Semantic Spotter Project Submission

1. Background

This project demonstrates the implementation of a Retrieval-Augmented Generation (RAG) system tailored for the insurance domain, using the LangChain framework.

2. Problem Statement

The objective is to develop a reliable generative search engine capable of accurately answering user queries using insights from a collection of insurance policy documents.

3. Document Source

The insurance policy documents utilized in this project can be found in the designated `Policy_Documents` directory.

4. Approach

LangChain is an open-source framework designed to simplify the creation of LLM-powered applications. It offers a modular architecture for building complex systems quickly, supporting providers like OpenAI, Cohere, and Hugging Face.

Modular Components:

- **LLM Interfaces**
- **Chains**
- **Retrievers**
- **Agents**
- **Memory**
- **Callbacks**

It provides both low-level building blocks and pre-built chains to support rapid and robust development.

5. System Layers

5.1 PDF Parsing

Loads and processes PDFs via `PyPDFDirectoryLoader`.

5.2 Document Chunking

Uses `RecursiveCharacterTextSplitter` to segment text into semantically meaningful chunks.

5.3 Generating Embeddings

Embeddings are generated through `OpenAIEmbeddings`, enabling semantic search and comparisons.

5.4 Storing Embeddings

Stored in ChromaDB with `CacheBackedEmbeddings` for efficient retrieval.

5.5 Retrieval

Leverages `VectorStoreRetriever` to fetch relevant document chunks for user queries.

5.6 Re-Ranking

Improves relevance by re-scoring results with the `HuggingFaceCrossEncoder` model (`BAAI/bge-reranker-base`).

5.7 RAG Chain Workflow

Combines all components using a `PromptTemplate` and the `rlm/rag-promp` template from LangChain Hub to complete the RAG pipeline.

6. System Architecture

Architecture visuals can be found in the `images/` folder (`arch1.png`, `arch2.png`).

7. Prerequisites

- Python 3.7+
- langchain == 0.3.13
- OpenAI API key saved as `OpenAI_API_Key.txt`

8. Running the Project

1. Clone the repo:
   `git clone https://github.com/vikrampawar88/semantic-spotter-langchain-project.git`
2. Open `semantic-spotter-langchain-project.ipynb` in Jupyter.
3. Run all cells to build and test the pipeline.