Help Mate AI Project PDF

Background

This project demonstrate “Building Effective Search Systems HelpMateAI” with a long insurance policy document using RAG techniques.

Problem Statement

The goal of the project is to build a robust generative search system capable of effectively and accurately answering questions from a policy document. We will be using a single long life insurance policy document for this project.

Document

Principal-Sample-Life-Insurance-Policy.pdf

Objectives

- To develop an AI-powered search system that retrieves contextually relevant information.
- To integrate semantic search and re-ranking techniques for better search results.
- To provide an efficient and user-friendly document search experience.

Approach

- Extract and preprocess text from PDF documents.
- Generate embeddings for document chunks using transformer-based models.
- Perform semantic search with caching for efficiency.
- Re-rank search results to improve relevance.
- Generate responses based on retrieved information.

System Layers

Reading & Processing PDF File: We will be using pdfplumber to read and process the PDF files. Pdf plumber 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.

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 Sentence Transformer with all-MiniLM-L6-v2 model.

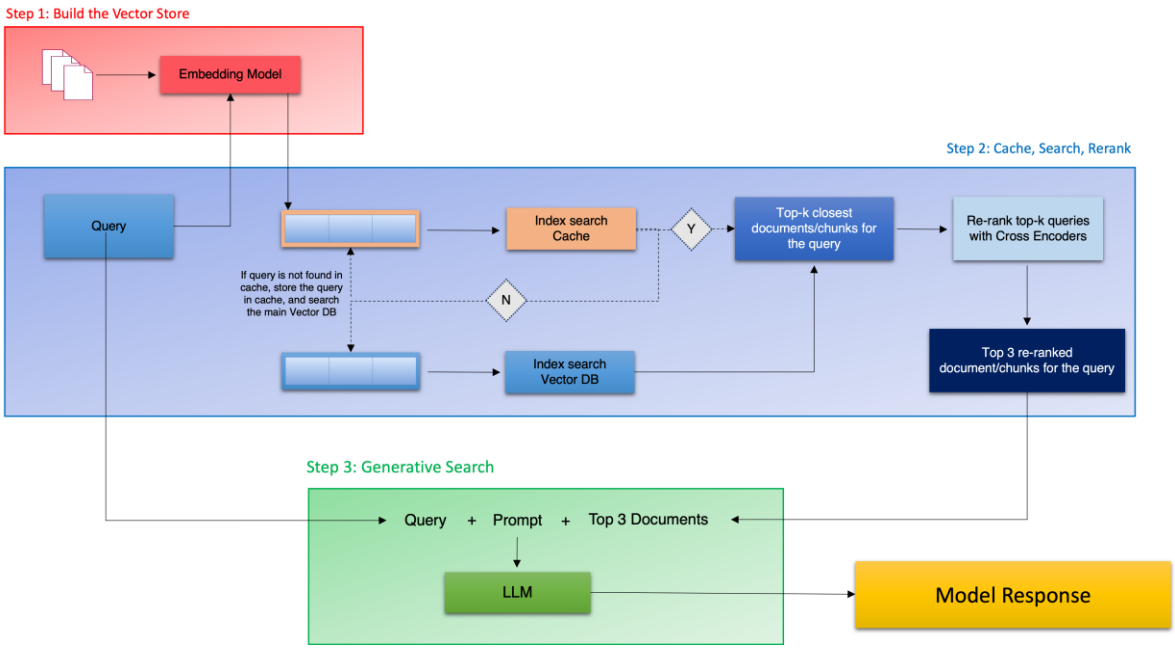
Store Embeddings In ChromaDB: In this section we will store embedding in ChromaDB.

Semantic Search with Cache: In this section we will introduce cache collection 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 w.r.t. 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.

System Architecture

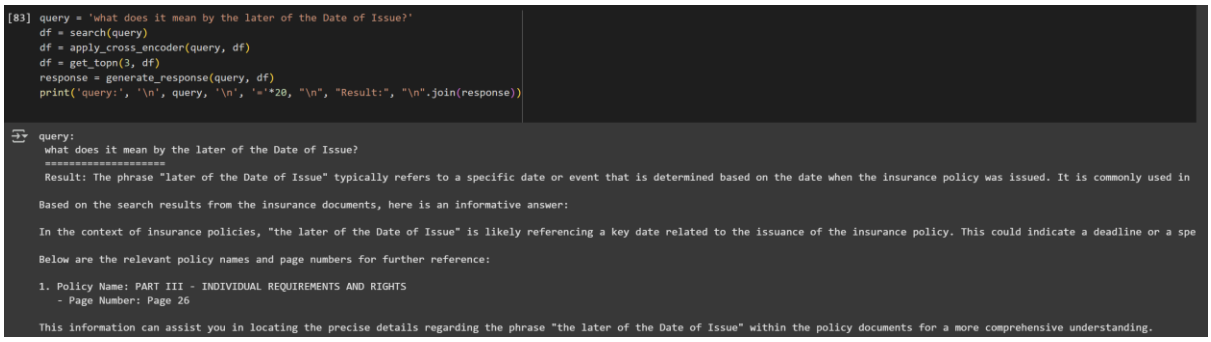


Results & Demonstration

Query1:

“What are the default benefits and provisions of the Group Policy?

Query 2:



Query 3:

Challenges:

- Performance Scaling: Address concerns about system performance with an increased number of
- documents or users by implementing vector databases and scaling up compute units.
  Cache Storage: Optimize the cache collection to store and retrieve queries and results efficiently.

Lessons Learned:

- Efficient Document Processing: Processing PDFs efficiently is crucial; libraries like “pdf plumber” and suitable data structures for storage play a vital role.
- Semantic Search Optimization: Fine-tune semantic search parameters and thresholds for optimal results.
- Cache Management: Implement an effective strategy to balance storage and retrieval efficiency.

Acknowledgements:

- The project references course materials from upGrad's curriculum.
- The project references presentations in upGrad’s recorded module given by Aditya
- Bhattacharya.
- The project references presentations in upGrad’s recorded module given by Akshay Ginodia.
- The project references insights and inferences from presentations in upGrad’s doubt clear
- session given by Shridhar Galande.
- The project references presentations in upGrad's live class given by Sheshanth AS.