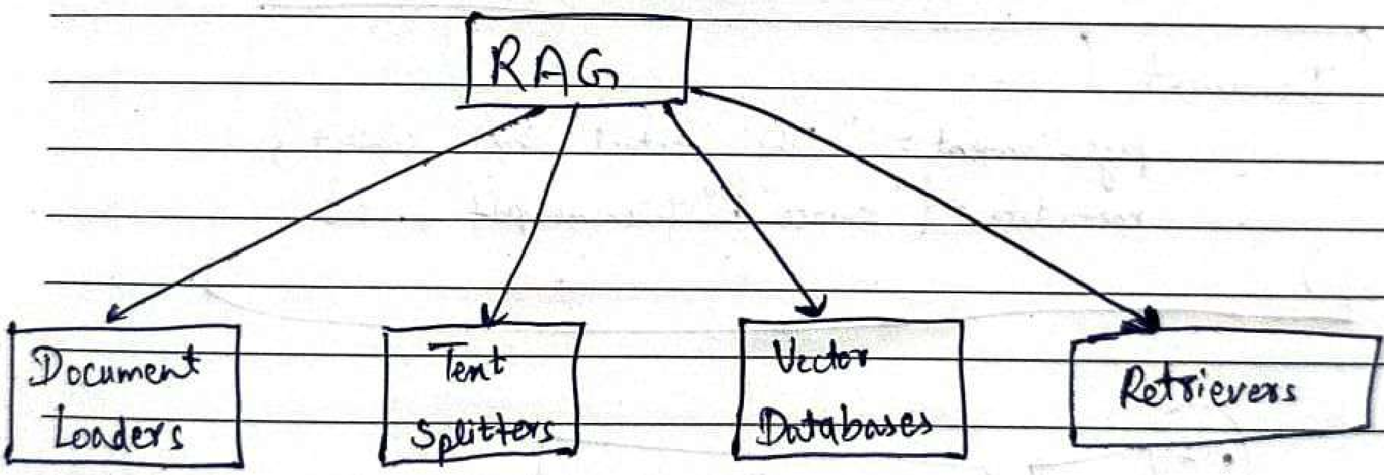
RAG

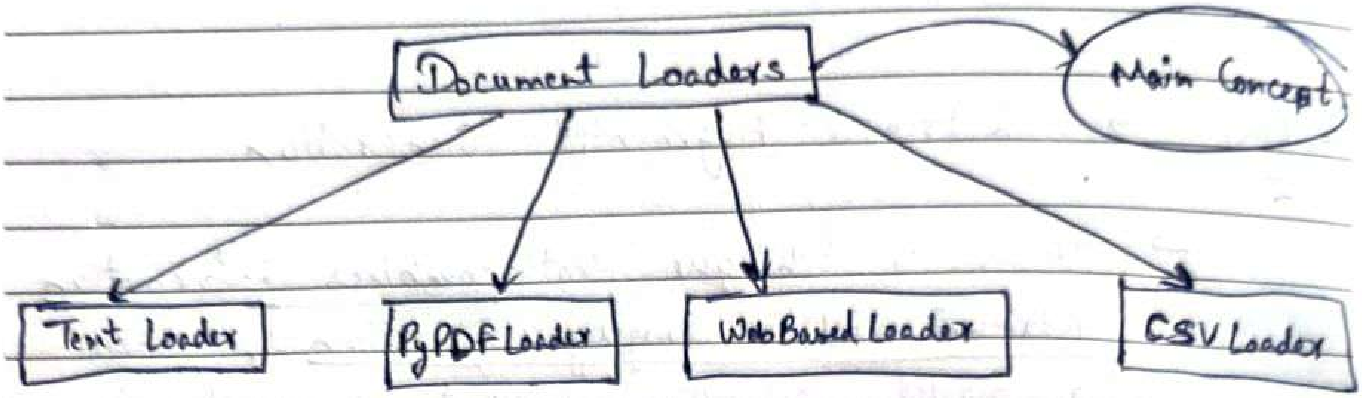
RAG → Retrieval - Augmented Generation

→ It is a technique that combines information retrieval with language generation, where a model retrieves relevant documents from a knowledge base and then uses them as context to generate accurate & grounded responses.

Benefits → ① Use of up to date info.
② Better privacy.
③ No limit on document size.



