LLM driven chatbot on ConvFinQA dataset

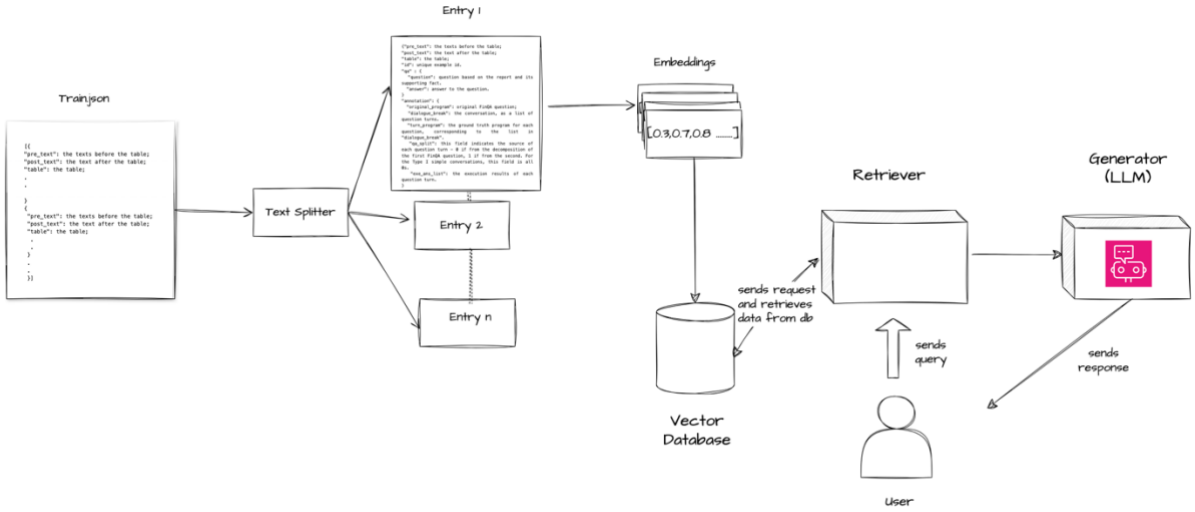
A Python RAG application that can be queried on a set of financial reports using natural language

Data

The data was downloaded from the repository https://github.com/czyssrs/ConvFinQA.

Methodology

I ran the entire project locally on my computer using Ollama (a platform that manages and runs open source LLMs locally on your computer). The prototype is thus built using a subset of train.json file (consisting of about 50 reports) to allow quicker execution without out-of-memory errors.



The train_subset.json file is split into chunks where each chunk has a single entry. This splitting is done using LangChain. As the entries already have ids where each id is a combination of report id and the entry id - I used that field for indexing and storing the data. This provides the ease of adding documents to the same database (without having to rebuild it from scratch) and ensures there are no duplicate entries.

From the chunks, all the key fields are extracted and fed to an embedding function. I have used GPT4AllEmbeddings(). It is very important to use a good embedding model as it drastically affects the quality of response generated from the LLM.

All the embeddings are then stored in a vector database. I have used ChromaDB. If you don't specify a matching list of IDs for the items that you're adding, then Chroma will generate new UUIDs for us automatically. It's convenient, but it also means that we won't be able to check for the existing items.

Then, when a user asks a question, the query is also turned into an embedding. The database

is searched using similarity score that fetches the k most relevant entries from the database. Using those entries together in a prompt we get our final response. For prompt generation, I have used Mistral as it is small in size and has good performance. It is pulled locally using Ollama.

Model Performance

The performance of the embedding model is good as the retriever almost always retrieves the correct entry when queried. However, prompt generation suffers as the dataset has lots of mathematical operations. The existing LLMs out there are not good at performing operations and require fine-tuning on the ConvFinQA dataset.

Unit Testing

To evaluate the quality of responses I have used unit testing with pytest
I have done positive and negative test cases on the following scenarios -

1. **Test Single Hop Questions**:

This test scenario focuses on querying the RAG+LangChain Chatbot with single-hop questions. These are questions that can be answered directly from the provided context without needing to infer or gather additional information. The test asserts that the response generated by the chatbot matches the expected response for the given query.

2. **Test Simple Add Questions:**

In this scenario, the chatbot is tested with questions that involve simple addition. The questions require the chatbot to calculate the sum of numerical values or amounts based on the provided context. The test verifies whether the chatbot correctly computes the sum and provides the expected response.

