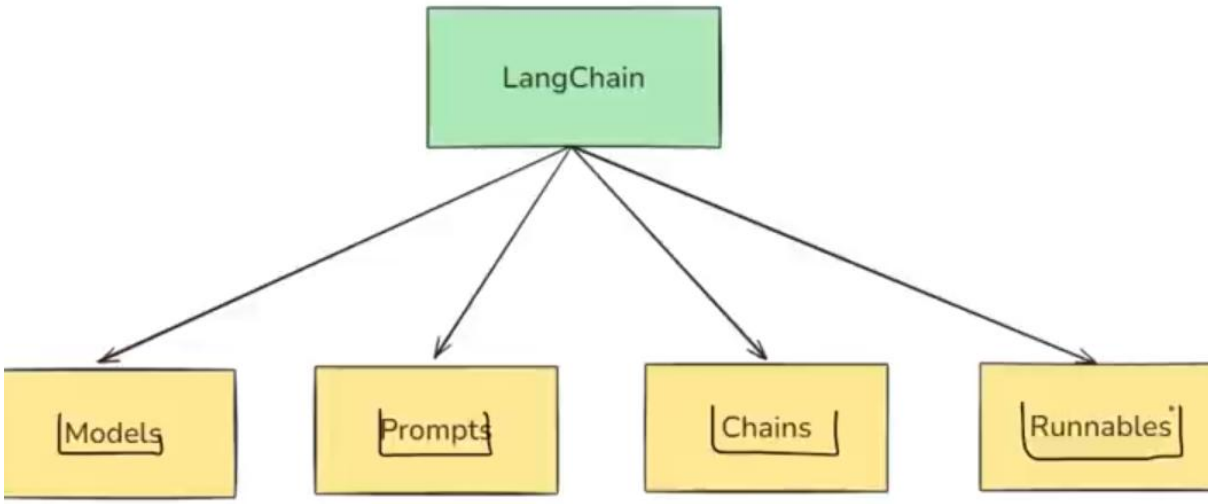


RAG is a technique that combines information retrieval with language generation, where a model retrieves relevant documents from a knowledge base and then uses them as context to generate accurate and grounded responses.

