### **Project Report: Mr HelpMate**

### **1. Objectives:**

* **Develop an Advanced Semantic Search System:** Implement a semantic search solution using the RAG pipeline to enhance document retrieval efficiency and accuracy.
* **Integrate Cross-Encoder Re-Ranking:** Employ cross-encoder models to re-rank the retrieved documents based on their relevance to the user query.
* **Generate Concise and Relevant Responses:** Utilize GPT-3.5 to produce precise responses derived from the top-ranked documents.
* **Cache Layer Implementation:**A cache layer is implemented to boost the performance of the semantic search system by storing and efficiently retrieving previous queries and their corresponding results. This reduces redundant computations, accelerates query processing times, and optimizes overall system responsiveness. The cache layer is built using ChromaDB, a vector database that facilitates embedding storage and retrieval operations.

#### 

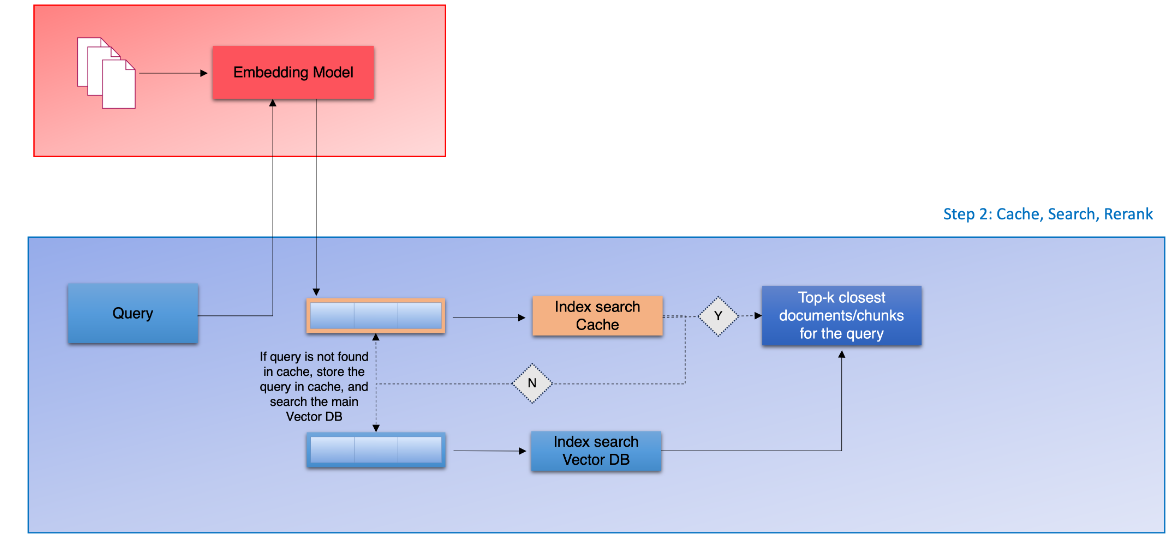
#### **2. Design:**

2.1 **RAG Pipeline:**

* **Embedding Layer:**
  + **Document Processing:** Extract text and tables from PDF documents, convert them into structured dataframes, and create vector embeddings using the SentenceTransformers’ all-MiniLM-L6-v2 model.
  + **Vector Storage:** Store these embeddings in ChromaDB to facilitate efficient semantic searches.
* **Search and Rank Layer:**
  + **Semantic Search:** Conduct semantic searches to find the top K closest documents or chunks based on user queries.
* **Generation Layer:**
  + **Response Generation:** Generate coherent and concise responses using GPT-3.5, based on the results from the previous layers.

2.2 **Cross-Encoder Re-Ranking:**

* **Re-Ranking Process:**
  + **Input Preparation:** Formulate query-document pairs from the top results of the semantic search.
  + **Cross-Encoder Scoring:** Use a cross-encoder model to assign relevance scores to these pairs and re-rank the documents.
* **Top Results Extraction:**
  + **Reranked Results:** Sort documents based on re-ranking scores to determine the most relevant results for final response generation.



