**Project :HelpMate AI**

**1.** **Objectives**:

The primary objectives of the project are as follows:

* Develop a semantic search system using the RAG (Embedding Layer, Search and Rank Layer, Generation Layer) pipeline for efficient document retrieval.
* Extract relevant information from PDF documents, store them in a structured format, and generate vector representations using embedding the pages in the dataframe through OpenAI's `text-embedding-ada-002` model, and store them in a ChromaDB collection .
* Implement a cache layer to enhance system performance by storing and retrieving previous queries and their results
* build a robust generative search system capable of effectively and accurately answering questions from a policy document.

**Data:** A single long life insurance policy document in PDF format for this project.

**2. Design:**

2.1. RAG Pipeline:

Embedding Layer: Extract text and tables from PDFs, convert them to a dataframe, using overlapping chunking strategy to chunk the document and generate vector representations using OpenAI's text-embedding-ada-002 model. Store these embeddings in ChromaDB.

Search and Rank Layer: Perform a semantic similarity search on the knowledge bank based on user queries, retrieving the top 3 closest documents or chunks.

Generation Layer: Utilize the results from the previous layer, including the original user query and a well-constructed prompt, to generate coherent answers using a language model.

2.2. Cache Implementation:

Set a threshold of 0.2 for semantic similarity.

Store queries and results in a cache\_collection in ChromaDB for easy embedding and searching.

Use ChromaDB's utility functions to add documents, ids, and metadata to the cache\_collection.

**3. Implementation:**

Use Jupiter notebook for development and leverage libraries such as pdfplumber, openai, chromaDB, for document processing, embedding, and caching.

Implement functions to extract text and tables from PDFs, chunk the text using overlap chunking, create a dataframe, generate vector embeddings, and perform semantic searches using the RAG pipeline.

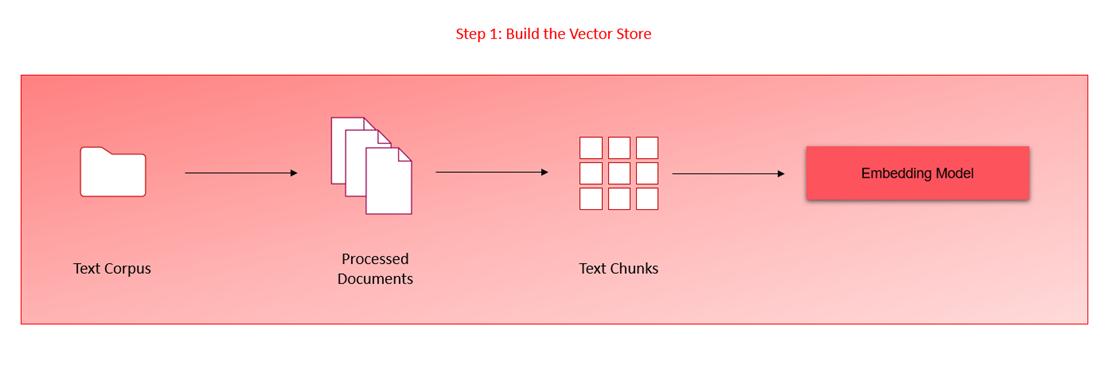
Develop a cache system using ChromaDB to store and retrieve previous queries and their results. Re ranking using cross encoder. In the generation layer, the final prompt is design, the relevant information is correctly passed to the prompt, some few-shot examples are given to improve the LLM output.

The three layers for RAG pipeline are:

1. Embedding Layer
2. Search Layer
3. Generation Layer

Embedding Layer:

Processing and Chunking: Explore and compare various strategies for effective PDF document processing, cleaning, and chunking. Evaluate the impact of different chunking strategies on the quality of the retrieved results.

