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

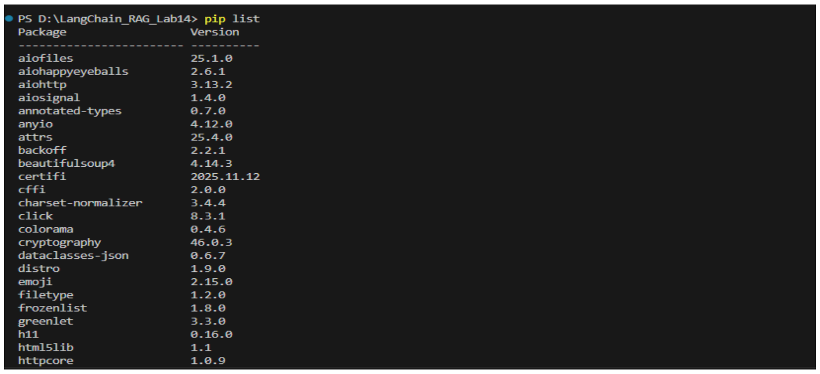
Tools & Libraries

- 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



```
PS D:\LangChain_RAG_Lab14> pip list
Package                  Version
------------------------ ----------
aiofiles                 25.1.0
aiohappyeyeballs         2.6.1
aiohttp                  3.13.2
aiosignal                1.4.0
annotated-types          0.7.0
anyio                    4.12.0
attrs                    25.4.0
backoff                  2.2.1
beautifulsoup4           4.14.3
certifi                  2025.11.12
cffi                     2.0.0
charset-normalizer       3.4.4
click                    8.3.1
colorama                 0.4.6
cryptography             46.0.3
dataclasses-json         0.6.7
distro                   1.9.0
emoji                    2.15.0
filetype                 1.2.0
frozenlist               1.8.0
greenlet                 3.3.0
h11                      0.16.0
html5lib                 1.1
httpcore                 1.0.9
```

Task 2: Understand the Main Concept – Document Loaders

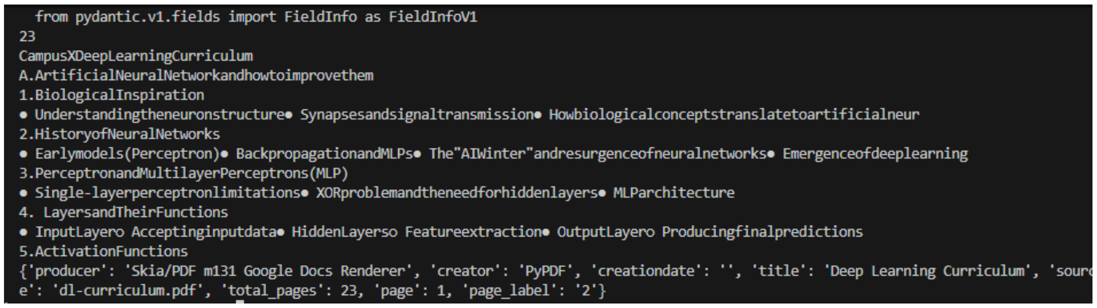
- Study the role of **Document Loaders** in RAG
- Explain how loaders convert raw data into LangChain Document objects

Document loaders in LangChain are used to load data from different sources such as text files, PDF documents, CSV files, and web pages. In this lab, different loaders like TextLoader, PyPDFLoader, CSVLoader, WebBaseLoader, and DirectoryLoader are used. These loaders convert raw data into LangChain Document objects. Each Document object contains page_content, which stores the extracted text, and metadata, which stores information such as file name, page number, or URL. In Retrieval-Augmented Generation (RAG), document loaders prepare external knowledge so that it can be retrieved and supplied to large language models for generating accurate and context-aware response

Task 3: Load PDF Data (PyPDFLoader)

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

```
from langchain_community.document_loaders import PyPDFLoader

loader = PyPDFLoader('dl-curriculum.pdf')

docs = loader.load()

print(len(docs))

print(docs[0].page_content)

print(docs[1].metadata)
```

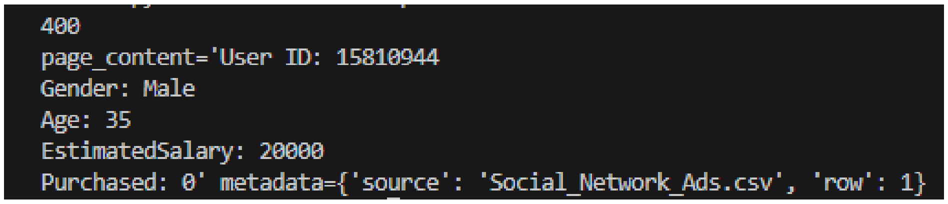


```
  from pydantic.v1.fields import FieldInfo as FieldInfoV1
23
CampusXDeepLearningCurriculum
A.ArtificialNeuralNetworkandhowtoimprovethem
1.BiologicalInspiration
• Understandingtheneuronstructure• Synapsesandsignaltransmission• Howbiologicalconceptstranslatetoartificialneur
2.HistoryofNeuralNetworks
• Earlymodels(Perceptron)• BackpropagationandMLPs• The"AIWinter"andresurgenceofneuralnetworks• Emergenceofdeeplearning
3.PerceptronandMultilayerPerceptrons(MLP)
• Single-layerperceptronlimitations• XORproblemandtheneedforhiddenlayers• MLParchitecture
4. LayersandTheirFunctions
• InputLayero Acceptinginputdata• HiddenLayerso Featureextraction• OutputLayero Producingfinalpredictions
5.ActivationFunctions
{'producer': 'Skia/PDF m131 Google Docs Renderer', 'creator': 'PyPDF', 'creationdate': '', 'title': 'Deep Learning Curriculum', 'sourc
e': 'dl-curriculum.pdf', 'total_pages': 23, 'page': 1, 'page_label': '2'}
```

