(p=0.0, inplace=F
            )
            (mlp): GPTNeoXMLP(
              (dense_h_to_4h): Linear4bit(in_features=2560,
              (dense_4h_to_h): Linear4bit(in_features=10240
              (act): GELUActivation()
            )
          )
        )
        (final_layer_norm): LayerNorm((2560,), eps=1e-05, e
      )
      (embed_out): Linear(in_features=2560, out_features=50
    )
  )
)
```

The lm_head layer is typically a linear layer with a softmax activation function. This layer takes the output of the previous layer, which is a sequence of hidden states, and generates a probability distribution over the vocabulary. The output of the lm_head layer is a vector of probabilities, one for each word in the vocabulary. Hence lm_head linear layer has weights of size embedding_size,vocab_size,

The embed_out layer is a linear layer with a ReLU activation function. This layer takes the output of the previous layer, which is a sequence of hidden states, and produces a sequence of embeddings. The embeddings are typically a fixed-size vector for each word in the vocabulary. Hence embedding matrix has a size (vocab_size, embedding_size).

iii. EleutherAI pythia 2.8b model finetuned using LoRA

EleutherAI/pythia-2.8b

```
#actor = TextRLActor(env, model, tokenizer)
actor = TextRLActor(env, model, tokenizer)          # repetition penalty from CTRL paper (https://arxiv

agent = actor.agent_ppo(update_interval=5, minibatch_size=256, epochs=20)  # 5, minibatch_size=1024, e
```

```
Traceback (most recent call last)
in <cell line: 2>:2

/usr/local/lib/python3.10/dist-packages/textrl/actor.py:46 in __init__

   43         self.device = torch.device("cuda:{}".format(gpu_id))
   44         self.model = model
   45         self.obs_size = model.config.hidden_size
❱  46         self.converter = self.model.lm_head
   47         self.act_deterministically = act_deterministically
   48         self.temperature = temperature
   49         self.top_k = top_k

/usr/local/lib/python3.10/dist-packages/torch/nn/modules/module.py:1614 in __getattr__

