#1 Friends Fan Persona Chatbot

Repo: https://github.com/vanshigupta04/Persona-LLM-Chatbot-For-QA

Team 19 Members

Advaith Shyamsunder Rao	asr209
Falgun Malhotra	fm466
Vanshita Gupta	vg422

Introduction

The project primarily focuses on building a QA Chatbot for the persona ***#1 Friends Fan*** using techniques such as fine-tuning Large Language Models and Retrieval Augmented Generation (RAG). This persona will be a bot well versed with information about the TV show *Friends (1994-2004).* As part of the model fine-tuning, PEFT techniques such as LoRA and QLoRA have been devised for efficient and quicker weight updates. The performance of the persona LLM was measured using MCQ as well as Free Response style evaluation, and benchmarked against the base LLM model.

Research Paper Summary

Paper Title: QLORA: Efficient Finetuning of Quantized LLMs

Summary: The paper introduces QLORA, an efficient finetuning approach that enables the finetuning of a 65B parameter model on a single 48GB GPU while maintaining full 16-bit finetuning task performance. The paper discusses the key innovations introduced by QLORA to conserve memory without compromising performance, including the use of 4-bit NormalFloat, Double Quantization, and Paged Optimizers. The document also presents the Guanaco model family, which outperforms all previously openly released models on the Vicuna benchmark. Additionally, the paper explores the implications of applying QLORA to finetune models of different scales and types, highlighting its impact on instruction following and chatbot performance across various datasets and model categories.

Dataset

For this persona the ConvoKit Friends corpus has been used, which contains data across the 10 seasons of the show. There are 236 episodes, 3,107 scenes (conversations), 67,373 utterances, and 700 characters (users). The raw corpus of text for each persona has been put through the data preparation pipeline and necessary data preprocessing and cleaning has been performed.

Data Preparation and Analysis

The ConvoKit Friends corpus framework provides necessary utilities to load and preprocess the friends corpus. The loaded data is on per utterance level. utterance_ids are indexed by the id sXX_eYY_cZZ_uAAA, where XX denotes the season (e.g. 01), YY denotes the episode (e.g. 01), ZZ denotes the conversation (e.g. 01) and AAA is the identifier for individual utterances. The text column contains the utterance (dialogues). The rows with no dialogue text are dropped.

	conversation_id	season	episode	scene	utterance_id	text	speaker
0	s01_e01_c01_u001	s01	e01	c01	s01_e01_c01_u001	There's nothing to tell! He's just some guy I ...	Monica Geller
1	s01_e01_c01_u001	s01	e01	c01	s01_e01_c01_u002	C'mon, you're going out with the guy! There's ...	Joey Tribbiani
2	s01_e01_c01_u001	s01	e01	c01	s01_e01_c01_u003	All right Joey, be nice. So does he have a hum...	Chandler Bing
3	s01_e01_c01_u001	s01	e01	c01	s01_e01_c01_u004	Wait, does he eat chalk?	Phoebe Buffay
5	s01_e01_c01_u001	s01	e01	c01	s01_e01_c01_u006	Just, 'cause, I don't want her to go through w...	Phoebe Buffay

The dataset for LLM fine tuning is prepared by creating another column script where the speaker and dialogues are merged on the conversation_id level. The grouped script column is passed to the LLM model for fine tuning.

	conversation_id	season	episode	scene	script
0	s01_e01_c01_u001	s01	e01	c01	[Monica Geller: There's nothing to tell! He's ...
1	s01_e01_c02_u001	s01	e01	c02	[Monica Geller: Now I'm guessing that he bough...
2	s01_e01_c03_u001	s01	e01	c03	[Phoebe Buffay: Love is sweet as summer shower...
3	s01_e01_c04_u001	s01	e01	c04	[Ross Geller: I'm supposed to attach a bracket...
4	s01_e01_c05_u001	s01	e01	c05	[Monica Geller: Oh my God!, Paul the Wine Guy:...

The following is an analysis of the number of utterances in each conversation:

```
count    3099.000000
mean       19.783801
std        16.273398
min         1.000000
25%        10.000000
50%        17.000000
75%        25.000000
max       255.000000
Name: script, dtype: float64
```

Model Selection

For the execution of this project, the selection of the LLAMA2 model, recognized as state-of-the-art in large language models (LLMs), was made as the foundational LLM. This model, developed and released by Meta, is made available under an open license for both research and commercial applications.

