Intelligent Document Processing and Agent System

System Overview

This system is designed to process documents, store embeddings for vector-based search, and execute natural language queries using reasoning agents. The architecture integrates LangChain's capabilities for document processing, embedding generation, vector search, and question-answering (QA) pipelines, with additional agent systems for data analytics and SQL queries.

System Features

1. **Document Processing:**
   - Parse PDF documents and preprocess content into manageable chunks.
2. **Embedding and Vector Search:**
   - Generate embeddings for textual data using a HuggingFace embedding model.
   - Store embeddings in a Chroma vector database.
   - Perform similarity searches for user queries.
3. **Reasoning Agents:**
   - **Conversational Agent:** Answer natural language queries conversationally.
   - **Data Analytics Agent:** Perform complex operations on tabular data.
   - **SQL Agent:** Execute SQL queries on structured databases.
4. **Question Answering:**
   - Use OpenAI's LLMs for generating answers from retrieved content.

System Architecture

1. Document Processing Layer

Components:

- **PDF Loader:** Uses LangChain's `PyPDFLoader` to extract content from PDF files.
- **Chunking Mechanism:** Splits extracted content into smaller chunks for efficient processing.

Workflow:

1. Load a PDF file.
2. Extract pages as text documents.
3. Chunk each document into smaller units of a specified size (e.g., 100 characters).

2. Embedding Layer

Components:

- **HuggingFace Embedding Model:** Generates dense vector embeddings for documents.

Workflow:

1. Input document chunks.
2. Use the HuggingFace embedding model to generate embeddings.
3. Store embeddings in the vector database.

3. Search Layer

Components:

- **Chroma Vector Store:** Stores and retrieves embeddings for similarity searches.

Workflow:

1. Index the generated embeddings in the Chroma database.
2. Perform a similarity search for a query embedding.
3. Retrieve the top-k most similar chunks.

4. Question-Answering Layer

Components:

- **OpenAI LLM:** Processes input queries and generates answers based on retrieved document chunks.
- **LangChain QA Chain:** Combines input documents and queries to create a coherent response.

Workflow:

1. Retrieve the top-k chunks from the Search Layer.
2. Input the chunks and user query into the QA chain.
3. Generate an answer using the OpenAI LLM.

5. Agent Systems

Components:

1. **Conversational Agent:** Responds conversationally to queries.
2. **Data Analytics Agent:** Performs operations on a pandas DataFrame.
3. **SQL Agent:** Executes SQL queries on the policies database.

Workflow:

1. **Conversational Agent:**

    - Use LLM tools to respond to natural language queries.
2. **Data Analytics Agent:**
    - Execute operations on the DataFrame, such as calculating policy durations.
3. **SQL Agent:**
    - Connect to the SQLite database.
    - Execute SQL queries and return results.

Execution Flow

1. **Input Processing:**
    - Load the PDF document and extract content.
    - Chunk the document for efficient embedding and search.
2. **Embedding and Indexing:**
    - Generate embeddings for document chunks using the embedding model.
    - Index embeddings into the Chroma vector store.
3. **Query Handling:**
    - For each query:
        - Generate embeddings for the query.
        - Retrieve the top-k relevant chunks from the vector database.
        - Generate an answer using the QA chain.
4. **Agent Execution:**
    - Execute conversational, analytics, or SQL queries as needed.

Example Use Case

Input

- **Document:** Life insurance policy PDF.
- **Queries:**
    1. "What are the benefits included under the Group Policy for Life Insurance?"
    2. "What is considered a 'Qualifying Event' for Accelerated Benefits?"

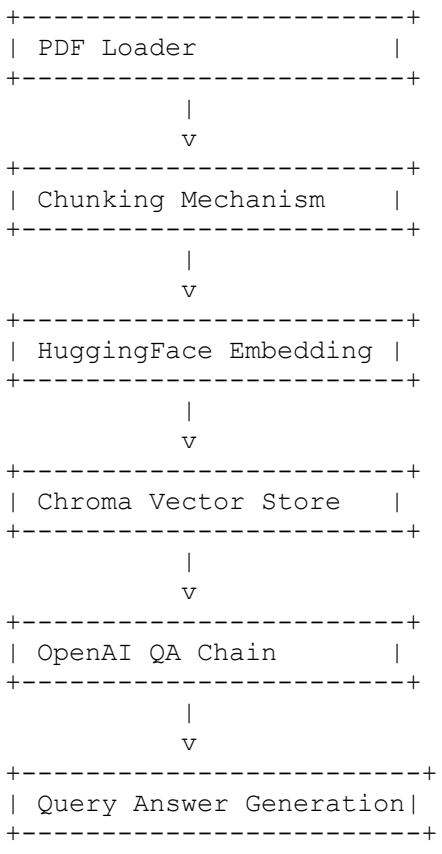
Process

1. Load the PDF and extract content.
2. Chunk the document.
3. Generate embeddings and store them in the vector database.
4. For each query, retrieve relevant chunks and generate an answer.

Output

- **Generated Answer:** "The Group Policy for Life Insurance includes benefits for terminal illness diagnosis and accelerated benefits for qualifying events."

System Design Diagram



Code Integration

Key Libraries

- LangChain
- HuggingFace
- Chroma
- OpenAI
- PyPDF2
- SQLite
- Pandas

Key Components

1. **EmbeddingLayer** - Handles embedding generation and chunking.
2. **SearchLayer** - Handles vector search and retrieval.
3. **GenerationLayer** - Handles QA generation.
4. **AgentSystems** - Provides reasoning and analytics capabilities.

Conclusion

This system efficiently processes documents, supports interactive querying, and integrates advanced reasoning capabilities. It can be extended to other domains such as financial analysis, legal document processing, or academic research.