Project Report:

Semantic Spotter Project

Kiran S. Dalmiya

September 24, 2024

1. Project Objectives

The main goal of this project was to build a smart system that can search through documents and give clear, helpful answers to questions. It combines two powerful tools:

- **LlamaIndex** (to search and find relevant information in documents)
- **GPT-3.5-turbo** (to improve the answers and make them more user-friendly).

This system was designed to answer questions specifically related to insurance policy documents.

Framework Used: LlamaIndex has been used due to its powerful query engine, fast data processing using data loaders and directory readers as well as easier and faster implementation using fewer lines of code.

Data Source: The data source for this project consists of insurance policy documents in PDF format, loaded from Google Drive using LlamaIndex's SimpleDirectoryReader.

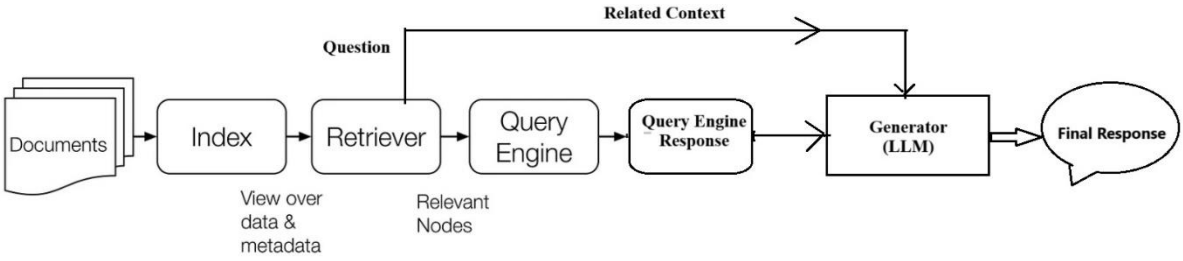
2. Design of the Project

The system has different layers, each performing a specific task:

1. **Document Parsing Layer**: This layer reads documents, breaks them into smaller sections, and adds extra details like the file name and page number for reference.
2. **Indexing Layer**: It creates a searchable "index" of the document. This helps find relevant parts of the document when a question is asked.
3. **Query Engine Layer**: This is where user questions are matched to the most relevant information from the indexed documents.
4. **User Interaction and Feedback Layer**: This allows users to ask questions, receive answers, and give feedback on the response (like saying whether the answer was "good" or "bad").
5. **Generative Enhancement Layer**: Once the initial search result is retrieved, it is enhanced using GPT-3.5-turbo, which provides a more detailed and understandable response.

   Also the Response are Evaluated for Faithfulness relevancy and completeness.

Flowchart for System Design:



Retrieval Augmented Generation using Llamaindex and GPT 3.5 model

3. How the System Works

1. Document Parsing

- The system reads documents stored in Google Drive using the SimpleDirectoryReader tool from LlamaIndex.
- The documents are broken down into smaller sections (called "nodes"). Each section has meaningful content that can be searched later.

2. Indexing

- The system then converts these document sections into searchable "vectors" (a kind of digital representation).
- This index allows the system to quickly search for and retrieve the most relevant parts of the document when a question is asked.

3. Query Engine

- When a user asks a question, the system searches the index for the most relevant sections of the document.
- It then provides the answer, along with details about where that information was found (e.g., file name and page number).

4. Generative Enhancement

- The response from the query engine is passed to GPT-3.5-turbo, which improves the answer, making it more complete, easier to understand, and adding extra useful information if necessary.

5. Response Evaluation

- The final response from the query engine is evaluated for faithfulness,relevancy and completeness.

6. User Feedback

- Users can give feedback on whether the response was good or bad. This feedback helps improve the system over time.

4. Implementation

Here's how the system was built step-by-step:

1. **Document Reading**:
    - We used SimpleDirectoryReader to load insurance policy documents from Google Drive.
2. **Indexing**:
    - The system created an index of the documents by breaking them into smaller parts and converting them into vectors for fast searching.
3. **Searching for Answers**:
    - When a user asks a question, the system searches the index to find the best match and returns the relevant sections of the document using llamaindex framework's query engine.
4. **Enhancing the Response**:
    - The raw answer from the query search is passed to GPT-3.5-turbo, which refines it and provides a clearer, more detailed response.
5. **Response Evaluation:**
    - The final response from the query engine is assessed for **faithfulness**, **relevancy**, and **completeness**. To achieve this, the function evaluate_responses() is used, which leverages specific evaluators for each aspect
6. **Feedback Collection**:
    - Users can rate the responses (good or bad), and we store this feedback for future improvements.

5. Challenges

1. **Speed of Response**:
    - Searching through large documents and generating responses with GPT-3.5-turbo can take some time, especially with large documents. We had to balance response quality with speed.
2. **Improving the Generated Answers**:
    - It took several tries to get the prompts for GPT-3.5-turbo right so that it gives the best possible response without being too long or too vague.
3. **Evaluating the Accuracy**:

- Making sure the answers were accurate and relevant, especially for complex questions, was a challenge. We have user different evaluation matric like relevance and faithfulness.
- We still had to rely on user feedback for future improve.

6. Lessons Learned

1. **Vector Indexing Improves Search**:
    - Using vector-based indexing made the system much more efficient at finding the most relevant sections of documents.
2. **Prompting GPT-3.5 Effectively**:
    - Crafting the right prompts for GPT-3.5 was key to getting clear, helpful answers. This required a lot of testing and adjusting based on how the model responded.
3. **User Feedback Helps Improve Accuracy**:
    - Feedback from users is crucial for understanding how well the system is performing and where it needs improvement.

7. Future Improvements

1. **Speed Optimization**:
    - We can improve the system's speed, especially for large documents, by optimizing how we handle the indexing and search process.
2. **Better Feedback Analysis**:
    - Analyzing user feedback more effectively can help improve the quality of responses and make the system smarter over time.
3. **Support for More Document Types**:
    - We could expand the system to handle a wider variety of document formats, like scanned images or older documents.
4. **Improving Citation Accuracy:**
    - Citation can be made more accurate by mentioning specific sections or paragraphs along with the page number from the policy.
    - Can Include a clickable link that takes users directly to the exact part of the document, making it easier for them to find the relevant information quickly.
5. **User-Friendly Visual Interface:**

- Create a simple visual interface, like a website or app, to make the project easy to use.

8. Conclusion

This project built an efficient query engine that quickly finds relevant information from documents and combines it with the conversational power of GPT-3.5-turbo with Llamaindex framework. It not only retrieves useful information but also improves the responses and adapts based on user feedback. The system can be further developed and applied to other areas like law, education, or finance, making it versatile beyond just insurance.