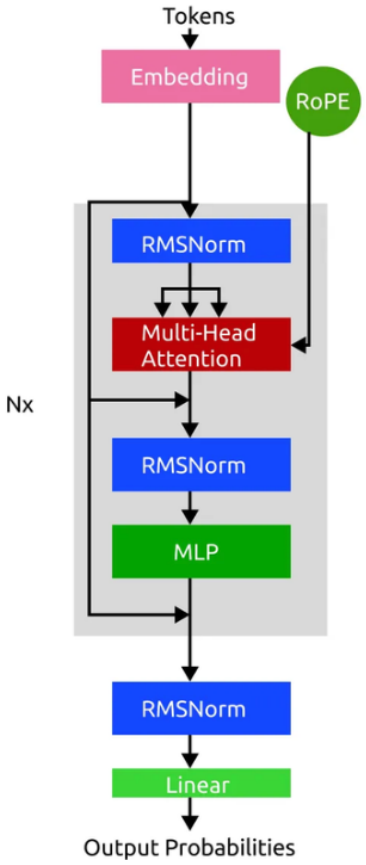


Figure: LLama Architecture

About the LLama2 Model:

Llama 2, akin to its predecessor, the original Llama model, is constructed upon the foundational Google transformer architecture but incorporates several enhancements. These enhancements within Llama encompass the integration of RMSNorm pre-normalization, drawing inspiration from the methodology seen in GPT-3, the utilization of a SwiGLU activation function inspired by Google's PaLM, the adoption of a multi-query attention mechanism in place of the conventional multi-head attention, and the incorporation of rotary positional embeddings (RoPE), inspired by GPT Neo.

During the training phase, Llama was optimized utilizing the AdamW optimizer. Notable distinctions in Llama 2 compared to its predecessor include an expanded context length, accommodating 4096 tokens as opposed to the original 2048 tokens, and the utilization of grouped-query attention (GQA) in lieu of multi-query attention (MQA) within the two larger models.

About LLama2 Chat Model:

Llama 2 Chat LLMs have been refined and tailored for dialogue-specific applications, showcasing superior performance compared to various open-source chat models across multiple benchmark tests. According to Meta's assessments involving human evaluations pertaining to usefulness and safety, the company suggests that Llama 2 could potentially serve as an adequate replacement for proprietary, closed-source models.

Overcoming Challenges:

The primary challenge arose from the considerable size of the LLAMA2 model, comprising 7 billion parameters, leading to significant demands on storage and computational resources. To address these constraints, an investigation was conducted to optimize the deployment of large language models on devices with limited storage and computational capabilities. A quantized version of the LLAMA2 model, involving the reduction of weight precision to diminish the LLM size while maintaining satisfactory performance, was chosen. Various quantization data types, including fp16, int8, and int4, were examined, along with a comparison of compressed model file formats, namely GPTQ, GGML, and GGUF.

Considering the limitations of available computational resources, a determination was made in favor of a hybrid CPU-GPU hosting approach for optimal efficiency. Given the suboptimal performance of GPTQ models on CPUs, the GGUF file format emerged as the preferred choice.

The selected base LLM model, denoted as llama-2-7b-chat.Q4_K_M.gguf, with a maximum RAM requirement of 6.58 GB, is accessible at:
https://huggingface.co/TheBloke/Llama-2-7B-Chat-GGUF

Model Hosting

In terms of model hosting, various frameworks and libraries were explored, including:

1. OpenLLM
2. Llama.cpp
3. GPT4All

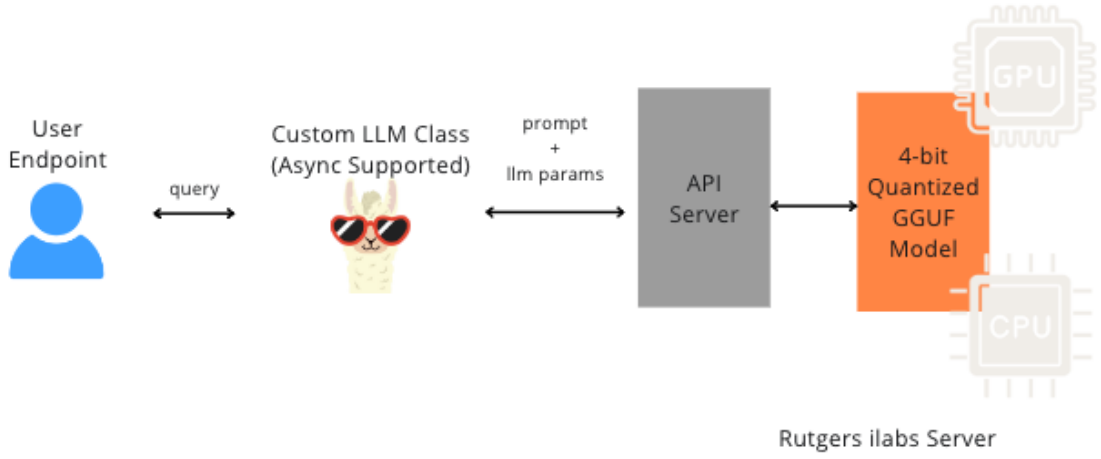


Figure: Hosted Llama2 model on Rutgers ilabs server

The initial experimentation involved the evaluation of model hosting using OpenLLM. However, encountered challenges related to package compatibility prompted the exploration of alternative methodologies. Although GPT4all is constructed upon Llama.cpp, investigations revealed user-reported compatibility issues, leading to the decision to employ Llama.cpp executables for hosting the model on the Rutgers iLabs server. The establishment of an API endpoint on the

iLabs machine was achieved by leveraging Llama.cpp's server module, which was exposed to facilitate Large Language Model (LLM) inference. The entirety of the model hosting process has been automated through shell scripts, conveniently accessible in the 'llm_hosting' folder within the Git repository. Comprehensive instructions for execution can be found therein.

A Custom LLM Wrapper Class was written in python. This allowed asynchronous calls to the Model. It also provides a method to format a user query into a Llama2 compatible prompt.

Experiment 1: Baseline

Parameters Used for Inference:

The following parameters were used during the inference from Llama2 model throughout the project:

- Top-p Sampling: 0.9
- Temperature: 0.5
- Repetition penalty: 1.5

Experiment 2 : Finetuning

Considering the huge size of Llama2 model i.e. 7B params, standard finetuning on all the model parameters was not feasible due to high storage and compute demand.

Thus, Experimentation was done with 2 Parameter Efficient Fine-Tuning (PEFT) Finetuning approaches:

- LoRA Finetuning on NF-4 Quantized Llama-2-7b-chat GGUF model
- QLoRA Finetuning on FP-16 Llama-2-7b-chat

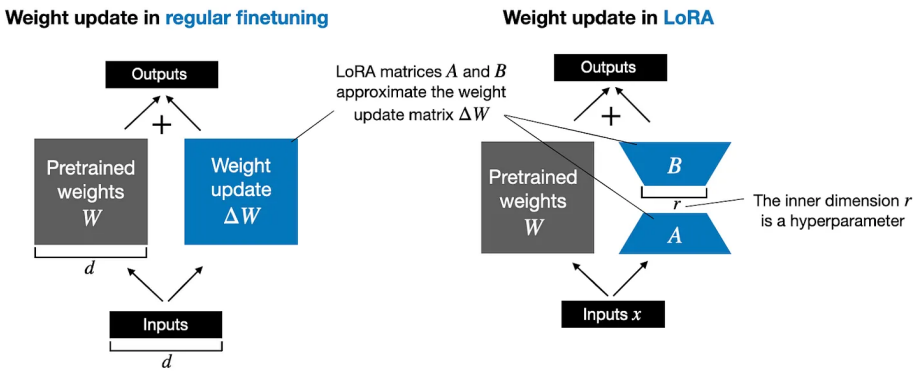


Figure: LoRA fine tuning

While LoRA helps in reducing the storage requirements, you would still need a large GPU to load the model into the memory for LoRa training. In QLoRA, you first quantize the LLM and then perform LoRa training.

