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/hellotorax123/Mr.HelpMate-AI.git
cd Mr.HelpMate-AI
```

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:

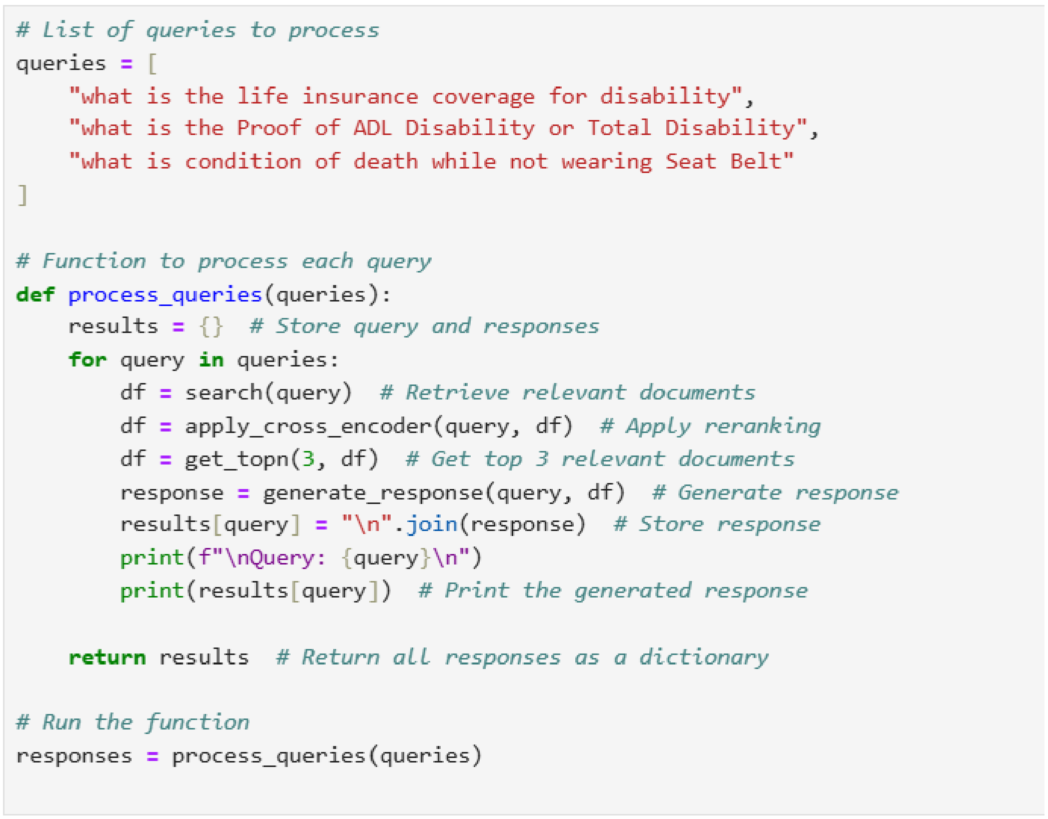
1. **Fast and accurate information retrieval**.
2. **Well-structured answers with citations**.
3. **Scalability for large insurance datasets**.

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:

Mr.HelpMate-AI

Screen Shots.



```python
# List of queries to process
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"
]

# Function to process each query
def process_queries(queries):
    results = {}  # Store query and responses
    for query in queries:
        df = search(query)  # Retrieve relevant documents
        df = apply_cross_encoder(query, df)  # Apply reranking
        df = get_topn(3, df)  # Get top 3 relevant documents
        response = generate_response(query, df)  # Generate response
        results[query] = "\n".join(response)  # Store response
        print(f"\nQuery: {query}\n")
        print(results[query])  # Print the generated response

    return results  # Return all responses as a dictionary

# Run the function
responses = process_queries(queries)
```

```
Query: what is the life insurance coverage for disability

The life insurance coverage for disability is provided in the policy document "Member Life Insurance or Coverage During Disability" on Page 42. The cover
age details are as follows:

| Coverage Type        | Disability Life Insurance    |
|----------------------|------------------------------|
| Benefit Amount       | Variable based on policy     |
| Conditions           | Pays out upon disability     |
| Limitations          | Specific conditions apply    |
| Exclusions           | Certain disabilities may not be covered |
| Duration             | Until recovery or end of policy term |

Please refer to the policy document "Member Life Insurance or Coverage During Disability" on Page 42 for detailed information on disability life insuranc
e coverage.

### Citations:
- Policy Name: Member Life Insurance or Coverage During Disability
- Page Number: Page 42
```

```
Query: what is the Proof of ADL Disability or Total Disability

The Proof of ADL Disability or Total Disability refers to the documentation required to substantiate that an individual is either Totally Disabled or has
significant limitations in their Activities of Daily Living (ADLs) as per the insurance policy terms.