3. **Implementation**:

Use Google Colab for development and leverage libraries such as pdfplumber, tiktoken, openai, chromaDB, and sentence-transformers for document processing, embedding, and caching.

Implement functions to extract text and tables from PDFs, 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.

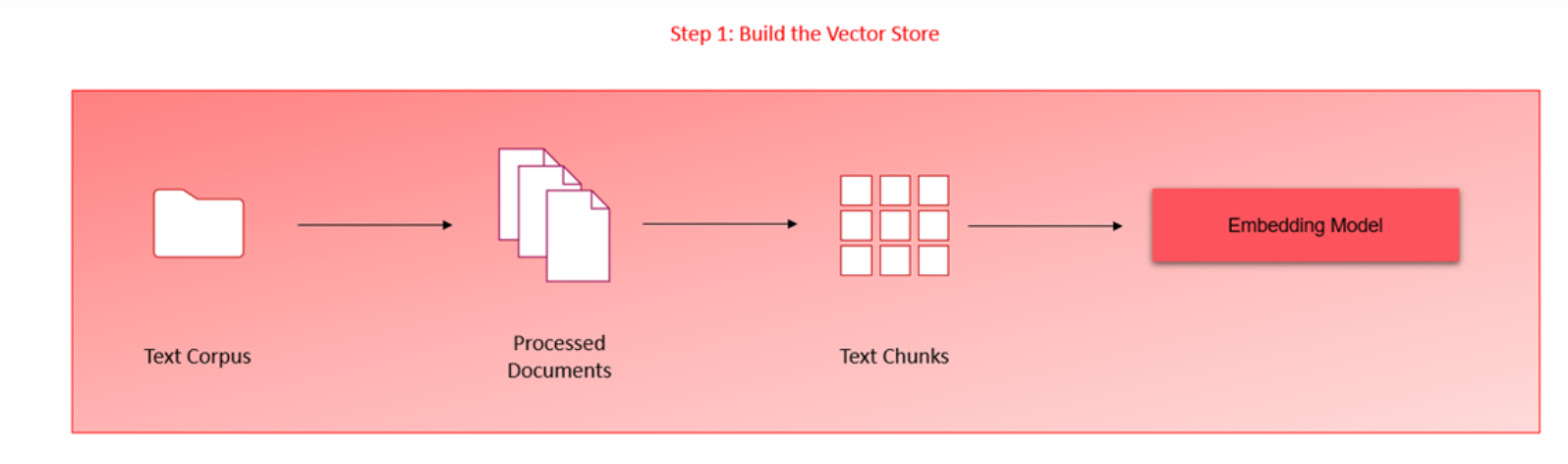
The three layers for RAG pipeline are:

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.