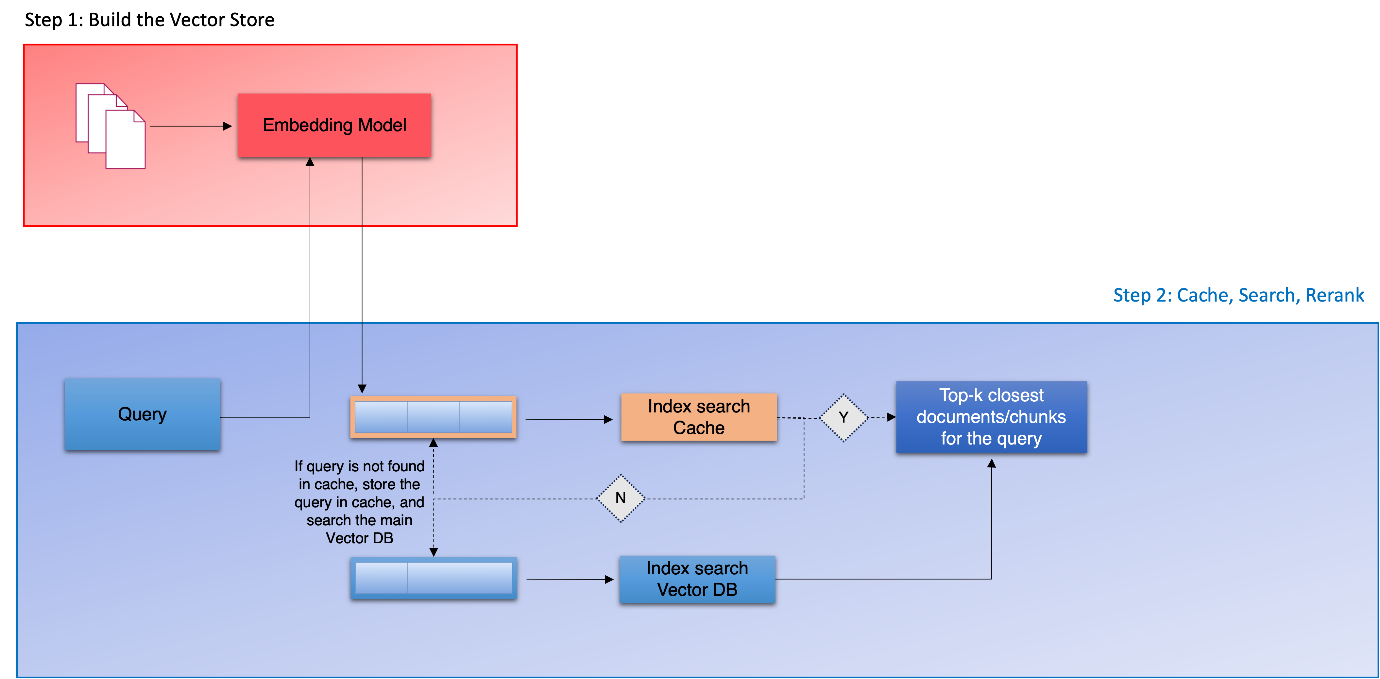


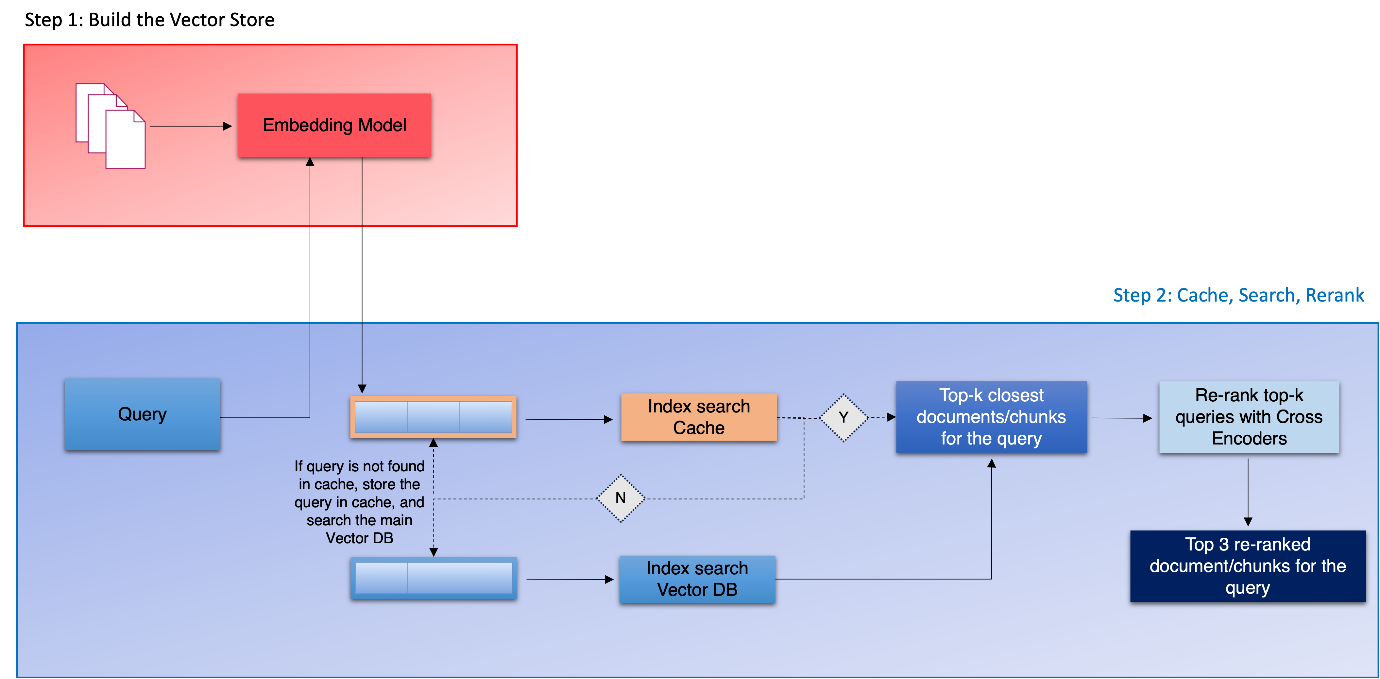
Search Layer:

Query Design: Design a minimum of three queries that reflect information seekers' potential questions in the policy document. Ensure queries cover diverse aspects of the document to thoroughly test the system.

Vector Database Search: Embed queries and perform searches against the ChromaDB vector database. Implement a cache mechanism to store and retrieve previous queries and their results.

Re-ranking Block: Integrate a re-ranking block utilizing cross-encoding models from HuggingFace to enhance the relevance and accuracy of search results.



When using cross\_encoder:  


Generation Layer:

Prompt Design: Focus on designing a comprehensive and instructive prompt for the Language Model (LM) in the generation layer. Ensure the prompt effectively conveys relevant information to the LM for coherent answer generation.

Few-shot Examples: Enhance LM performance by providing few-shot examples in the prompt to guide the model in generating more contextually accurate responses.

**4.Challenges Faced:**

1. Data Quality and Preprocessing

Extracting relevant information from insurance documents, especially from complex text structures, is challenging.

2. Effective Chunking Strategies

Chunk Overlap and Size: Determining the optimal chunk size and overlap to capture meaningful context without losing coherence is challenging.

3. Query Understanding and Matching

Designing queries which are relevant to the document and that require sophisticated understanding and reasoning

**5.Lessons Learned:**

Efficient Document Processing: Processing PDFs efficiently is crucial; libraries like pdfplumber play a vital role.

Semantic Search Optimization: Fine-tune semantic search parameters and thresholds for optimal results.

Cache Management: Implement an effective cache management strategy to balance storage and retrieval efficiency.

**6.Conclusion:**

The project successfully implements a semantic search system with the RAG pipeline and cache layer and Generative search. The objectives are met, and the challenges are overcome with lessons learned for future improvements. The system provides a scalable and efficient solution for document retrieval and information extraction.

Project Report on HelpMate AI by Madhusmita Ghosh