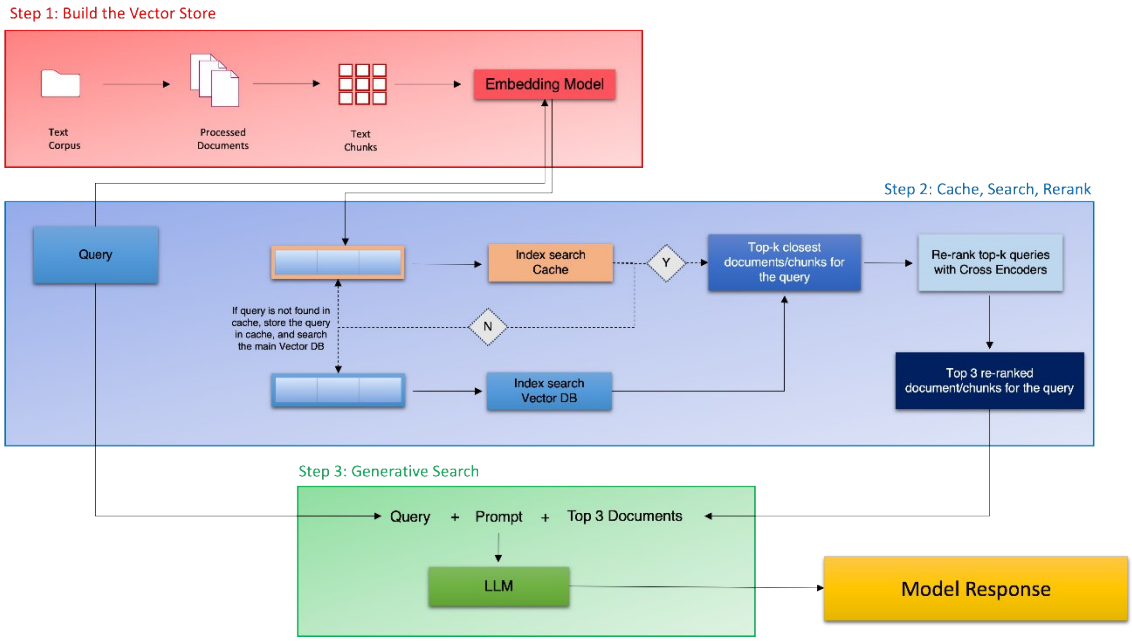
HelpMate AI

Th goal of the project will be to build a robust generative search system, HelpMate AI capable of effectively and accurately answering questions from a policy document.

The Project implementation consists for Three layers.



1. **Embedding Layer:** The PDF document is effectively processed, cleaned, and chunked for the embeddings.
2. **Search Layer:** In this layer 3 queries are used to test the system. Queries are embedded and searched against ChromaDB vector database. Cache mechanism is implemented against each of these queries.
3. **Generation Layer:** In this layer few-shop prompt is designed to accept the query response, process and generate summary response.

Implementation Steps:

1. Read, Process, and Chunk the PDF Files

PDFPlumber is used to read and process the PDF files. PDFPlumber allows for 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.

2. Generate and Store Embeddings using OpenAI and ChromaDB

In this section, Pages are ebmed in the dataframe through OpenAI's `text-embedding-ada-002` model, and store them in a ChromaDB collection.

3. Semantic Search with Cache

Performed semantic search of a query in the collections embeddings to get several top semantically similar results.

4. Re-Ranking with a Cross Encoder

Re-ranking the results obtained from your semantic search can 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 w.r.t. the query.

5. Retrieval Augmented Generation

Now that 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, rather than returning whole pages/chunks.