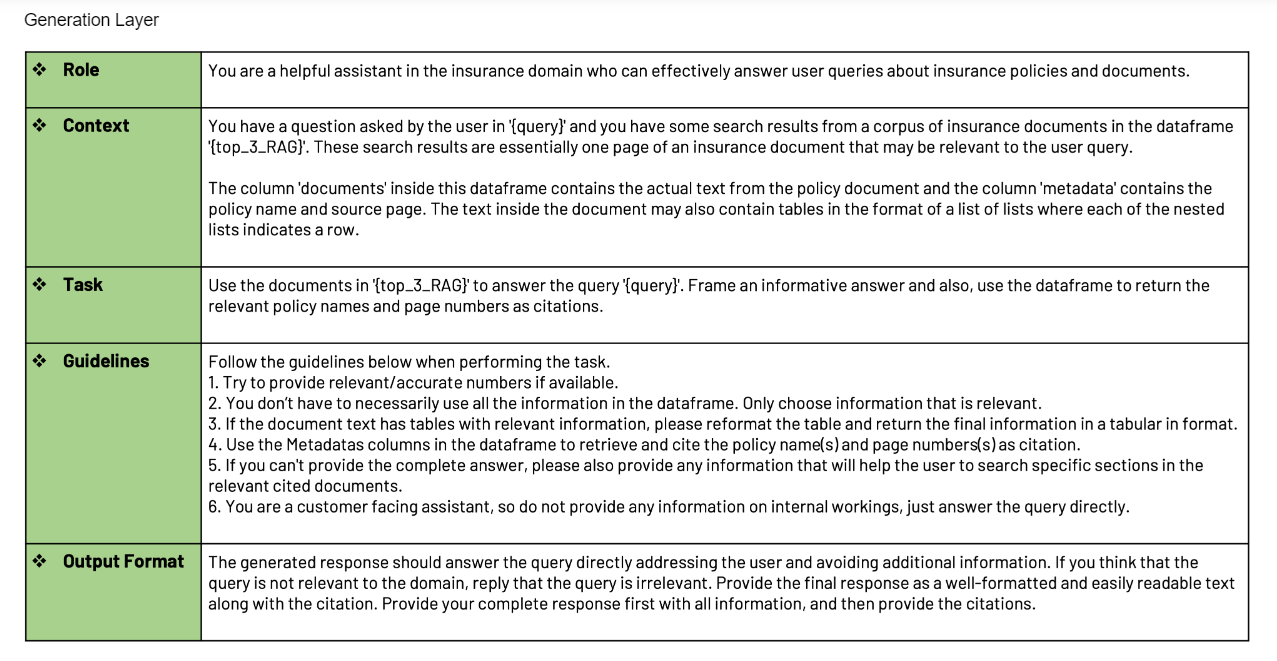
Embedding Model Choice: Choose between OpenAI's embedding model and SentenceTransformers from HuggingFace. Assess the impact of the selected model on the quality of vector representations.



### **Search Layer:**

#### **Query Design:**Develop at least three distinct queries that represent common questions users might have about the policy document. These queries should address various sections or topics within the document to test the system’s breadth and accuracy.

### **Generation Layer:**



1. **Prompt Design:**

* + **Create Effective Prompts:** Develop clear and comprehensive prompts for the language model to guide it in generating accurate and relevant answers. The prompts should effectively convey the query and the context provided by the retrieved documents.

1. **Few-shot Examples:**
   * **Incorporate Examples:** Include a few-shot learning approach in the prompts by providing examples of similar queries and their ideal responses. This helps the language model understand the expected format and improve the quality of generated answers.
2. **Generate Responses:**
   * **Generate and Refine:** Use the language model to generate responses based on the combined context of the user’s query and the relevant documents. Ensure that the responses are concise, accurate, and directly address the query.

#### 

#### **4. Challenges:**

* **Cross-Encoder Integration:** Ensuring the cross-encoder model effectively re-ranks documents to improve relevance and accuracy.
* **Response Conciseness:** Crafting prompts for GPT-3.5 that elicit concise and relevant responses based on varied document contexts.
* **System Performance:** Maintaining efficiency and speed of the system, particularly when handling large volumes of documents and user queries.

#### **5. Lessons Learned:**

* **Effective Re-Ranking:** The cross-encoder model substantially enhances the relevance of search results.
* **Prompt Design:** Well-structured prompts for GPT-3.5 are critical in obtaining useful and succinct responses.
* **System Optimization:** Optimal document processing and re-ranking strategies are essential for balancing system performance and accuracy.

#### **6. Conclusion:**

The project successfully implemented a semantic search system using the RAG pipeline and a cache layer to boost performance. By processing PDF documents into vector embeddings, performing semantic searches, and generating coherent responses, the system effectively retrieves relevant information. The cache layer enhances efficiency by storing and quickly accessing previous queries and results, reducing processing time. Re-ranking with cross-encoding models improves search result relevance.