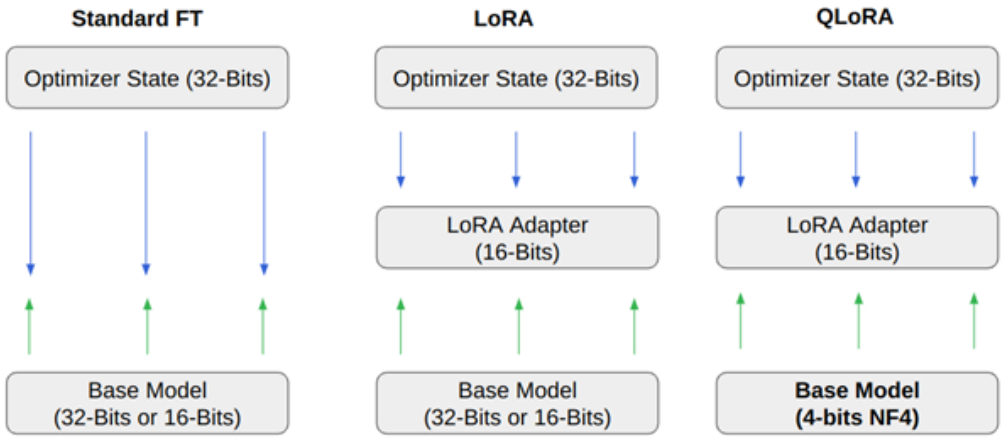


Figure: Comparison of Finetuning Approaches

1) LoRA: Was performed using No-Code Approach

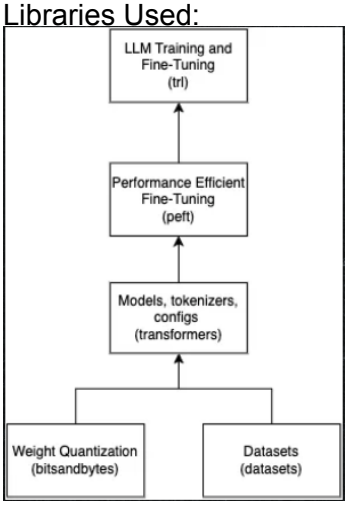
For this, LLama.cpp finetune executables were used along with customized configuration parameters to train, generate adapters and merge them to the original model. (finetune.sh, convert.sh, server.sh)

2) QLoRA: Was performed using Code Approach

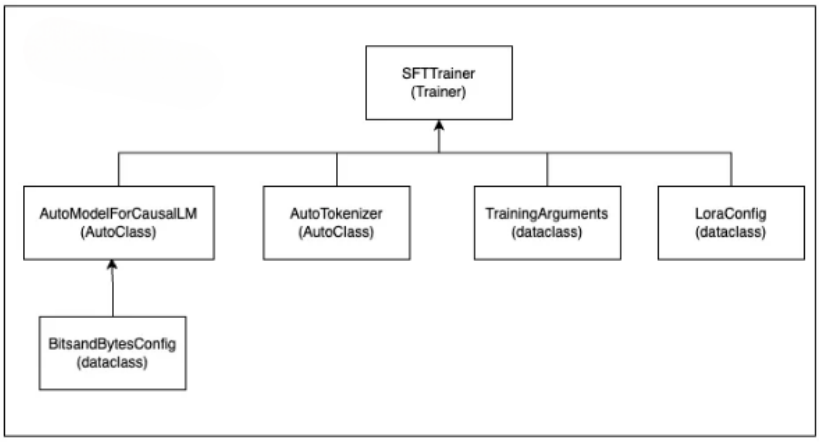
The QLoRa finetuning was done on a ipynb notebook which can be found under finetuning folder in the Git repository.

Code Overview:

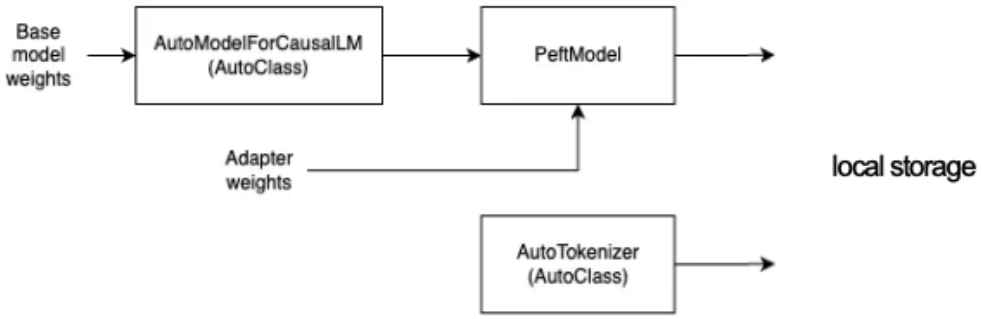
Libraries Used:



Fine-tuning Process:



Merging Model weights and saving final model:



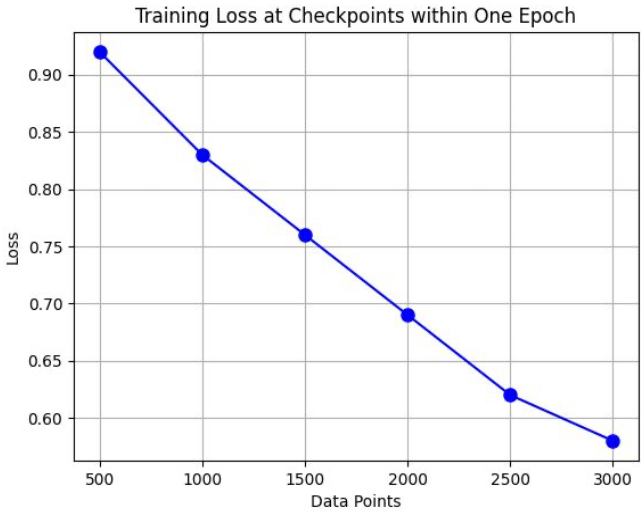
Configuration Parameters Used for LoRA and QLoRA:

Some of the important configuration parameters used in finetuning are as follows:

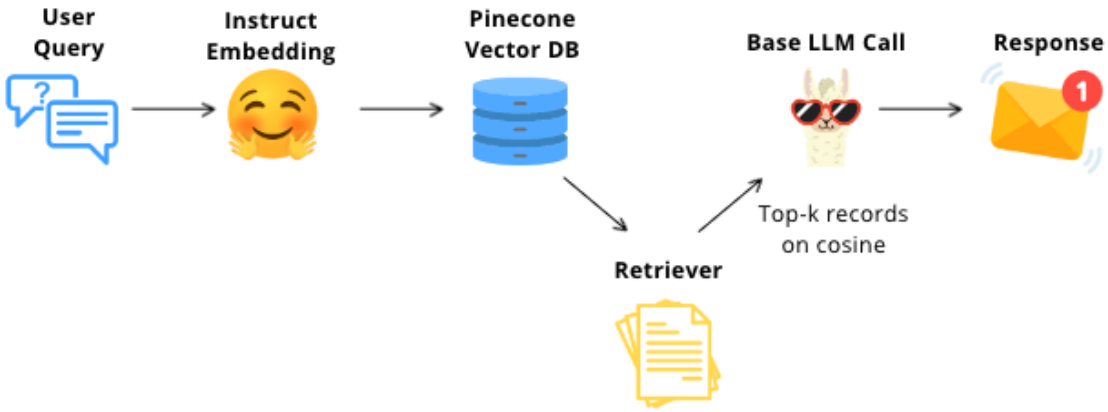
- LoRA attention Dimension: --lora_r N : 4
- Alpha Parameter for LoRA scaling: --lora_alpha N : 4
- Dropout probability for LoRA layers: --lora_dropout N : 4
- Model Context Size: –n_ctx N : 4096
- Batch size: –batch N : 1
- Adam learning rate : –adam-alpha F : 2e-5
- Number of epochs : –epochs N : 1
- Frequency base for ROPE : –rope-freq-base F : 1
- Rope Frequency Scale: –rope-freq-scale F : 1

NOTE: Due to limited compute, the finetuning could be done only for one epoch.

Training Loss:



Experiment 3 : Retrieval Augmented Generation (RAG)



In this experiment, the Retrieval Augmented Generation-based LLaMA 2 model has been deployed. Fine-tuning a large language model can be a strenuous process, requiring heavy loads of computing resources as well as memory. The experiment aims to see if instead of fine-tuning the LLM, providing the LLM with some context relevant to the user question, does the model perform better?

The context mentioned above was provided as the top 4 retrieved dialogues from the training dataset based on cosine similarity between the embeddings of the user search query and each of the dialogues. After retrieving the top 4 dialogues for a search query, the LLM prompt is appended with the top 4 dialogues, to assist the LLM provide better answers for user questions.