  1611             modules = self.__dict__['_modules']
  1612             if name in modules:
  1613                 return modules[name]
❱ 1614         raise AttributeError("'{}' object has no attribute '{}'".format(
  1615             type(self).__name__, name))
  1616
  1617     def __setattr__(self, name: str, value: Union[Tensor, 'Module']) -> None:
AttributeError: 'GPTNeoXForCausalLM' object has no attribute 'lm_head'
```

Same is the error as in dolly v2 3b mentioned above due to presence of embed_out layer

5 Conclusions

DialoGPT is conversational as it is taking user's chat history. ROUGE SCORE of DialoGPT is not at par with google flan-t5. But it is able to retain the conversation memory for 4-5 conversations. Also it is giving better results than Dolly v2-3B model. When using Langchain framework to incorporate the conversation memory to the finetune models such as DialoGPT, Flan T5 base and Dolly v2 3b model are not able to retain conversation history while answering as well as the answers generated are not correct. Whereas EleutherAI pythia 1.3B model was able to retain chat history while answer generating but due to low rouge score of finetuned model the answers are not accurate.

For Question Answering, when finetuning Dolly-v2-3B model with LoRA is not able to generate better Rouge Score than DialoGPT. Further conducted experments for different model size vs epoch vs training data. It can be concluded that there is not much significant change in ROUGE SCORE when increasing epochs on Dolly v2-3B model keeping training data constant. Cannot be much concluded (wrt model size, epoch, training data) from the results that's why we have continued with next set of experiments of comparing Dolly and EleutherAI models. Only Dolly v2 3b model can be trained that on GPU using quantization and CPU. There is no significant change in rouge score by increasing epochs when experimented with different training data size. Model runtime increases in the proportion of increase in epoch but in the case of EleutherAI models model runtime increases at faster rate as we increase the epochs. Dolly v2-7B & -12B architecture are not compatible with 4 bit QLoRA quantization using GPU.

GPT NEOX 20B model can be loaded using quantization but as it is text generation and not build for instruction response work hence it underperforms than dolly v2-3b. Other larger models of EleutherAI model cannot be loaded as they are not utilizing GPU ram such as 7B and 12B models. These models were utilizing gpus when quantization was removed. Hence quantization would have restricted gpu utilization. Insights from ROUGE SCORE are inconclusive as there is not much distinguishable difference in model performance by varying model size wrt constant training data.

Finetuning with RLHF, it is not possible to finetune with human feedback when we are loading model such as Flan T5 which is finetuned on ICICI data. Instead pretrained Flan T5 base model is used, it is giving the Rouge score values on NEW QUES greater than OLD QUES as the answer generated are mainly shorter. Unlike the answers in OLD QUES set expect the generated answer to be much more detailed for providing more information to user. Also the output is not better than Flan T5 base model finetuned with LoRA.

6 Further Work

Finetuned DialoGPT-small model is able to retain conversation history further we can try finetuning larger DialoGPT models and increasing the context conversations while finetuning. Some of the finetuned models are not working when used with Langchain which reduces the accuracy of the model output, in this we have try remove this issue. Question Answering on Dolly v2 3B and EleutherAI GPT NEOX 20B model are limited due to GPU overloading, if working on higher GPU hardware you can try finetuning larger models with larger dataset. Reinforcement learning with Human Feedback tend to give better output for this method was used in ChatGPT but in our case the finetuned methods are not working which we can further work on.

7 References

1. DialoGPT: Large-Scale Generative Pre-training for Conversational Response Generation; https://towardsdatascience.com/dialogpt-large-scale-generative-pre-training-for-conversational-response-generation-5ceb783428dc
2. Using DialoGPT For Conversational Response Generation: https://cobusgreyling.medium.com/using-dialogpt-for-conversational-response-generation-559e2a13b191
3. Fine-Tuning DialoGPT-Medium on Daily Dialog Dataset: A Step-by-Step Guide: https://blog.gopenai.com/fine-tuning-dialogpt-medium-on-daily-dialog-dataset-a-step-by-step-guide-4eaecc1b9323
4. Zhao, W.X. *et al.* (2023) *A survey of large language models*, *arXiv.org*. Available at: https://arxiv.org/abs/2303.18223 (Accessed: 19 July 2023).
5. *Prompting for a conversation: How to control a dialog model?* Available at: https://aclanthology.org/2022.cai-1.1.pdf (Accessed: 18 July 2023).
