Help-Mate AI Project

1. Background

This project focuses on developing an effective search system**, HelpMateAI**, designed to process and retrieve information from a lengthy insurance policy document using **Retrieval-Augmented Generation (RAG) techniques**.

2. Problem Statement

The objective is to build a **reliable generative search system** that can accurately and efficiently answer questions based on the content of an insurance policy document. For this project, we will be working with a **single long life insurance policy document**.

3. Document

The reference document used is **Principal-Sample-Life-Insurance-Policy.pdf**

4. Approach

The project should implement all the three layers effectively. It will be key to try out various strategies and experiments in various layers in order to build an effective search system. Let's explore what we need to do in each of the layers.

1) **The Embedding Layer:** The PDF document needs to be effectively processed, cleaned, and chunked for the embeddings. Here, the choice of the chunking strategy will have a large impact on the final quality of the retrieved results. So, we need to make sure that we try out various strategies and compare their performances.

   Another important aspect in the embedding layer is the choice of the embedding model. we can choose to embed the chunks using the OpenAI embed- ding model or any model from the SentenceTransformers library on HuggingFace.

2) **The Search Layer:** Here, we need to design at least 3 queries against which you will test your system. You need to understand and skim through the document, and accordingly come up with some queries, the answers to which can be found in the policy document.

   Next, we need to embed the queries and search your ChromaDB vector database against each of these queries. Implementing a cache mechanism is also mandatory.

   Finally, we need to implement the re-ranking block, and for this we can choose from a range of cross-encoding models on HuggingFace.

3) **The Generation Layer:** In the generation layer, the final prompt that we design is the major component. We need to make sure that the prompt is exhaustive in its instructions, and the relevant information is correctly passed to the prompt. We may also choose to provide some few-shot examples in an attempt to improve the LLM output.

5. System Layers

- **Reading & Processing PDF File:** We will be using pdfplumber to read and process the PDF files. pdfplumber allows better parsing of the PDF file as it can read various elements of the PDF apart from the plain text, such as, tables, images, etc. It also offers wide functionalities and visual debugging features to help with advanced preprocessing as well.

- **Document Chunking:** The document contains several pages and contains huge text, before generating the embeddings, we need to generate the chunks. Let's start with a basic chunking technique, and chunking the text with fixed size.

- **Generating Embeddings:** Generates embedding with SentenceTransformer with all-MiniLM-L6-v2 model.

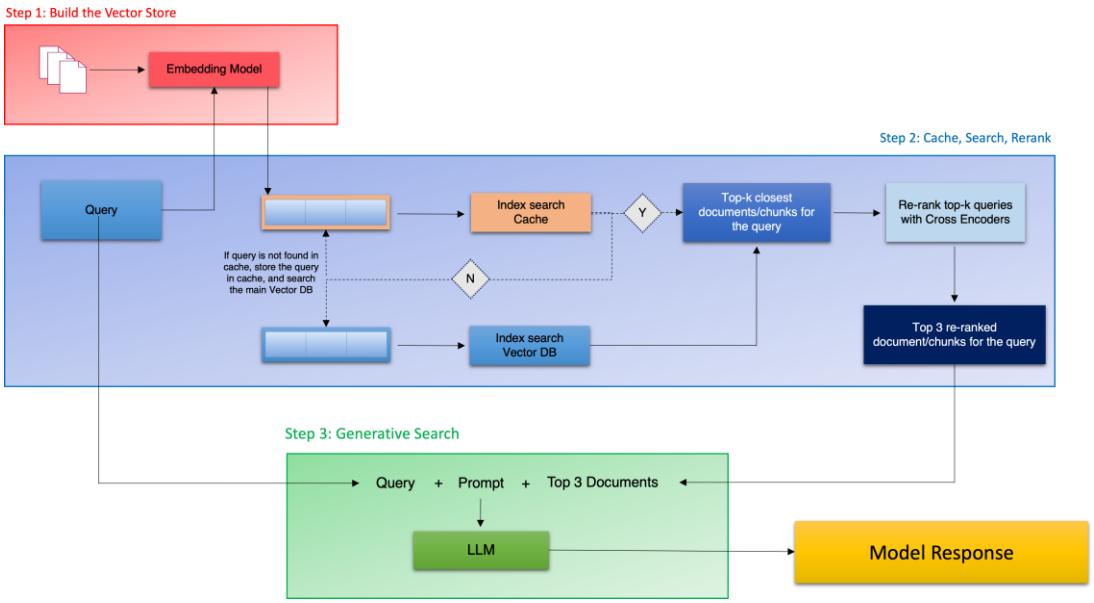
- **Store Embeddings In ChromaDB:** In this section we will store embedding in Chroma DB.