Embeddings

Two different vector embedding models were experimented with to embed the dialogues,

1. Instruct Embeddings (Huggingface) - 768 dimensions
2. Cohere Embeddings - 384 dimensions

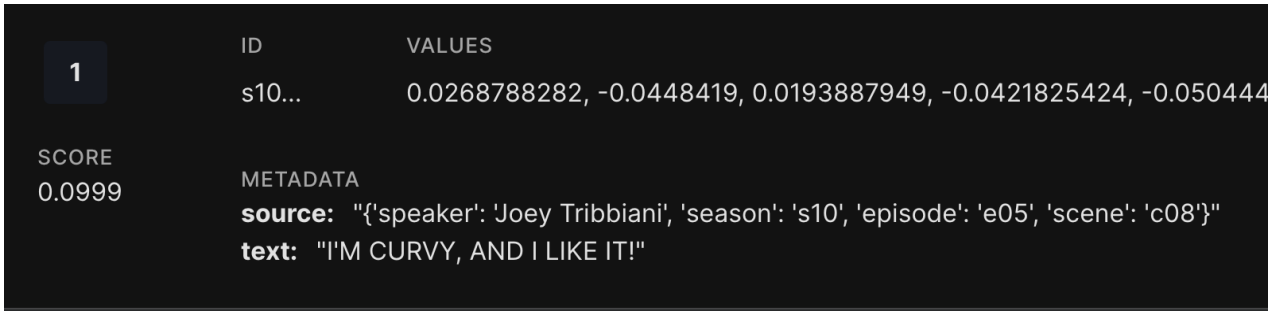
The following observations were noted for each of the embeddings,

Embedding Model	**Name**	**Dimensions**	**Upsert Time (rows/sec)**
Instructor	HF-Instruct	768	40.2
Cohere	Embed-Small	384	7.33

Upon the above observations, Instruct Embedding has been chosen as the optimal embedding model task based on dimension size and efficacy.

Vector Database

A Pinecone vector database is populated with the Instruct Embeddings of all utterances, as well as the metadata for each utterance such as Text, Speaker, Season, Episode, and Scene. The data stored on the knowledge base is indexed on a conversation level, as using conversation as the index helps efficiently search and update the database. The upsert operation for the 67K rows of data was observed to work at the rate of 13 rows per second.



Evaluation

A curated evaluation dataset is used to validate the #1 Friends Fan persona. Friends Trivia from different sources have been parsed to get MCQ Style questions. The trivia question set consists of several easy to hard questions with the respective answers. The dataset has been cleaned to consistently keep 4 option choices for every question. In case of a question containing less than 4 options, *None of the Above* and *All of the Above* has been added. The columns in the evaluation dataset are,

1. **Question:** Question to be evaluated
2. **Options:** 4 answer choices out of which 1 is the ground truth answer
3. **Reference**: The ground truth answer in the format - A,B,C, or D

	Question	Options	Correct Answer	Correct_Answer_no
0	How many times was Ross legally divorced? \n	['Twice', 'Three times', 'Five times', 'Six ti...	Three times	B
1	Where did Carol first meet Susan? \n	['In college', 'At work', 'At the gym', 'At Ce...	At the gym	C
2	How did Susan and Ross come up with Ben's name?	["It was the doctor's name \n", 'They both ha...	It was on the janitor's name tag	D
3	What were Ben's first words? \n	['Hi', 'Bye', 'Mom', 'Dumb']	Hi	A
4	How long did Ross and Emily date before they g...	['14 days', '6 weeks \n', 'A year \n', '3 mo...	6 weeks \n	B

MCQ Evaluation

As part of MCQ evaluation, the evaluator outputs the following features for the evaluation of a question,

1. **Answer:** LLM Response in the format - A, B, C or D
2. **Correct Format Rate:** The number of times the LLM output follows the desired answer format.

A row of the output of running MCQ evaluation would look like this,


```
{
    "question": "Which one of Rachel's sisters was played by Reese Witherspoon?  \n",
    "option":
    [
        "Amy",
        "Jill",
        "None of the above",
        "All of the above"
    ],
    "reference": "B",
    "answer": "C",
    "correct_format_rate": 1.0,
```


Across the evaluation dataset the above features are used to calculate the MCQ evaluation metrics,

1. **MCQ Accuracy:** Ratio of number of correct answers in the evaluation dataset.
2. **Mean Correct Format Rate**: Mean of the Correct Format Rate across the dataset.

Prompt Engineering

MCQ Evaluation has been performed for the base LLaMA 2 model on the dataset and results have been noted through 2 different experiments

1. **Only-System Prompt:** Simple system prompt used to return the LLM response for a question set.
2. **Engineered Prompt:** Utilizing several prompt engineering techniques to format the LLM response to match the desired output style (answer in A, B, C, or D only).

