Aaron Moss

AIE4 Midterm

9/24/24

Background

The CEO and corporate, with permission of the board, have assembled a crack data science and engineering team to take advantage of RAG, agents, and all of the latest open-source technologies emerging in the industry. This time it's for real though. This time, the company is aiming squarely at some Return On Investment - some ROI - on its research and development dollars.

The Problem

You are an AI Solutions Engineer. You've worked directly with internal stakeholders to identify a problem: `people are concerned about the implications of AI, and no one seems to understand the right way to think about building ethical and useful AI applications for enterprises.`

This is a big problem and one that is rapidly changing. Several people you interviewed said that *they could benefit from a chatbot that helped them understand how the AI industry is evolving, especially as it relates to politics.* Many are interested due to the current election cycle, but others feel that some of the best guidance is likely to come from the government.

Task 1: Dealing with the Data

You identify the following important documents that, if used for context, you believe will help people understand what's happening now:

2022: Blueprint for an AI Bill of Rights: Making Automated Systems Work for the American People (PDF)

2024: National Institute of Standards and Technology (NIST) Artificial Intelligent Risk Management Framework (PDF)

Your boss, the SVP of Technology, green-lighted this project to drive the adoption of AI throughout the enterprise. It will be a nice showpiece for the upcoming conference and the big AI initiative announcement the CEO is planning.

✅ Deliverables:

Describe the default chunking strategy that you will use.

After reviewing pdfs, I wanted to maintain the sections or headers of the documents. In order to execute this method, I found the sections via table of contents in each pdf. Then I used functions that found the regex pattern to associate sections to content.

Articulate a chunking strategy that you would also like to test out.

The alternative strategy will be section-based chunks plus semantic chunking.

Describe how and why you made these decisions

In order to maintain the sections or headers of the documents, I found the sections via table of contents in each pdf and converted in-syntax (dict or list). Then I used functions that found the regex pattern (nist_docs) or page_num (wh_paper) to associate sections to content. Due to the low N of source docs, I was able to execute static rules (regex or page_num) that will work for the sample. These approaches will have difficulty being replicated if new samples were provided. In future, using headers to chunk with around sections may be viable.

Task 2: Build an end-to-end RAG application

Build a prototype and deploy to a Hugging Face Space, and include the public URL link to your space create a short (< 2 min) loom video demonstrating some initial testing inputs and outputs.

✅ Deliverables:

Here is my HF created space and Loom video review.

How did you choose your stack, and why did you select each tool the way you did?

When selecting the stack for this project, the focus was on ensuring seamless interaction between the user, model, and tools, while optimizing functionality and scalability. Docker was chosen to containerize the model and app, creating a self-contained environment with all necessary dependencies, simplifying deployment.

Chainlit was used for user interaction, enabling smooth communication between the user's query and the model. Qdrant, the vector database, efficiently stores and retrieves document embeddings generated from chunked text, allowing for quick query processing.

OpenAI's "text-embedding-3-large" model was used to generate embeddings, with plans to fine-tune it using Snowflake Arctic for improved task performance. Hugging Face hosts the model, providing easy access through an open API.

For generating responses, OpenAI's GPT-4 serves as the core, using the user query and context chunks to produce accurate answers. Langchain orchestrates the integration of tools like OpenAI, Qdrant, and others, ensuring a smooth, cohesive workflow.

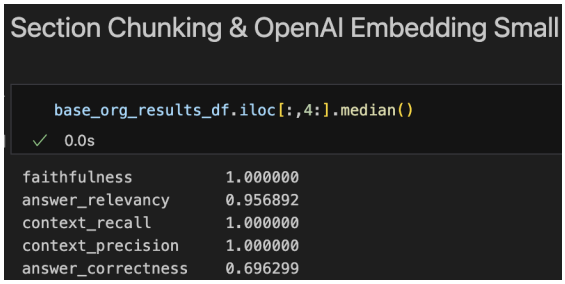
Task 3: Creating a Golden Test Data Set

Generate a synthetic test data set and baseline an initial evaluation

✅ Deliverables:

Assess your pipeline using the RAGAS framework including key metrics faithfulness, answer relevancy, context precision, and context recall. Provide a table of your output results.

Using a testset of 60 observations and LLM-as-judge (OpenAI LLM), on median:



What conclusions can you draw about performance and effectiveness of your pipeline with this information?

While answer correctness (0.70) is lower, it suggests that despite the model retrieving the correct context and staying faithful to the source, there might be some issues in how the final answer is processed or interpreted. Precision and recall in context retrieval are both perfect, which suggests the problem lies not in the retrieval step, but in generating or interpreting the final answer. Increasing complexity or steps in the generation LLM could boost performance.

