LangChain Document Loaders

1. RAG Context (for background)

- **RAG (Retrieval-Augmented Generation)** combines:
    - **Information Retrieval** → fetching relevant documents from a knowledge base.
    - **Language Generation** → using those documents as context to create accurate and grounded answers.
- **Benefits of RAG**:

1. Uses up-to-date information.
2. Provides better privacy.
3. No strict document size limitations.

- **RAG Components**:

1. **Document Loaders** ✅ (focus of these notes)
2. Text Splitters
3. Vector Databases
4. Retrievers

---

2. Document Loader: Main Concept

- **Purpose**: Load raw content (text, PDFs, webpages, CSVs, etc.) into a **standard document object** that LangChain can process.
- **Why Needed**: Different file types need different strategies to extract text.
- **Common Loaders**:
    - TextLoader
    - PyPDFLoader
    - WebBaseLoader
    - CSVLoader

---

3. Types of Document Loaders

🔹 TextLoader

- **What it does**: Loads plain .txt files.
- **Use Case**: Chat logs, transcripts, code snippets, or raw text files.
- **Limitation**: Works only with .txt format.
- **Example (no-code)**: Imagine you have a diary saved as a .txt file → TextLoader reads it and turns it into chunks for analysis.

🔹 PyPDFLoader

- **What it does**: Loads PDFs, converts each page into a document object.
- **Use Case**: Clean, text-based PDFs (like reports, e-books).
- **Limitation**: Not great with scanned PDFs or tables.
- **Example (no-code)**: You upload a company report PDF → PyPDFLoader splits it page by page for retrieval.

📊 **Other PDF Loaders for Special Needs**:

- **PDFPlumberLoader** → Extracts tables/columns properly.
- **UnstructuredPDFLoader / AmazonTexttractPDFLoader** → Works on scanned/image PDFs.
- **PyMuPDFLoader** → Preserves layout and images.
- **UnstructuredPDFLoader** → Best when structure (headings, lists) must be preserved.

🔹 DirectoryLoader

- **What it does**: Loads **multiple documents** from a folder at once.
- **Use Case**: If you have a folder with hundreds of files (e.g., data/reports/).
- **Glob Patterns**:

    - **/*.txt → all text files (recursive).
    - **/*.pdf → all PDFs in a folder.
    - data/*.csv → all CSV files in data/.
- **Example (no-code)**: Think of a folder "Research Papers" with 100 PDFs → DirectoryLoader loads them all in one go.

◆ Load vs Lazy Load

- **load()** (Eager Loading):
    - Loads all documents into memory at once.
    - Best when: Few documents + you want everything upfront.
    - Analogy: Carrying all your groceries in one trip.
- **lazy_load()** (Lazy Loading):
    - Loads documents one by one, only when needed.
    - Best when: Large number of files or very big PDFs.
    - Analogy: Ordering groceries one by one when you need them.

◆ WebBaseLoader

- **What it does**: Loads text from webpages (using BeautifulSoup).
- **Use Case**: Blogs, news articles, or static websites.
- **Limitation**:
    - Doesn’t handle JavaScript-heavy sites (need SeleniumURLLoader).
    - Only extracts static HTML text (not dynamic content).
- **Example (no-code)**: You want to analyze news from a blog → WebBaseLoader fetches and cleans the text.

◆ CSVLoader

- **What it does**: Loads .csv files (tables).
- **Use Case**: Customer records, product catalogs, survey data.
- **Limitation**: Works best when data is well-structured in columns.
- **Example (no-code)**: You upload a customer purchase history CSV → CSVLoader converts each row into a retrievable document.

---

4. Other Document Loaders

- **JSONLoader** → Loads JSON files (e.g., API responses, structured logs).
- **EmailLoader** → Extracts text from email files.
- **UnstructuredFileLoader** → General-purpose loader for mixed formats.
- **S3/Cloud Loaders** → Load directly from cloud storage (AWS S3, GDrive, etc.).

---

5. Making a Custom Document Loader (No-Code Explanation)

Sometimes your data doesn't fit neatly into .txt, .pdf, or .csv. In that case, you can **make your own loader**.

How?

1. **Identify Data Source**: e.g., WhatsApp chat export, custom app logs, or voice transcripts.
2. **Define Extraction Rules**: Decide how to break that raw data into text chunks.
   - Example: For WhatsApp, split by each message.
3. **Wrap into Document Objects**: Store each chunk as a document with metadata (like date, sender, source).

Example (no coding):

- Imagine you run a call center. You export customer complaints as .xml files.
- No existing loader reads .xml.
- You write rules like:
  - Extract <customer_name> as metadata.

    - Extract <complaint_text> as document content.

- Now each complaint is a document ready for RAG.

✅ **In summary**:

- **Document Loaders** are the first step in RAG pipelines.
- They convert raw files (text, PDF, web, CSV, etc.) into structured documents.
- Multiple loaders exist depending on file type.
- For huge datasets, choose between load() and lazy_load().
- You can even build **custom loaders** for special formats (e.g., XML, chat logs, voice transcripts).