As part of prompt engineering the following techniques have been used,

1. **Few-Shot Prompting:** The prompt to the LLM call is provided with two examples of MCQ evaluation helping add a specific output format for our LLM response.

2. **Chain-of-Thought:** The prompt is provided with the necessary steps to evaluate the response the right way and provide the right answer in the desired format.
3. **Self Consistency:** The LLM response for a question is sampled *N=5* times and the highest occurring answer has been noted down as the LLM response for the question.

```
system_instruction = """
    You are a huge fan of the TV Show Friends. You will be given a QUESTION and four OPTIONS.
    I want you to ANSWER the QUESTION with the following steps.

    Evaluation Steps:
    1. Read the QUESTION carefully.
    2. Choose the correct OPTION from OPTIONS best of your knowledge.
    3. Output the ANSWER which is a single alphabet from A, B, C, D which is the right OPTION for the QUESTION
    4. The Output format for each OPTION is
        for A: 'ANSWER: A'
        for B: 'ANSWER: B'
        for C: 'ANSWER: C'
        for D: 'ANSWER: D'

    Here are a few Examples for how I expect the answer to be.
    Examples:

    {
        QUESTION: What is the name of Ross and Rachel's daughter,
        OPTIONS:
            A. Emma
            B. Delilah
            C. Deborah
            D. Claire
        ANSWER: A
    },

    {
        QUESTION: What is Chandler Bing's Middle Name,
        OPTIONS:
            A. Meredith
            B. Muriel
            C. Richard
            D. Robert
        ANSWER: B
    }

    Based on the above Evaluation Steps and Examples now ANSWER the QUESTION I give you
```

