HelpMateAI-Project

Introduction

HelpMateAI is an **AI-powered Retrieval-Augmented Generation (RAG) system** designed to assist with **insurance-related queries**. It efficiently retrieves information from insurance documents and generates precise responses using **LLMs like GPT-3.5**. The system incorporates **semantic search**, **cross-encoder re-ranking**, and **LLM-based response generation** to ensure that answers are fact-based and contextually accurate.

The project leverages **vector embeddings** and **retrieval mechanisms** to enhance the accuracy of insurance-related information retrieval. It is designed to work with a **predefined set of insurance documents** that are preprocessed and stored in a **vector database**. Users can ask **natural language queries**, and the system fetches the most relevant sections, refines the ranking using a **cross-encoder**, and generates a well-structured answer with citations.

Problem Statement

Insurance policies and legal documents are often lengthy and difficult to navigate. Users, including **policyholders, agents, and legal teams**, may need quick access to **specific details** regarding coverage, claims, and policy terms.

Challenges:

1. **Complexity of Documents**: Insurance documents contain technical jargon and are often **hundreds of pages long**.
2. **Time-Consuming Manual Search**: Finding specific clauses in large documents requires **manual reading** and **keyword-based searches**, which may **miss context**.
3. **Inconsistent Answers**: Traditional **search engines** return **entire documents**, leaving users to **extract relevant information manually**.
4. **Contextual Relevance**: Simple keyword-based searches lack **semantic understanding** and may return **irrelevant sections**.

Goal: Develop an AI-driven system that **automates the extraction of relevant insurance policy details**, **ranks retrieved information accurately**, and **generates precise answers with citations**.

Approach

The project uses **Retrieval-Augmented Generation (RAG)**, which combines **document retrieval** and **text generation**.

1. **Document Processing & Storage**
    - Extracts text from **PDF insurance documents** using **pdfplumber**.
    - Splits documents into **fixed-size text chunks**.
    - Converts chunks into **vector embeddings** using **SentenceTransformer**.
    - Stores embeddings in a **vector database (ChromaDB)** for **fast retrieval**.
2. **Query Processing & Document Retrieval**
    - Encodes the **user's query** into an **embedding** using the same model.
    - Performs **similarity search** in **ChromaDB** to retrieve the **top-K relevant chunks**.
    - Applies a **Cross-Encoder model** to **re-rank** the retrieved results for **higher accuracy**.
3. **Answer Generation & Citation**
    - The **top 3 most relevant chunks** are passed to **GPT-3.5**.
    - The model generates a **well-structured answer**, **grounded in retrieved documents**.
    - Includes **citations** (document name and page number).
4. **User Query Handling & API Support**
    - Allows users to **send queries via a Python script or API**.
    - Retrieves **precise** insurance-related information.
    - Provides a **concise answer** instead of entire documents.

System Design

The system consists of three key components:

1. Document Processing & Vector Store

- **Preprocessing**: Extract text from PDFs.
- **Chunking**: Divide text into **fixed-size blocks**.
- **Embedding**: Convert chunks into **numerical vector representations**.
- **Storage**: Save vectors in **ChromaDB**.

2. Query Processing & Retrieval

- **Query Embedding**: Convert user query into a **vector representation**.
- **Vector Search**: Find similar embeddings in **ChromaDB**.
- **Re-ranking**: Use a **Cross-Encoder model** to rank retrieved results.

3. Answer Generation

- **Input**: User query + retrieved document chunks.
- **LLM Processing**: GPT-3.5 generates an answer based on the provided context.
- **Output**: A precise, **cited** response.

Current Implementation

The system follows a **structured workflow** to **retrieve and generate responses**. Below is the core logic:

1. Query Processing

```
query = "What is the life insurance coverage for disability?"
df = search(query)  # Retrieve relevant documents
df = apply_cross_encoder(query, df)  # Re-rank results
df = get_topn(3, df)  # Select the top 3 most relevant chunks
response = generate_response(query, df)  # Generate final response
print("\n".join(response))  # Print the response
```

2. Optimized Query Handling Using Loops

```
queries = [
    "what is the life insurance coverage for disability",
    "what is the Proof of ADL Disability or Total Disability",
    "what is condition of death while not wearing Seat Belt"
]

def process_queries(queries):
    results = {}
    for query in queries:
        df = search(query)
        df = apply_cross_encoder(query, df)
        df = get_topn(3, df)
        response = generate_response(query, df)
        results[query] = "\n".join(response)
        print(f"\nQuery: {query}\n")
        print(results[query])

    return results

responses = process_queries(queries)
```

3. API Endpoint for External Integration

```
from fastapi import FastAPI
from pydantic import BaseModel

app = FastAPI()

class QueryRequest(BaseModel):
    query: str

@app.post("/query")
def answer_query(request: QueryRequest):
    df = search(request.query)
    df = apply_cross_encoder(request.query, df)
```

```
    df = get_topn(3, df)
    response = generate_response(request.query, df)
    return {"query": request.query, "response": response}
```

Project Setup

1. Clone the Repository

```
git clone https://github.com/ivineettiwari/HelpMateAI-Project.git
cd HelpMateAI-Project
```

2. Create a Virtual Environment

```
python -m venv env
source env/bin/activate  # On Mac/Linux
env\Scripts\activate     # On Windows
```

3. Install Dependencies

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

4. Set Up API Keys

Create a `.env` file and add your **OpenAI API Key**:

```
OPENAI_API_KEY=your-api-key-here
```

5. Run the Application

```
View Jupyter Notebook
```

Future Scope

- **Support for Additional Domains**: Extend beyond insurance to **legal, finance, and healthcare documents**.
- **Hybrid Search**: Combine **vector search + keyword-based search** for enhanced accuracy.
- **Model Fine-Tuning**: Fine-tune **cross-encoders** and **LLMs** on domain-specific datasets.
- **Multi-Language Support**: Expand support for **queries in multiple languages**.
- **Improved UI & API**: Build a **web-based front-end** for easier interaction.

Conclusion

The **HelpMateAI-Project** provides a powerful **AI-driven document retrieval system** tailored for insurance-related queries. By leveraging **vector embeddings, cross-encoder ranking, and GPT-3.5**, it offers:

1. **Fast and accurate information retrieval**.
2. **Well-structured answers with citations**.
3. **Scalability for large insurance datasets**.

This **open-source project** can be further enhanced by integrating **real-time policy updates, UI improvements, and domain-specific model optimizations**.

For more details and contributions, visit the GitHub repository:
HelpMateAI-Project