Task : 4 Structured Data (CSVLoader)

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

```
from langchain_community.document_loaders import CSVLoader

loader = CSVLoader(file_path='Social_Network_Ads.csv')

docs = loader.load()


print(len(docs))

print(docs[1])
```



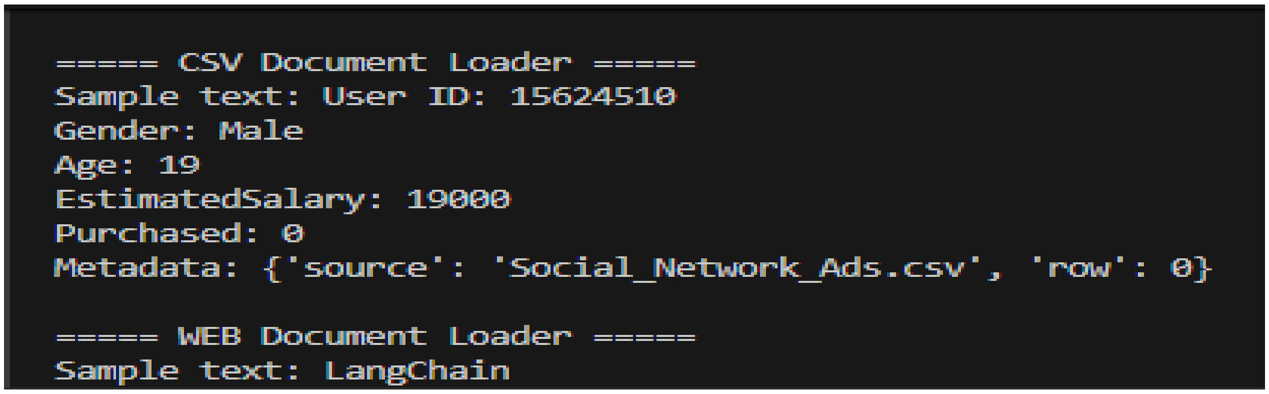
```
400
page_content='User ID: 15810944
Gender: Male
Age: 35
EstimatedSalary: 20000
Purchased: 0' metadata={'source': 'Social_Network_Ads.csv', 'row': 1}
```

Task 6: Compare All Loaders

Students must compare:

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

```
from langchain_community.document_loaders import PyPDFLoader, CSVLoader,
WebBaseLoader, TextLoader

loaders = {

   "PDF": PyPDFLoader("dl-curriculum.pdf"),

   "CSV": CSVLoader("Social_Network_Ads.csv"),

   "WEB": WebBaseLoader("https://www.langchain.com"),

   "TEXT": TextLoader("cricket.txt")

}

for name, loader in loaders.items():

   docs = loader.load()

   print(f"\n===== {name} Document Loader =====")

   print("Sample text:", docs[0].page_content[:200])

   print("Metadata:", docs[0].metadata)
```



```
===== CSV Document Loader =====
Sample text: User ID: 15624510
Gender: Male
Age: 19
EstimatedSalary: 19000
Purchased: 0
Metadata: {'source': 'Social_Network_Ads.csv', 'row': 0}

===== WEB Document Loader =====
Sample text: LangChain
```

COMPARISON TABLE:

Loader	**Content Format**	**Metadata**
PyPDFLoader	Page-wise text	page number, source
WebBaseLoader	Clean web text	URL
CSVLoader	Row-wise text	file path, row index

Lab Questions

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

Answer:

They load external data and convert it into **Document objects** so LLMs can retrieve and use knowledge

2. Why is metadata important in LangChain documents?

Answer:

Metadata helps track **source, page number, URL**, improving accuracy and traceability.

3. Difference between TextLoader and PyPDFLoader?

Answer:

TextLoader: Loads plain text files

PyPDFLoader: Loads PDF files page by page

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

Answer:

Text cannot be extracted without **OCR**, so content will be empty or unreadable.

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

Answer:

It allows loading **multiple files automatically**, useful for large document collections.

6. How does document quality affect RAG performance?

Answer:

Clean, accurate documents give **better retrieval and more correct answers**.