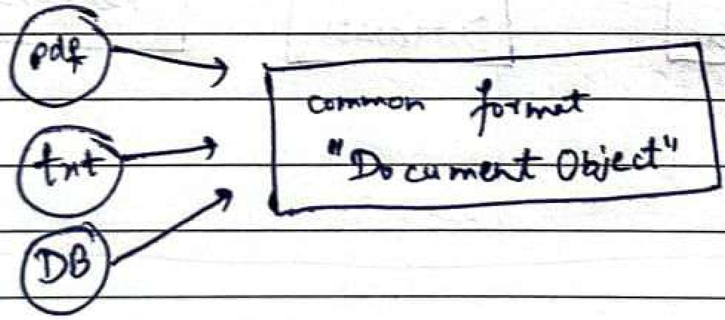
A. Document Loaders



→ Document loaders are components in LangChain used to load data from various sources into a standardized format (usually as Document objects), which can be then used for chunking, embedding, retrieval and generation.

The Standard format is Document object :-

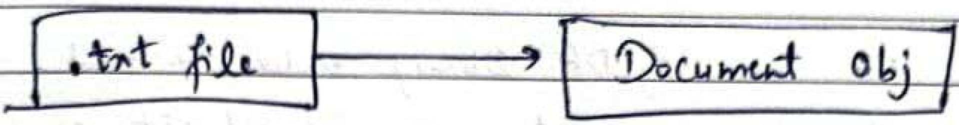
```
Document (
        page-content = " The actual text content",
        metadata = {" source" : " filename.pdf", ... }
)
```



① <u>Text Loader</u> → simple and commonly used doc. loader that reads plain text files and convert them into Document Objects.

- Use → for loading chat logs, scraped text, transcripts, code snippets, or any plane text data into langchain pipeline.

- Limitations → works only with .txt files.



```
loader = TextLoader("poem.txt")
docs = loader.load()
```

→ The docs is a 'list' of 'document objects'.

docs[0] → is a document obj.

docs[0].page-content → is the content.

② <u>PyPDF Loader</u> → a document loader in langchain used to load content from PDF files and convert ~~them~~ each page into a document obj.

```
[
Document(page_content=" .... ", metadata={"page":0,"source":"file"}),
Document(page_content="....", metadata={"page":0,"source:"file"})
]
```

<u>Limitations:</u> It uses PyPDF library under the hood. Hence not great with scanned PDF or complex layouts.

<u>More PDF loaders:-</u>

Use Case	Loader
Simple Clean PDF	PyPDF Loader
PDFs with tables/columns	PDFPlumber Loader
Scanned / Image PDF	UnstructuredPDFLoader (or) AmazonTextractPDFLoader
Need layout & image data	PyMuPDFLoader
Want best structure extraction	UnstructuredPDFLoader

⑤ Directory Loader → a document loader that lets you to load multiple documents from a directory/folder of files.

```
loader = DirectoryLoader (
                path = "./SampleDir",
                glob = "*.pdf",
                loader_cls = "PyPDFLoader
          )
```

glob pattern	What it loads
"**/*.txt"	All '.txt' files in all subfolders
"*.pdf"	All ".pdf" files in root folder
"data/*.csv"	All ".csv" files in "data/" folder
"**/*"	All file types in all folders

Note → • loading a lot of document at once is very slow operation.

Therefore we will use 'Lazy Loading'

→ Load v/s Lazy Load →

load()	lazy_load()
Eager loading (loads everything at once)	lazy loading (loads on demand)
Returns : a list of documents objects.	Returns: a generator of document objects.
Loads all documents immediately into the memory.	Documents are not loaded all at once; they're fetched one at a time as needed.
Best when:- • The no. of documents is small. • You want everything loaded upfront	Best when:- • You are dealing with large documents or lots of files. • You want to stream processing (eg. chunking, embedding) without using lots of memory.

- Create your own document loader at:-

python.langchain.com/docs/how_to/document_loader_custom/

④ WebBaseLoader → a document loader ~~to~~ in LangChain used to load & extract text content from web pages (URLs).

It uses BeautifulSoup under the hood to parse HTML and extract visible text.

When to use :- • for blogs, news articles, or public websites where the content is primarily text-based and static.

Limitations:- • Doesn't handle JavaScript heavy pages well (Use 'SeleniumURLLoader' for that).
• Loads only static content (what's in the HTML, not that loads after the page render).

⑤ CSVLoader → used to load CSV files into langChain Document objects - one per row, by default.

⇒ More loaders can be found at :-

python.langchain.com/docs/integrations/document_loaders/