Data Ingestion and Retrieval Pipeline Practice (RAG)

This repository contains a learning session focused on building a Retrieval-Augmented Generation (RAG) pipeline using **LangChain**. It covers the essential steps of data ingestion, chunking, embedding generation, and vector retrieval.

Overview

The primary focus is on understanding how to load various document formats, split them into manageable chunks, create vector embeddings, and store them in a vector database for efficient retrieval.

Key Concepts Covered

1. Data Ingestion (Loaders)

We explore different ways to load data using `langchain_community.document_loaders`: - **TextLoader**: Loading simple text files (`.txt`). - **PDFMinerLoader**: Loading PDF documents (`.pdf`). - **DirectoryLoader**: Loading multiple files from a directory, demonstrating integration with both text and PDF loaders (using `PyMuPDFLoader`).

2. Text Splitting (Chunking)

To prepare the data for embedding, large documents are split into smaller chunks: - **RecursiveCharacterTextSplitter**: Used to split text based on characters with specific chunk size and overlap to maintain context. - *Configuration used:* Chunk size: 400, Chunk overlap: 200.

3. Embeddings

We use a pre-trained model to generate vector embeddings for the text chunks: - **SentenceTransformer**: Utilizing the `all-MiniLM-L6-v2` model from `sentence-transformers` to encode text into dense vector representations.

4. Vector Store (Retrieval)

For storing and querying the embeddings: - **ChromaDB**: An open-source vector database. - **Process**: - Initializing a ChromaDB client. - Creating a collection (`my_embeddings`). - Adding documents and their embeddings to the collection. - **Querying**: Performing semantic search to retrieve relevant documents based on a query (e.g., "List out company names in which vijay is experienced").

Dependencies

To run the notebook, ensure you have the following libraries installed:

- `langchain`
- `langchain_community`
- `chromadb`
- `sentence-transformers`
- `scikit-learn` (for cosine similarity if needed)
- `pdfminer.six` (for PDFMinerLoader)
- `pymupdf` (for PyMuPDFLoader)

Usage

1. Navigate to the `notebooks` directory.

2. Open `document.ipynb`.
3. Run the cells to see the step-by-step execution of the pipeline.