6. Meet Dolly: How Databricks Finetuned a Two-Year-Old LLM to Follow Instructions like ChatGPT: https://pub.towardsai.net/meet-dolly-how-databricks-finetuned-a-two-year-old-llm-to-follow-instructions-like-chatgpt-faa8c839eb9f
7. *Free Dolly: Introducing the world's first truly open instruction-tuned LLM* (no date) *Databricks*. Available at: https://www.databricks.com/blog/2023/04/12/dolly-first-open-commercially-viable-instruction-tuned-llm (Accessed: 19 July 2023).
8. *Databricks/Dolly-V2-3B · hugging face* (no date) *databricks/dolly-v2-3b · Hugging Face*. Available at: https://huggingface.co/databricks/dolly-v2-3b (Accessed: 19 July 2023).
9. *EleutherAI/pythia-12b · hugging face* (no date) *EleutherAI/pythia-12b · Hugging Face*. Available at: https://huggingface.co/EleutherAI/pythia-12b (Accessed: 19 July 2023).
10. Raschka, S. (2023) *Parameter-efficient LLM Finetuning with low-rank adaptation (Lora)*, *Sebastian Raschka, PhD*. Available at: https://sebastianraschka.com/blog/2023/llm-finetuning-lora.html (Accessed: 19 July 2023).
11. *Low-rank adaptation of Large Language Models (Lora)* (no date) *Low-Rank Adaptation of Large Language Models (LoRA)*. Available at: https://huggingface.co/docs/diffusers/training/lora (Accessed: 19 July 2023).

12. Sean, X. (2023) *Fine-tuning llms made easy with Lora and generative AI-Stable Diffusion Lora*, *Medium*. Available at: https://xiaosean5408.medium.com/fine-tuning-llms-made-easy-with-lora-and-generative-ai-stable-diffusion-lora-39ff27480fda (Accessed: 19 July 2023).
13. Ben Dickson *et al.* (2023) *What is low-rank adaptation (Lora)?*, *TechTalks*. Available at: https://bdtechtalks.com/2023/05/22/what-is-lora/ (Accessed: 19 July 2023).
14. Sean, X. (2023a) *Fine-tuning llms made easy with Lora and generative AI-Stable Diffusion Lora*, *Medium*. Available at: https://xiaosean5408.medium.com/fine-tuning-llms-made-easy-with-lora-and-generative-ai-stable-diffusion-lora-39ff27480fda (Accessed: 19 July 2023).
15. *Making LLMS even more accessible with bitsandbytes, 4-bit quantization and Qlora* (no date) *Hugging Face – The AI community building the future.* Available at: https://huggingface.co/blog/4bit-transformers-bitsandbytes (Accessed: 19 July 2023).
16. Dettmers, T. *et al.* (no date) *Qlora: Efficient finetuning of quantized llms*, *– arXiv Vanity*. Available at: https://www.arxiv-vanity.com/papers/2305.14314/ (Accessed: 19 July 2023).
17. Artidoro (no date) *Artidoro/Qlora: Qlora: Efficient finetuning of quantized llms*, *GitHub*. Available at: https://github.com/artidoro/qlora (Accessed: 19 July 2023).
18. *Making LLMS even more accessible with bitsandbytes, 4-bit quantization and Qlora* (no date a) *Hugging Face – The AI community building the future.* Available at: https://huggingface.co/blog/4bit-transformers-bitsandbytes (Accessed: 19 July 2023).
19. S.L.U., A. (2023) *Bringing LLM fine-tuning and RLHF to everyone*, *Open-source data curation platform for LLMs*. Available at: https://argilla.io/blog/argilla-for-llms/ (Accessed: 19 July 2023).
20. "What is reinforcement learning from human feedback (RLHF)?," *TechTalks*, 16-Jan-2023. [Online]. Available: https://bdtechtalks.com/2023/01/16/what-is-rlhf/ . [Accessed: 19-Jul-2023]
21. Malhotra, Tanya. "5 Reasons Why Large Language Models (LLMs) Like ChatGPT Use Reinforcement Learning Instead of Supervised Learning for Finetuning." MarkTechPost, 5 Mar. 2023, https://www.marktechpost.com/2023/03/05/5-reasons-why-large-language-models-llms-like-chatgpt-use-reinforcement-learning-instead-of-supervised-learning-for-finetuning/
22. Fine-Tuning 20B LLMs with RLHF on a 24GB Consumer GPU. https://huggingface.co/blog/trl-peft . Accessed 19 July 2023.
23. Huyen, Chip. "RLHF: Reinforcement Learning from Human Feedback." Chip Huyen, 2 May 2023, https://huyenchip.com/2023/05/02/rlhf.html .
24. Baghdadlian, Serop. "Fine-Tuning Large Language Models with LangChain." Medium, 17 Apr. 2023, https://ai.plainenglish.io/fine-tuning-large-language-models-with-langchain-1cf453349001 .
25. Baghdadlian, Serop. "FineTuning Local Large Language Models on Your Data Using LangChain." Medium, 24 May 2023, https://pub.towardsai.net/finetuning-local-large-language-models-on-your-data-using-langchain-9229da66ad9b .
26. Fine-Tuning ChatGPT with Lang-Chain | CreativeDev. https://www.thecreativedev.com/finetuning-chatgpt-with-langchain/, https://www.thecreativedev.com/finetuning-chatgpt-with-langchain/ ,Accessed 19 July 2023.
27. Greyling, Cobus. "Using LangChain To Create Large Language Model (LLM) Applications Via HuggingFace." Medium, 3 May 2023, https://cobusgreyling.medium.com/langchain-creating-large-language-model-llm-applications-via-huggingface-192423883a74.
28. Aman's AI Journal • Reinforcement Learning from Human Feedback (RLHF). https://aman.ai/primers/ai/RLHF/#reinforcement-learning-vs-supervised-learning-for-finetuning . Accessed 19 July 2023.