Benefits of using RAG

1. Use of up-to-date information
2. Better privacy
3. No limit of document size

Chatbots → ChatGPT

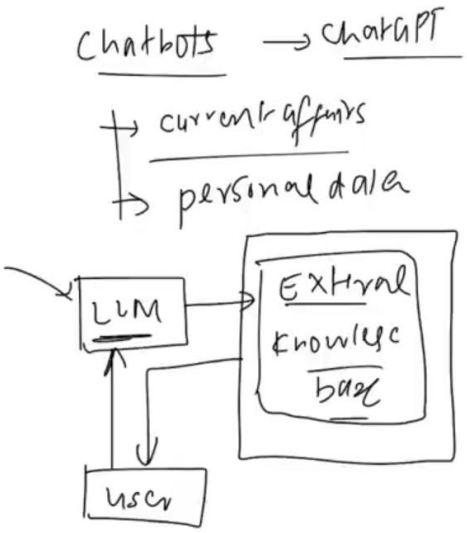
→ current affairs

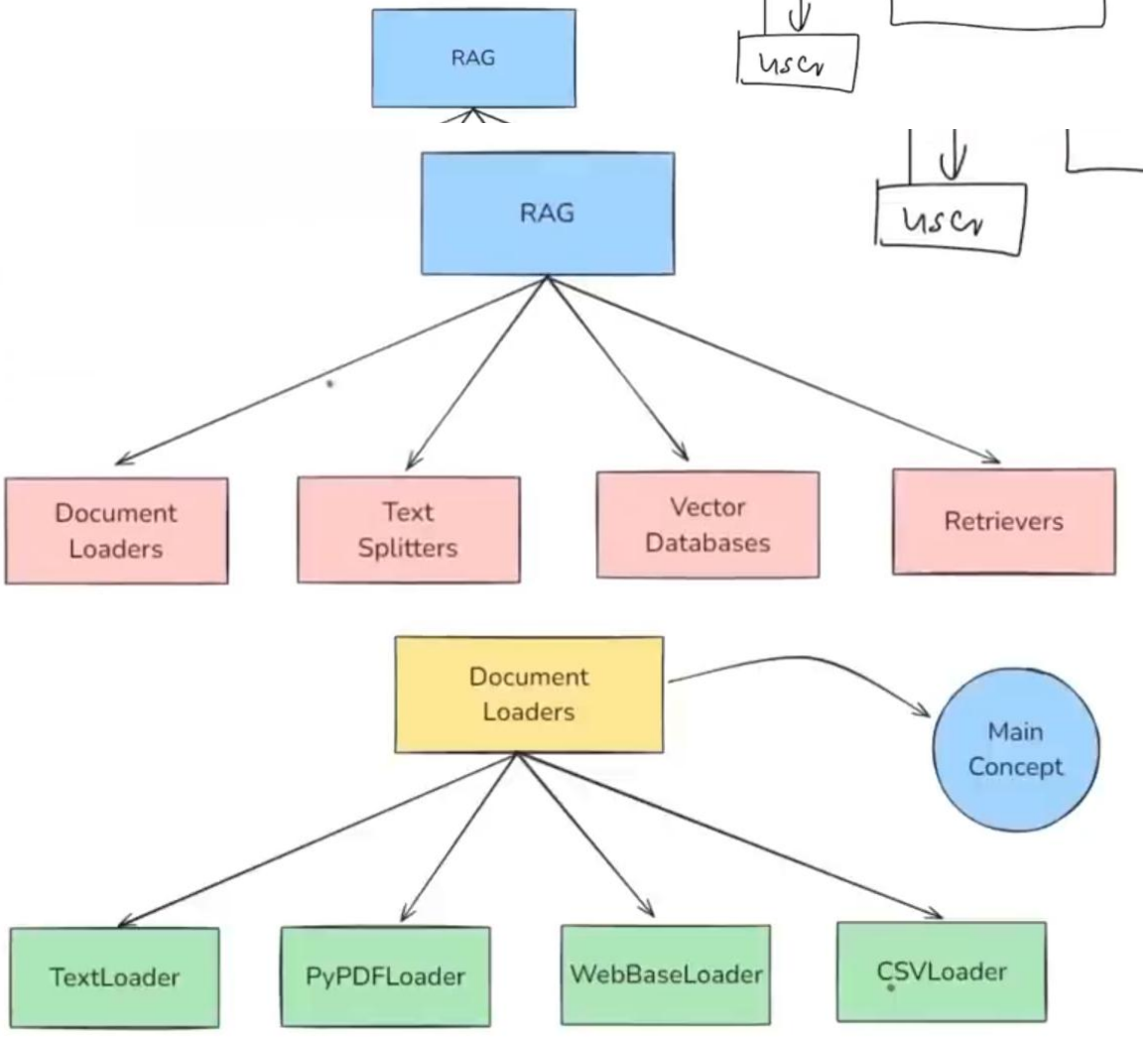
→ personal data

RAG 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 and grounded responses.