3. **Test Simple Subtract Questions**:

Similar to the previous scenario, this test involves questions that require the chatbot to perform subtraction. The questions typically ask for the difference between two numerical values or amounts. The test ensures that the chatbot accurately computes the difference and returns the expected response.

4. **Test Simple Multiply Questions**:

This scenario evaluates the chatbot's ability to handle questions involving multiplication. The questions may ask for the product of numerical values or quantities provided in the context. The test validates whether the chatbot correctly performs the multiplication and provides the expected result.

5. **Test Simple Divide Questions:**

Here, the chatbot is tested with questions that involve division. These questions typically ask for the quotient or ratio of numerical values or proportions available in the context. The test checks whether the chatbot accurately divides the numbers and returns the expected quotient or ratio.

6. **Test Multiple Hop Questions:**

In this scenario, the chatbot is presented with questions that require multiple steps or inferences to answer. These questions cannot be answered directly from the given context and may involve combining information or making deductions. The test assesses whether the chatbot is capable of navigating through multiple hops of reasoning to provide the correct response.

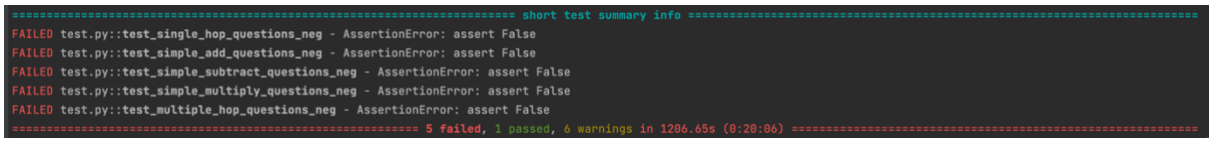
Test Summary

50% of the positive test cases passed and the remaining failed.

Single hop, subtraction and division test cases passed successfully while the LLM gave incorrect responses for addition, multiplication and multihop questions.

Results for positive testing

As pytest shows explanation of the Failed test cases only, to ensure that the LLM has not passed incorrect answers as correct, I have performed negative testing as well.



Results for negative testing

Out of 6, 5 test cases failed which makes the framework about 80% accurate in negative test scenarios.

Setting the threshold for the passing of test cases allow us to get a fair idea of the overall accuracy in the responses of the Chatbot.

Conclusion

The LLM-driven Chatbot is about 50% accurate in generating responses.

It shows good performance in single-hop questions, almost always fetching the right result given that k is set to 1.

Limitations and future work

1. **LLM Performance on Mathematical Operations:**
The current implementation of the Language and Logic Model (LLM) exhibits a 50% failure rate in accurately performing mathematical operations. This could be attributed to inherent limitations in the model's architecture or training data. To address this issue, we can adopt a methodology similar to the FinQANet pipeline proposed in the original ConvFinQA research. By training the model on a curated set of financial operations, we can fine-tune the LLM to generate more precise responses for mathematical queries.

2. **Updating Chroma DB Entries:**
While the current implementation allows for the addition of new data without rebuilding the entire Chroma database, it lacks robustness when it comes to modifying existing entries. For instance, editing a chunk within a page does not update the corresponding chunk ID, leading to inconsistencies in the database. To overcome this limitation, we can integrate approaches that ensure the Chroma DB is updated accurately when modifying existing pages. This could involve implementing mechanisms to re-index affected chunks or maintaining version control for each database entry.

3. **Local Pipeline Execution and Memory Constraints:**
Due to memory limitations, the pipeline currently operates on a small subset of data when running locally. While this presents a constraint in terms of scalability and comprehensive analysis, the pipeline itself is highly adaptable and can be seamlessly deployed on cloud infrastructure using various API integrations. By leveraging cloud resources, we can overcome memory constraints and process larger datasets more efficiently.

4. **Enhanced Response Handling for Multi-Question Queries:**
To improve the usability of the system, the output response from the LLM can be enhanced to support multiple questions on the same database entry. This feature would enable users to ask follow-up questions or inquire about related information within the same context. By facilitating multi-question interactions, the system can provide more comprehensive responses and better address complex queries, particularly those involving multiple hops or layers of inference.

5. **Latency in Response Generation**:

One significant challenge with the current pipeline is the considerable latency in generating responses, typically ranging from 4 to 5 minutes. This prolonged response time can significantly impact user experience and system usability, particularly in real-time or interactive applications where quick responses are essential. Cloud deployment can solve this issue coupled with techniques like parallel and distributed computing.

Links : https://github.com/AnveshaM/convfinqa_chatbot