Figure: MCQ Evaluation Prompt

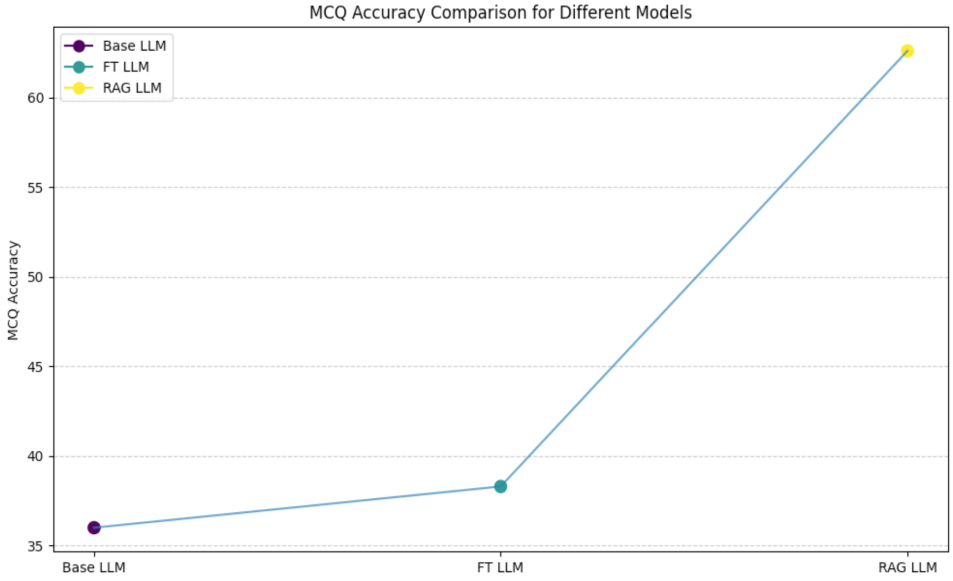
Free Response Evaluation

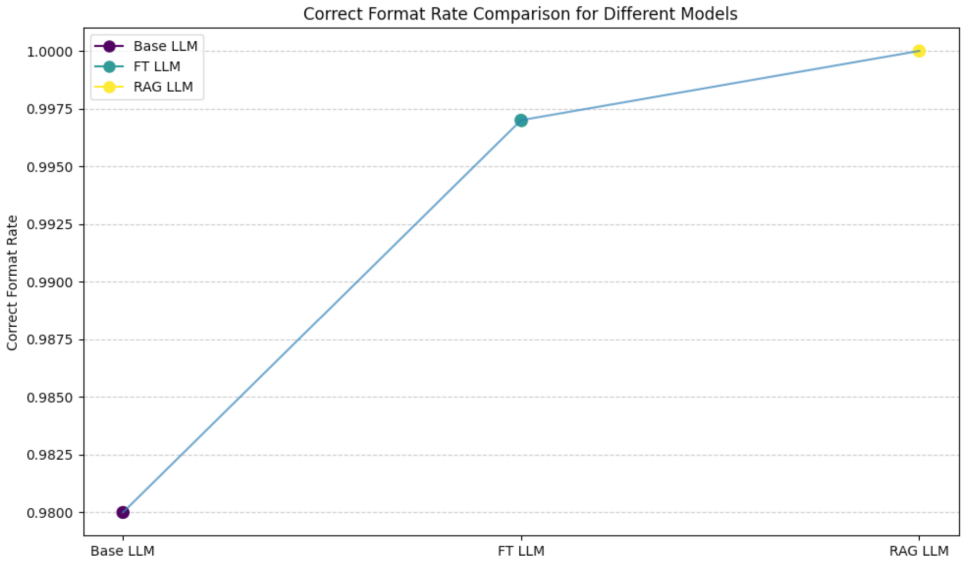
Calculates average score which comprises the following metrics.

- Bleu
- Rouge-1
- ChrF
- Jaccard Similarity

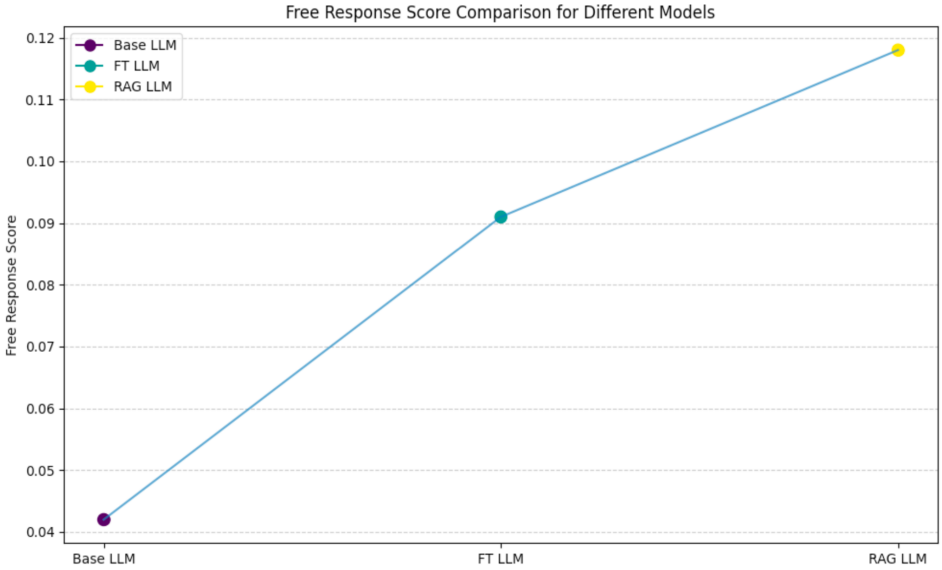
Results

MCQ Evaluation





Free Response Evaluation



Future Improvements

- Compute!!! Increasing Number of epochs and choosing a larger LoRA dimension for fine-tuning.
- Fine-tuning was done on dialogue data and not on QA data(Instruction Fine-tuning)
- Increase the (context) top-k value for the retriever from 4 to 128
- The Feedback loop
    - Self-reflexion
    - Self-consistency
- RLHF

Contributions

1) Advaith Shyamsunder Rao: Retrieval Augmented Generation (RAG), MCQ Evaluation.
2) Falgun Malhotra: Model Selection, Hosting and Finetuning.
3) Vanshita Gupta: Data Preparation (Fine Tuning dataset and MCQ-style QA dataset), Vector Database Setup and Upsert, Free response Evaluation.

References

1. Evaluation:
    a. MMLU MCQ Evaluation
    b. Ngram Overlap Metrics
2. Prompt Engineering:
    a. Prompt Engineering
3. Model Hosting:
    a. GPT4ALL
    b. LLaMA CPP
    c. OpenLLM
4. Datasets:
    a. Friends Conversation Corpus
5. Model Fine Tuning
    a. Parameter Efficient Fine Tuning
    b. QLoRA