1.Temperature parameter Meaning of Models

`temperature` is a parameter that **controls the randomness of a language model's output**. It affects how **creative or deterministic** the responses are.

- **Lower values** (`0.0 - 0.3`) → More **deterministic** and predictable.
- **Higher values** (`0.7 - 1.5`) → More **random**, creative, and diverse.

Use Case	Recommended Temperature
Factual answers (math, code, facts)	`0.0 - 0.3`
Balanced response (general QA, explanations)	`0.5 - 0.7`
Creative writing, storytelling, jokes	`0.9 - 1.2`
Maximum randomness (wild ideas, brainstorming)	`1.5+`

Temperature =0, will gives you the same output for the same input as many time you will print, but if you increase the value of Temperature, than you will see that for the same input many time, it will print different outputs and in more creative way.

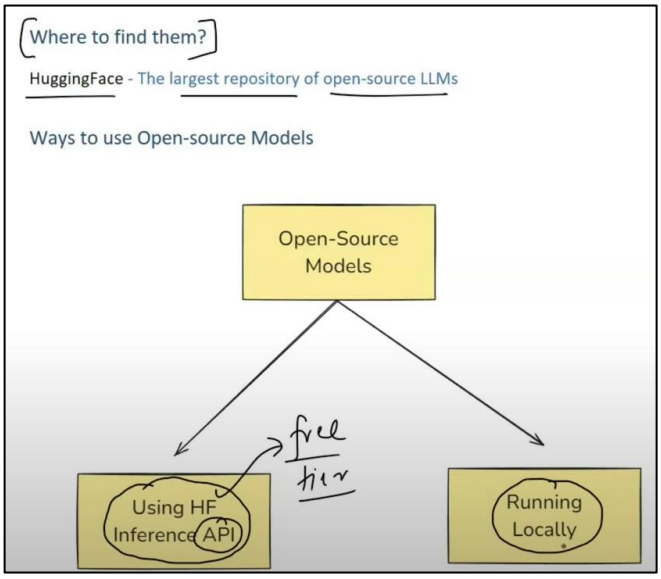
2. Tokens roughly means words i.e. (max_completion_tokens=10) roughly means 10 words.

