Athina AI - Internship Assessment

Thought Process

Dataset Construction

To create the dataset for testing the RAG-powered chatbot, I have used the following steps:

1. **PDF Selection**: I chose a PDF document that is rich in content and likely to generate meaningful questions and general, so I chose a document related to Stock market analysis
   Link to the document

2. **Data Extraction**: Generally we use Python libraries to extract the text from each page of the PDF. Libraries such as `PyPDF2` or `pdfminer.six` were used to accomplish this.
But I have used LlamaParser (it is a dedicated tool used for extracting data from pdf in RAG) to extract the pdf content into a text file(.txt)

3. **Question Generation for testing**: To generate relevant questions based on the extracted text, I used a pre-trained language model(Gemini) . By implementing tools such as CrewAI Agents, I automated the process of reading each page and generating questions that a user might typically ask. This step involves generating both factual and inferential questions to ensure the dataset is diverse.

4. **Dataset Compilation**: The generated questions were extracted using agents and the relevant answers are generated at the time when testing questions are being processed into the RAG chain by using a LLM that generates correct answers interpreting the provided question and the relevant similar text extracted from the document.

Evaluation Metrics

To measure the accuracy and performance of the RAG-powered chatbot, I have a list the following evaluation metrics in my mind:

1. **RAGAS Framework**: RAGAS is a specialised framework designed to evaluate Retrieval Augmented Generation (RAG) models. It provides a comprehensive set of metrics tailored for this purpose:

- ***Answer Similarity***: Measures how closely the chatbot's response matches the expected answer.
- ***Faithfulness***: Assesses whether the chatbot's response is factually accurate and directly derived from the retrieved documents.
- ***Answer Correctness***: Evaluates the correctness of the chatbot's answers in a broader context, considering the relevance and accuracy of the information provided.

2. **Hallucination Detection**: Using hallucination and rank_grader prompt templates, I evaluated the extent to which the chatbot produces responses that are not supported by the retrieved documents. Reducing hallucinations is crucial for maintaining the reliability of the chatbot.

3. **Machine Learning Algorithm**: Additional ML-based evaluations were conducted to cross-verify the performance metrics. This included training classifiers to predict the correctness and relevance of the chatbot's responses based on labelled training data.

I have chosen RAGAS for efficient evaluation of RAG!!.

Steps to Improve Accuracy/Performance

To enhance the accuracy and performance of the chatbot, I implemented the following strategies:

1. **Prompt Engineering**: By designing specific prompt templates, I ensured that the language model generates more precise and contextually relevant responses. This involves chaining similar pieces of information and refining the prompts to better guide the RAG..

2. **Iterative Fine-Tuning**: The chatbot model was fine-tuned iteratively on the dataset to improve its ability to understand and respond to the questions accurately. Fine-tuning helps the model to adapt better to the specific context of the external data.
Nowadays many compute-efficient Fine-Tuning techniques are in use that reduce the computational overhead issue by 20x times using LORA andQ-LORA techniques!.

3. **Use of RAG Evaluator and Hallucinator**: By incorporating tools like the RAG evaluator and hallucinator, I continuously monitored and adjusted the model to reduce the incidence of hallucinated answers and improve overall response faithfulness.

4. **Feedback Loop**: By implementing a feedback loop where user interactions with the chatbot are analysed to identify patterns of incorrect or suboptimal responses. This feedback is then used to further refine the model.

Submission

The complete code is available in the GitHub repository. Please refer to the repository for detailed implementation, code snippets, and further explanations of the methodologies used.
[GitHub Repository Link]

Final dataset with all evaluation results link