LAB No. 14

Document Loading using LangChain for Retrieval-Augmented Generation (RAG)

This lab introduces students to Document Loaders in LangChain, a key component of **Retrieval-Augmented Generation (RAG)** systems. Students will learn how to load, preprocess, and structure data from different document formats such as text and PDF files. By converting documents into LangChain's Document objects, students will understand how external knowledge can be prepared and supplied to large language models for improved, context-aware responses.

LAB Objectives

- Understand the role of document loaders in RAG
- Load data from multiple file formats using LangChain
- Inspect document metadata and content
- Prepare documents for downstream tasks like chunking and retrieval

Tools & Libraries

- Python 3.9+
- Required libraries:
  - langchain
  - langchain-community
  - pypdf
  - unstructured

Lab Tasks (Practice Steps)

Task 1: Environment Setup

- Create a virtual environment
- Install required LangChain libraries
- Verify installation

Task 2: Understand the Main Concept – Document Loaders

- Study the role of **Document Loaders** in RAG
- Explain how loaders convert raw data into LangChain Document objects

Task 3: Load PDF Data (PyPDFLoader)

- Load lecture_notes.pdf
- Count total pages
- Display content of first page
- Attach code with output screenshot

Task 4: Load Web Data (WebBaseLoader)

- Use **WebBaseLoader** to load a webpage
- Extract main textual content
- Observe metadata (URL source)
- Attach code with output screenshot

Task 5: Load Structured Data (CSVLoader)

- Load students.csv
- Inspect how rows are converted into documents
- Print one document sample
- Attach code with output screenshot

Task 6: Compare All Loaders

Students must compare:

- Content format
- Metadata fields
- Attach code with output screenshot

Lab Questions

1. **What is the role of document loaders in RAG?**

Document loaders are tools that read and import data from different sources (like PDFs, Word files, websites, or databases) into a system. In RAG (Retrieval-Augmented Generation), they are used to collect and organize documents so the AI can retrieve relevant information when answering questions. Without document loaders, the AI wouldn't have access to the source knowledge it needs for accurate responses.

Key point: They act as the bridge between raw documents and the AI system.

2. **Why is metadata important in LangChain documents?**

Metadata is extra information about a document, like its title, author, date, or topic. In LangChain, metadata helps the AI filter, organize, and retrieve the right documents quickly. It makes search and retrieval more accurate and efficient, especially when dealing with large collections of documents.

Key point: Metadata is like a label system that helps the AI know what each document is about.

3. **Difference between TextLoader and PyPDFLoader?**

    TextLoader: Loads plain text files (like `.txt`). Simple and fast for text-based documents.

    PyPDFLoader: Loads PDF files by extracting text from PDFs. Can handle multi-page documents and maintain structure.

Key point: TextLoader = plain text files
PyPDFLoader = PDF files

4. **What happens if a PDF has scanned images instead of text?**

    If a PDF contains scanned images of text, loaders like PyPDFLoader cannot read it because they only extract digital text, not images. To use such PDFs, you need OCR (Optical Character Recognition) tools, which convert images of text into actual readable text for the AI.

5. **Why is directory-based loading useful in real applications?**

Directory-based loading lets the AI automatically load all documents from a folder at once, instead of loading files one by one. This is useful in real applications because data often comes in large batches or multiple files, saving time and making the system more efficient and scalable.

Key point: It's a time-saver for handling many documents at once.

6. **How does document quality affect RAG performance?**

AG relies on documents to retrieve accurate information. If documents are incomplete, outdated, or poorly written, the AI may give wrong or confusing answers. High-quality, well-structured documents ensure the AI retrieves relevant and correct information, improving overall performance.

Key point: Better document quality = more accurate AI responses.