Benefits of using RAG

1. Use of up-to-date information
2. Better privacy
3. No limit of document size

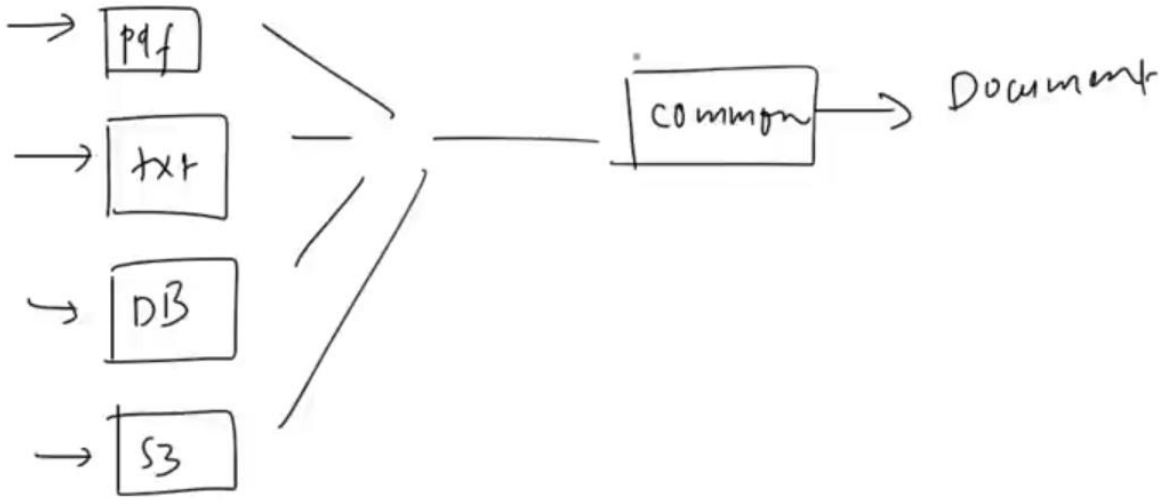




Document Loaders in LangChain

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

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



TextLoader

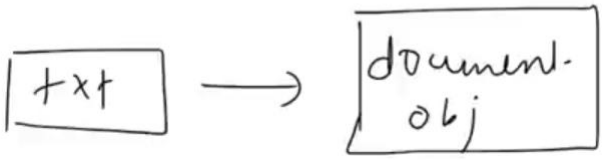
TextLoader is a simple and commonly used document loader in LangChain that reads plain text (.txt) files and converts them into LangChain Document objects.

Use Case

- Ideal for loading chat logs, scraped text, transcripts, code snippets, or any plain text data into a LangChain pipeline.

Limitation

- Works only with .txt files



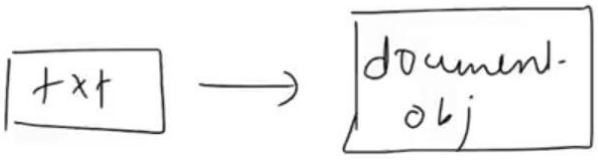
TextLoader is a simple and commonly used document loader in LangChain that reads plain text (.txt) files and converts them into LangChain Document objects.

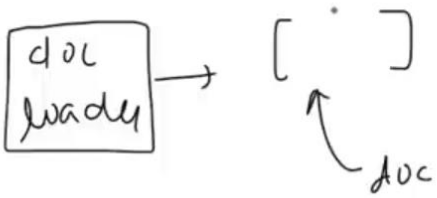
Use Case

- Ideal for loading chat logs, scraped text, transcripts, code snippets, or any plain text data into a LangChain pipeline.

Limitation

- Works only with .txt files





PyPDFLoader is a document loader in LangChain used to load content from PDF files and convert each page into a Document object.

```
[
    Document(page_content="Text from page 1", metadata={"page": 0, "source": "file.pdf"}),
    Document(page_content="Text from page 2", metadata={"page": 1, "source": "file.pdf"}),
    ...
]
```

Limitations:

- It uses the **PyPDF** library under the hood — not great with scanned PDFs or complex layouts.

DirectoryLoader

27 March 2025 18:44

DirectoryLoader is a document loader that lets you load multiple documents from a directory (folder) of files.

Glob Pattern	What It Loads
`"**/*.txt"`	All `.txt` files in all subfolders
`"*.pdf"`	All `.pdf` files in the root directory
`"data/*.csv"`	All `.csv` files in the `data/` folder
`"**/*"`	All files (any type, all folders)

`**` = recursive search through subfolders

Load vs Lazy load

27 March 2025 23:51

✅ `load()`

- **Eager Loading** (loads everything at once).
- Returns: A **list of** `Document` **objects**.
- Loads all documents **immediately** into memory.
- Best when:
  - The number of documents is small.
  - You want everything loaded upfront.

🌀 `lazy_load()`

- **Lazy Loading** (loads on demand).
- Returns: A **generator** of `Document` objects.
- Documents are **not all loaded at once**; they're fetched one at a time as needed.
- Best when:
  - You're dealing with **large documents or lots of files**.
  - You want to **stream** processing (e.g., chunking, embedding) without using lots of memory.

WebBaseLoader

28 March 2025 00:34

WebBaseLoader is a document loader in LangChain used to load and 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 what loads after the page renders).