SEMANTIC SPOTTER PROJECT SUBMISSION- C61 SUNDARALINGAM

1. Background

This project showcases the development of a Retrieval-Augmented Generation (RAG) system for the insurance domain using LangChain.

2. Problem Statement

The objective is to build a robust generative search system that efficiently and accurately answers queries using policy documents.

3. Document

Policy documents can be accessed in Git Link

4. Approach

LangChain simplifies the development of Large Language Model (LLM) applications by providing a comprehensive framework with modular components and integrations. It supports various LLM providers such as OpenAI, Cohere, and Hugging Face.

LangChain enables seamless integration with diverse data sources, allowing developers to create advanced language model-powered applications. The open-source framework supports both Python and JavaScript/TypeScript, with a core focus on modularity and composition.

5. System Layers

Reading & Processing PDF Files

We use PyPDFDirectoryLoader from LangChain to read and process PDF files from a specified directory.

Document Chunking

The RecursiveCharacterTextSplitter is used to split text efficiently while preserving semantic coherence. It prioritizes paragraph, sentence, and word-level chunking to maintain contextual relevance.

Generating Embeddings

Embeddings are created using OpenAIEmbeddings from LangChain, which supports multiple embedding providers such as OpenAI and Cohere, along with sentence transformers from Hugging Face. These embeddings facilitate similarity search, text comparison, and sentiment analysis.

Storing Embeddings in ChromaDB

Generated embeddings are stored in ChromaDB, backed by LangChain's CacheBackedEmbeddings.

Retrievers

Retrievers enhance query-document matching by fetching relevant documents. They provide an interface for retrieving documents based on unstructured queries. The most commonly used retriever is VectorStoreRetriever.

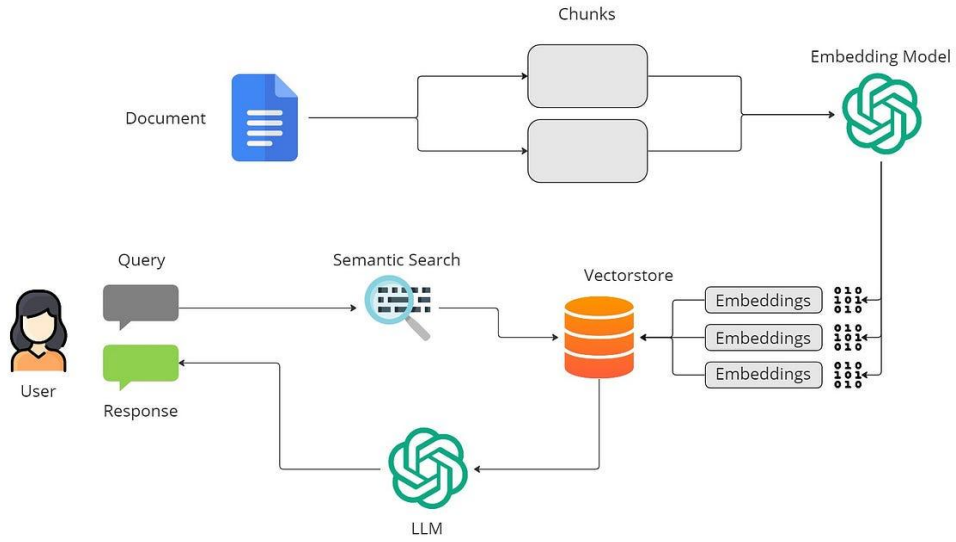
Re-Ranking with a Cross Encoder

To improve search relevance, we apply a cross-encoder (BAAI/bge-reranker-base from Hugging Face) to re-rank retrieved results based on query-response relevance.

Chains

LangChain's Chains enable combining multiple components into a structured workflow. For instance, we use the rlm/rag-promp from LangChain Hub to format user input before passing it to an LLM in the RAG chain.

6. System Architecture



7. Prerequisites

- Python 3.7+
- LangChain 0.3.13
- OpenAI API key

8. Running the Project

1. Clone the GitHub repository.
2. Open the Jupyter notebook.
3. Run all cells to execute the project