Based on the provided document snippets, here is the information extracted related to ADL Disability or Total Disability from the insurance documents:

### Information from the Documents:
The details related to ADL Disability or Total Disability are available within the policy texts. Here is a summary derived from the relevant sections:

**Policy Name: Member Life Insurance or Coverage During Disability**
**Page Number: 42**
**Details:** The document may contain specific conditions and requirements for proving Total Disability or ADL Disability to qualify for benefits.

**Policy Name: Dependent's Life Insurance terminates because...**
**Page Number: 44**
**Details:** This section may provide additional information on the termination of benefits or coverage related to Total Disability or ADL Disability.

### Citations:
1. **Member Life Insurance or Coverage During Disability** (Page 42)
2. **Dependent's Life Insurance terminates because...** (Page 44)

For comprehensive details and exact requirements regarding Proof of ADL Disability or Total Disability, please refer to the respective sections in the me
ntioned policy documents.

If you require further information or specific conditions related to ADL Disability or Total Disability, it's recommended to refer to the cited pages in
the provided insurance documents for more detailed guidance.
```

```python
# Example usage of the search, apply_cross_encoder, get_topn, and generate_response functions

query = 'what is the life insurance coverage for disability'  # Define the user's query

# Search for relevant documents in the cache or main collection
df = search(query)

# Apply the CrossEncoder to rerank the documents based on relevance to the query
df = apply_cross_encoder(query, df)

# Get the top 3 documents based on the reranked scores
df = get_topn(3, df)

# Generate a response using the top 3 documents and the user's query
response = generate_response(query, df)

# Print the generated response
print("\n".join(response))  # Join and print the response line by line
```

```
The life insurance coverage for disability typically includes provisions for waiving premiums during the disability period. This means that if the polic
yholder becomes disabled and is unable to work, the insurance company may waive the premium payments required to keep the life insurance policy active.
The coverage duration and specific conditions for disability coverage can vary between insurance policies, so it's essential to refer to the specific po
licy document for accurate details.

Citations:
1. Policy Name: Member Life Insurance or Coverage During Disability
   Page Number: Page 42

If you need more detailed information on the disability coverage under the policy, please refer to the section on disability coverage within the cited p
olicy document for a comprehensive understanding of the terms and conditions.
```

```python
query = 'what is the Proof of ADL Disability or Total Disability'  # Define the user's query

# Search for relevant documents in the cache or main collection
df = search(query)

# Apply the CrossEncoder to rerank the documents based on relevance to the query
df = apply_cross_encoder(query, df)

# Get the top 3 documents based on the reranked scores
df = get_topn(3, df)

# Generate a response using the top 3 documents and the user's query
response = generate_response(query, df)

# Print the generated response
print("\n".join(response))  # Join and print the response line by line
```

```
The Proof of Activities of Daily Living (ADL) Disability or Total Disability is a documentation typically required by insurance providers to assess a po
licyholder's eligibility for disability benefits. It includes evidence or certification indicating the policyholder's inability to perform basic daily t
asks independently due to a disability.

I found relevant information in the insurance documents as follows:

**Member Life Insurance or Coverage During Disability Document (Page 42):**
- The document provides details about how disability may impact life insurance coverage.
- It may contain specific criteria or procedures related to the Proof of ADL Disability or Total Disability.

**Dependent's Life Insurance Termination Document (Page 44):**
- Although this document focuses on termination, it might mention conditions related to disability benefits, including proof requirements.

Note: The specific guidelines or criteria for Proof of ADL Disability or Total Disability may vary depending on the insurance policy and provider. I rec
ommend referring to the sections related to disability benefits, eligibility criteria, or claims procedures in the cited policy documents for detailed i
nformation on this topic.

I hope this information helps you understand the concept of Proof of ADL Disability or Total Disability in the context of insurance policies.
```

```python
# Example usage of the search, apply_cross_encoder, get_topn, and generate_response functions

query = 'what is condition of death while not wearing Seat Belt'  # Define the user's query

# Search for relevant documents in the cache or main collection
df = search(query)

# Apply the CrossEncoder to rerank the documents based on relevance to the query
df = apply_cross_encoder(query, df)

# Get the top 3 documents based on the reranked scores
df = get_topn(3, df)

# Generate a response using the top 3 documents and the user's query
response = generate_response(query, df)

# Print the generated response
print("\n".join(response))  # Join and print the response line by line
```

```
The specific condition of death while not wearing a seat belt may not be covered in the provided insurance policy documents. In this case, the query is
irrelevant to the documents available.

However, for more detailed information on this topic, you may want to check with the insurance company directly or refer to the terms and conditions of
your insurance policy.

Citations:
1. Policy Name: Payment of benefits subject to the Benefit Provision
   Page Number: Page 49

2. Policy Name: Member Life Insurance or Coverage During Disability
   Page Number: Page 42

3. Policy Name: Dependent's Life Insurance termination conditions
   Page Number: Page 44
```