Task 1: Chat with PDF Using RAG Pipeline

Overview

The goal is to implement a Retrieval-Augmented Generation (RAG) pipeline that allows users to interact with semi-structured data in multiple PDF files. The system should extract, chunk, embed, and store the data for efficient retrieval. It will answer user queries and perform comparisons accurately, leveraging the selected LLM model for generating responses.

Functional Requirements

1. Data Ingestion

- **Input**: PDF files containing semi-structured data.
- **Process**:
    - Extract text and relevant structured information from PDF files.
    - Segment data into logical chunks for better granularity.
    - Convert chunks into vector embeddings using a pre-trained embedding model.
    - Store embeddings in a vector database for efficient similarity-based retrieval.

2. Query Handling

- **Input**: User's natural language question.
- **Process**:
    - Convert the user's query into vector embeddings using the same embedding model.
    - Perform a similarity search in the vector database to retrieve the most relevant chunks.
    - Pass the retrieved chunks to the LLM along with a prompt or agentic context to generate a detailed response.

3. Comparison Queries

- **Input**: User's query asking for a comparison
- **Process**:
    - Identify and extract the relevant terms or fields to compare across multiple PDF files.
    - Retrieve the corresponding chunks from the vector database.
    - Process and aggregate data for comparison.
    - Generate a structured response (e.g., tabular or bullet-point format).

4. Response Generation

- **Input**: Relevant information retrieved from the vector database and the user query.
- **Process**:
    - Use the LLM with retrieval-augmented prompts to produce responses with exact values and context.
    - Ensure factuality by incorporating retrieved data directly into the response.

Example Data:

https://www.hunter.cuny.edu/dolciani/pdf_files/workshop-materials/mmc-presentations/tables-charts-and-graphs-with-examples-from.pdf

Extract accurate information:

1. From page 2 get the exact unemployment information based on type of degree input
2. From page 6 get the tabular data

Task 2: Chat with Website Using RAG Pipeline

Overview

The goal is to implement a Retrieval-Augmented Generation (RAG) pipeline that allows users to interact with structured and unstructured data extracted from websites. The system will crawl, scrape, and store website content, convert it into embeddings, and store it in a vector database. Users can query the system for information and receive accurate, context-rich responses generated by a selected LLM.

Functional Requirements

1. Data Ingestion

- **Input**: URLs or list of websites to crawl/scrape.
- **Process**:
    - Crawl and scrape content from target websites.
    - Extract key data fields, metadata, and textual content.
    - Segment content into chunks for better granularity.
    - Convert chunks into vector embeddings using a pre-trained embedding model.
    - Store embeddings in a vector database with associated metadata for efficient retrieval.

2. Query Handling

- **Input**: User's natural language question.
- **Process**:
    - Convert the user's query into vector embeddings using the same embedding model.
    - Perform a similarity search in the vector database to retrieve the most relevant chunks.
    - Pass the retrieved chunks to the LLM along with a prompt or agentic context to generate a detailed response.
    -

3. Response Generation

- **Input**: Relevant information retrieved from the vector database and the user query.
- **Process**:
    - Use the LLM with retrieval-augmented prompts to produce responses with exact values and context.
    - Ensure factuality by incorporating retrieved data directly into the response.

Example website links :

https://www.uchicago.edu/
https://www.washington.edu/
https://www.stanford.edu/
https://und.edu/