Semantic Spotter Project Submission

Table of Contents

Background

The **Semantic Spotter Project** aims to revolutionize document search and retrieval in the insurance industry. Traditional **keyword-based search** mechanisms often fail to capture the **semantic meaning** of queries, leading to inaccurate or incomplete search results.

This project leverages **Retrieval-Augmented Generation (RAG)** to **enhance query responses** by combining **retrieved document snippets** with **generative AI capabilities**. The system ensures that answers are **factual, context-aware, and document-backed**, thereby improving trust and reliability in insurance-related searches.

Problem Statement

Challenges in Insurance Document Search

Insurance policy documents are often:

- **Lengthy and complex**, making manual search inefficient.
- **Full of legal jargon**, requiring expert interpretation.
- **Difficult to search using traditional methods**, as policies often contain synonymous terms that keyword-based search engines miss.

Objective of the Semantic Spotter Project

The **goal of the project** is to build a **robust, AI-powered generative search system** that can:

- Accurately **retrieve and extract relevant content** from a large set of policy documents.
- Provide **contextually accurate** answers to complex queries.
- Minimize **hallucinations in AI-generated responses** by grounding answers in actual document text.
- Ensure fast and **scalable retrieval** of policy-related information.

By addressing these challenges, the **Semantic Spotter Project** aims to significantly improve how users interact with **policy documents** and **insurance-related information**.

Approach

The project follows a **multi-layered retrieval and generation approach**:

1. **Preprocessing & Storage**:
    - Extract text from **insurance policy documents**.
    - **Split large documents** into smaller, meaningful text chunks.
    - **Convert text into embeddings** using OpenAI's `text-embedding-ada-002`.
    - Store embeddings in a **vector database (ChromaDB)** for fast retrieval.
2. **Retrieval & Reranking**:
    - Perform **semantic similarity search** using Maximal Marginal Relevance (**MMR**) to ensure **diverse, relevant** results.
    - Apply **cross-encoder reranking** to refine search accuracy.
3. **Query Processing & Response Generation**:
    - Retrieve **relevant document snippets** based on user queries.
    - Pass the **retrieved context** to **GPT-based generative AI** for **answer generation**.
    - Format and **present the response** in a structured manner with citations.
4. **Output Validation**:
    - Provide **source document references** for user verification.
    - Ensure **legal compliance** by restricting model output to policy-specific content.

System Layers

The project is divided into multiple layers, each handling a specific function:

1. **Data Processing Layer**:
    - Extracts and **cleans** text from PDF policy documents.
    - Splits documents into **manageable chunks**.
2. **Embedding & Storage Layer**:
    - Converts text into **vector embeddings**.
    - Stores embeddings in **ChromaDB** for fast retrieval.
3. **Retrieval & Ranking Layer**:

    - Fetches relevant document snippets from the vector store.
    - Reranks retrieved documents using **cross-encoder-based reranking**.
4. **RAG Generation Layer**:
    - Generates responses using **GPT-4**, incorporating retrieved document snippets.
    - Formats the output into a **readable and structured response**.
5. **Validation & Output Layer**:
    - Ensures that generated responses include **document citations**.
    - Filters irrelevant or misleading content.

System Architecture

1. Document Preprocessing & Storage

- Extracts text from **insurance PDFs**.
- Splits text into **smaller chunks**.
- Converts text chunks into **vector embeddings**.
- Stores embeddings in **ChromaDB**.

2. Query Processing

- Accepts user queries.
- Converts queries into **vector representations**.
- Uses **semantic similarity search** to retrieve **top-matching document snippets**.

3. Context-Enhanced Generation

- Formats retrieved documents into a **structured RAG prompt**.
- Uses **GPT-4** to generate **contextually aware** responses.

4. Response Validation & Output

- Outputs the AI-generated response **with citations** from policy documents.
- Ensures **legal and factual accuracy**.

Prerequisites

To run this project, you need:

1. **Python 3.8+**
2. **OpenAI API Key** (for embeddings and GPT-4 responses)
3. **ChromaDB** (for vector storage)

4. **PyPDF** (for document extraction)
5. **LangChain** (for retrieval & LLM integration)

Install Required Dependencies

```
pip install -r requirements.txt
```

Setup Project

1. Clone the Repository

```
git clone https://github.com/ivineettiwari/Semantic-Spotter-prj.git
cd Semantic-Spotter-prj
```

2. Set Up API Key

```
echo "your-openai-api-key" > API_Key.txt
```

3. Prepare Data

- Place insurance policy PDFs in the `Policy_Documents/` folder.

4. Run the Project

```
Run Jupyter NoteBook Code.
```

Usage

1. Run a Query

```
query = "What is the life insurance coverage for disability?"
response = rag_chain.invoke(query)
print(response)
```

2. Retrieve Relevant Documents

```
retriever = get_retriever(50)
retrieved_docs = retriever.invoke("What happens if a person dies in an
accident without wearing a seatbelt?")
for doc in retrieved_docs:
    print(doc.page_content[:300])  # Preview first 300 characters
```

Example Queries

1. "Can a 100-year-old person apply for term insurance?"
2. "What are the key benefits of life insurance?"
3. "What are the conditions for claim rejection?"
4. "What is the eligibility for HDFC group insurance?"
5. "Explain HDFC Life Sanchay Plus Life Long Income Option."

Future Enhancements

1. **Hybrid Search (Vector + BM25 Keyword Search)**
2. **Real-Time Document Updates & Dynamic Indexing**
3. **Confidence Scoring & Response Ranking**
4. **API Deployment Using FastAPI**
5. **Support for OCR-based Text Extraction from Scanned PDFs**

Conclusion

The **Semantic Spotter Project** successfully integrates **semantic search, document retrieval, and generative AI** to provide **factually backed answers** for insurance queries. By combining **ChromaDB for retrieval** and **GPT-4 for generation**, the system ensures **accurate, structured, and reliable** responses.

This project has significant potential applications in **insurance customer support, legal advisory, and document search automation**. Future enhancements will further **improve search accuracy, retrieval speed, and multi-modal capabilities**.

This document provides a **comprehensive submission report** for your project. Let me know if you'd like any refinements.