Task 4: Fine-Tuning Open-Source Embeddings (In Colab)

✅ Deliverables

Swap out your existing embedding model for the new fine-tuned version. Provide a link to your fine-tuned embedding model on the Hugging Face Hub.

Colab Notebook used for Dev & HF Hub link.

How did you choose the embedding model for this application?

I chose the base open-source Snowflake due to its fine-tuned performance at the sentence level and relatively small size (only 0.11B (109M) parameters) that can easily fit on a Google colab instance to utilize free GPU. Next iterations of the embedding model could be selected from Massive Text Embedding Benchmark Leaderboard using a few key parameters (ie., No. parameters, memory usage, and performance relative to domain task).

Task 5: Assessing Performance

Test out new ft embedding model performance on RAG

✅ Deliverables

Base Chunk RAG

```
base_results_df.iloc[:,4:].median()
✓ 0.0s
faithfulness          1.000000
answer_relevancy      0.964068
context_recall        1.000000
context_precision     1.000000
answer_correctness    0.673044
dtype: float64
```

Section + Semnatic Chunk

```
results_df.iloc[:,4:].median()
✓ 0.0s
faithfulness          1.000000
answer_relevancy      0.963330
context_recall        1.000000
context_precision     1.000000
answer_correctness    0.699359
dtype: float64
```

Section & Semnatic Chunk + Ft Artic Emb model

```
results_df_ft.iloc[:,4:].median()
✓ 0.0s
faithfulness          1.000000
answer_relevancy      0.966723
context_recall        1.000000
context_precision     1.000000
answer_correctness    0.747258
```

Using a testset of 60 observations and LLM-as-judge (OpenAI LLM), on median:

- Section-chunking only yielded a perfect score on faithfulness, context recall, and recall precision. Answer relevancy was very high with 96% correct. The answer was judged, by LLM, as being correct 67% of the time.
- Second chunking strategy, section-based chunking with Langchain semantic chunking using OpenAI embeddings, was evaluated as well. Near perfect match with section-based chunking only condition of difference to note was answer correctness was improved to 70% correct by LLM-judge.
- With improved in answer correctness, the second chunking strategy documents retrieved using our Snowflake Fine-tuned model. This RAG pipeline was found to be perfect on context recall, context precision, and faithfulness. It was observed that this pipeline improved over the second strategy alone. Answer correctness was evaluated by LLM-judge as being 75% correct and answer relevancy was 97% correct. This condition (Sec + Semantic Chunking + Ft Snowflake Embed model) was the most performant. As such, this is the best pipeline to test with internal stakeholders next week.
- Noted: Section-based chunking + SF Fine-tuned model was evaluated as well. While it performed better on metrics than chunking strategies alone, it was not evaluated better

on answer relevancy & correctness compared to “Sec + Semantic Chunking + Ft Snowflake Embed” condition

Task 6: Managing Your Boss and User Expectations

You are the SVP of Technology. Given the work done by your team so far, you're now sitting down with the AI Solutions Engineer. You have tasked the solutions engineer to test out the new application with at least 50 different internal stakeholders over the next month.

✅ Deliverables

What is the story that you will give to the CEO to tell the whole company at the launch next month?

Technology is ever-changing, AI even more so, employees find themselves increasingly worried about the implications of artificial intelligence. Questions swirl about how to develop AI applications that are both ethical and beneficial. The lack of clear understanding creates unease, especially as the upcoming election brings AI and politics into sharp focus. Many feel that guidance from the government might offer clarity, but they need a way to access and interpret that information effectively.

To address this concern, the company introduces the AI Gov Bot—a sophisticated chatbot designed to demystify the AI landscape. This tool connects users with up-to-date information on AI developments, ethics, and related political movements. By pulling data from trusted sources, including government publications, it provides accurate answers to pressing questions.

Employees begin using the chatbot to explore topics like ethical AI practices and new regulations. The accessible format encourages them to learn and ask more. As they interact with the tool, they start to adopt a growth mindset, seeing challenges as opportunities to expand their knowledge. This shift not only alleviates their concerns but also fosters a culture of continuous learning and adaptability within the company.

There appears to be important information not included in our build, for instance, the 270-day update on the 2023 executive order on Safe, Secure, and Trustworthy AI. How might you incorporate relevant white-house briefing information into future versions?

There are two main perspectives and following approaches that need to be assessed:

From the user perspective, the most valued approach would be to allow for uploading new sources that would be to apply chunker into documents that can be added to the retrieval process and processed by the embedding model into the vector store. It would give the user the flexibility and freedom to dynamically add more info to in-context learning for the generating LLM to consider.

From the developer perspective, the simplest approach would be to retain control of sources and process uniquely based on the quality & quantity of the source so that the user doesn't have to deal with the complexity of uploading helpful sources and less opportunity for backend breakage. For the demo scenario and low N of sources given, I would favor this approach so that the demo experience is as frictionless as possible.