- **Semantic Search with Cache:** In this section we will introduce cache col- lection layer for embeddings.

- **Re-Ranking with a Cross Encoder:** Re-ranking the results obtained from the semantic search will sometime significantly improve the relevance of the retrieved results. This is often done by passing the query paired with each of the retrieved responses into a cross-encoder to score the relevance of the response with respect to the query.

- **Retrieval Augmented Generation:** Now we have the final top search results, we can pass it to an GPT 3.5 along with the user query and a well-engineered prompt, to generate a direct answer to the query along with citations.

6. System Architecture



7. Prerequisites

- Python 3.12
- Please ensure that you add your OpenAI API key to the empty text file named “My_API_Key” in order to access the OpenAI API.

8. Query Screenshots

```
query = 'what is the life insurance coverage for disability'
df = search(query)
df = apply_cross_encoder(query, df)
df = get_topn(3, df)
response = generate_response(query, df)
print("\n".join(response))
```

```
The life insurance coverage for disability provided in the insurance policies is subject to the benefits outlined in the policy document. The spec
fic details regarding coverage for disability, including any applicable terms, conditions, and benefits, can be found in the policy document secti
ns related to Member Life Insurance or Coverage During Disability.

For more detailed information on the disability coverage under life insurance, please refer to the following policy documents and their respective
pages:
1. Policy Name: Member Life Insurance or Coverage During Disability
   Page Number: Page 42

Feel free to review the mentioned policy document for comprehensive details on the life insurance coverage for disability.
```

```
query = 'what is the Proof of ADL Disability or Total Disability'
df = search(query)
df = apply_cross_encoder(query, df)
df = get_topn(3, df)
response = generate_response(query, df)
print("\n".join(response))
```

```
The Proof of ADL Disability or Total Disability refers to the documentation required to demonstrate that an individual meets the criteria for Activities
of Daily Living (ADL) Disability or Total Disability as defined by the insurance policy. This proof typically includes medical records, assessments from
healthcare professionals, and other relevant documentation that substantiates the claimant's inability to perform daily tasks or work due to a disabilit
y.

Citation:
- Policy Name: Member Life Insurance or Coverage During Disability
  - Page Number: Page 42

- Policy Name: Dependent's Life Insurance
  - Page Number: Page 44

In the provided documents, specific details or criteria regarding the Proof of ADL Disability or Total Disability may be outlined. It is recommended to
refer to the mentioned policy names and respective page numbers for detailed information on what constitutes acceptable proof in the context of these in
surance policies.
```

```
query = 'what is condition of deatht while not wearing Seat Belt'
df = search(query)
df = apply_cross_encoder(query, df)
df = get_topn(3, df)
response = generate_response(query, df)
print("\n".join(response))
```

```
The query about the condition of death while not wearing a seatbelt is not directly relevant to the provided insurance documents. I recommend checking t
he specific sections related to exclusions, coverage limitations, and conditions in your insurance policy documents to find detailed information on the
subject.

If you need further assistance or more specific details, I suggest reviewing the sections related to fatalities, accidents, or exclusions within the ins
urance policies listed below:

1. Policy Name: Member Life Insurance or Coverage During Disability
   Page Number: Page 42

2. Policy Name: Dependent's Life Insurance Terminations
   Page Number: Page 44

3. Policy Name: Payment of Benefits
   Page Number: Page 49

Please refer to the cited policies for comprehensive information related to your query.
```