3. Open Source Models

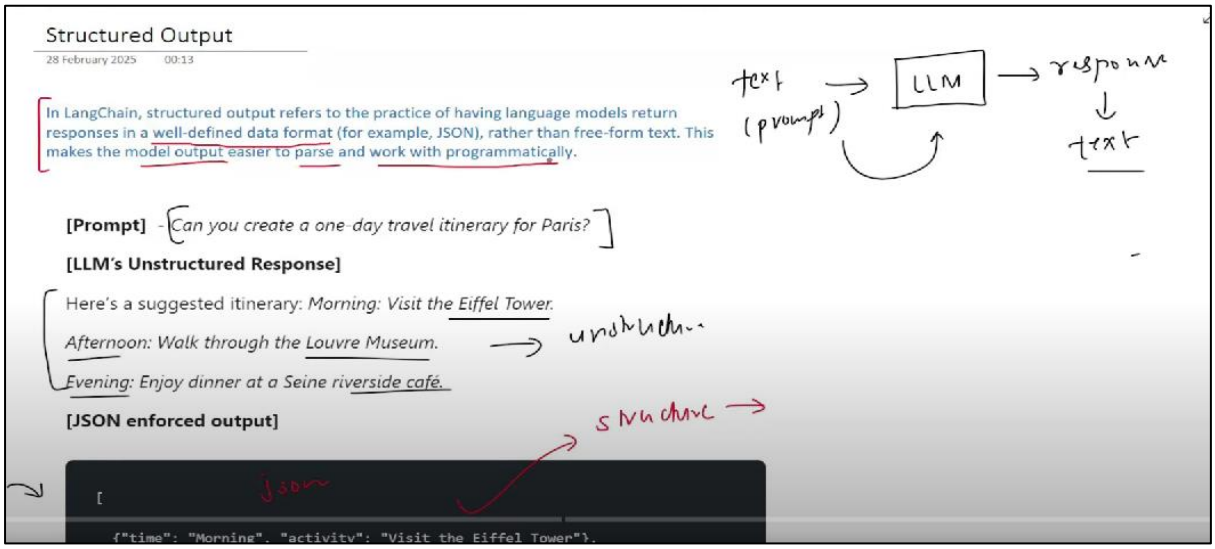
Some Famous Open Source Models

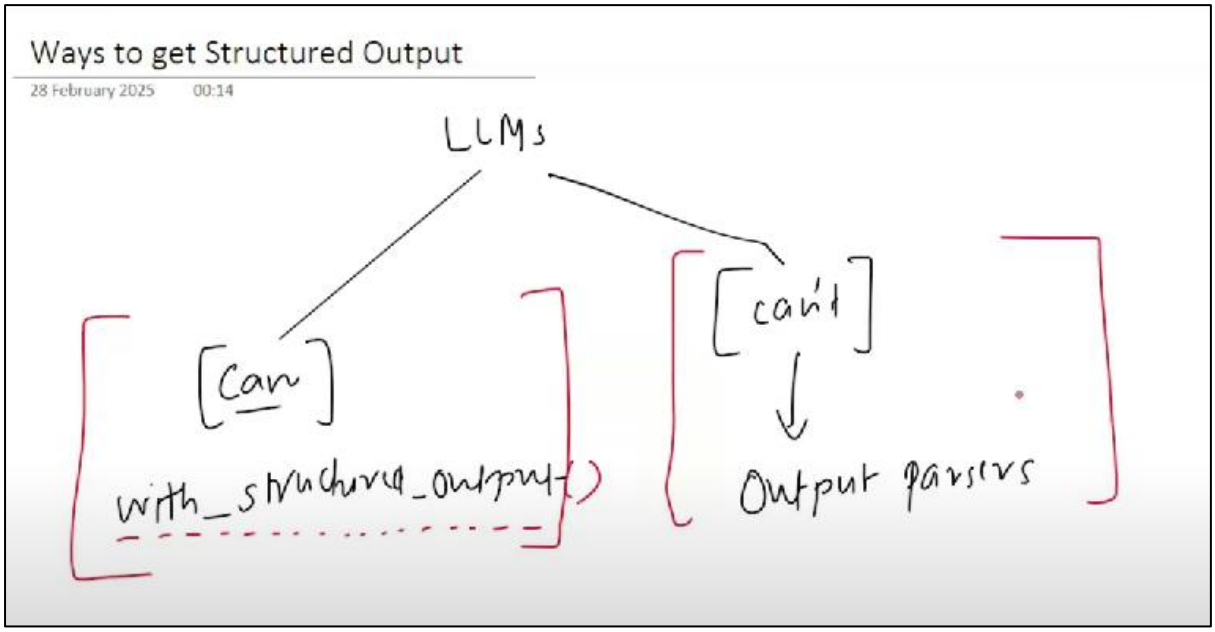
Model	Developer	Parameters	Best Use Case
LLaMA-2-7B/13B/70B	Meta AI	7B - 70B	General-purpose text generation
Mixtral-8x7B	Mistral AI	8x7B (MoE)	Efficient & fast responses
Mistral-7B	Mistral AI	7B	Best small-scale model (outperforms LLaMA-2-13B)
Falcon-7B/40B	TII UAE	7B - 40B	High-speed inference
BLOOM-176B	BigScience	176B	Multilingual text generation
GPT-J-6B	EleutherAI	6B	Lightweight and efficient
GPT-NeoX-20B	EleutherAI	20B	Large-scale applications
StableLM	Stability AI	3B - 7B	Compact models for chatbots

4. Where to find the Open Source Models

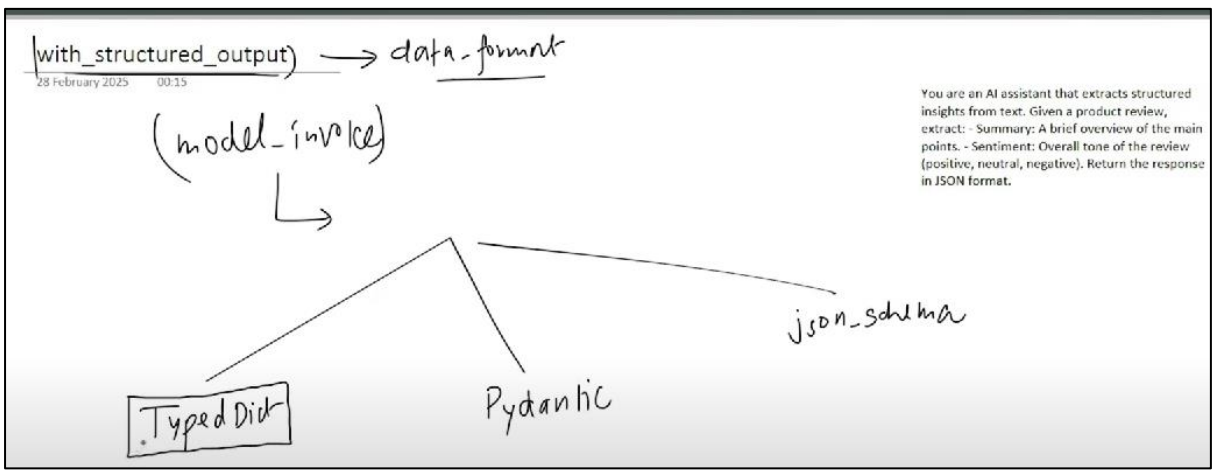


5. Structured Output and Unstructured Output



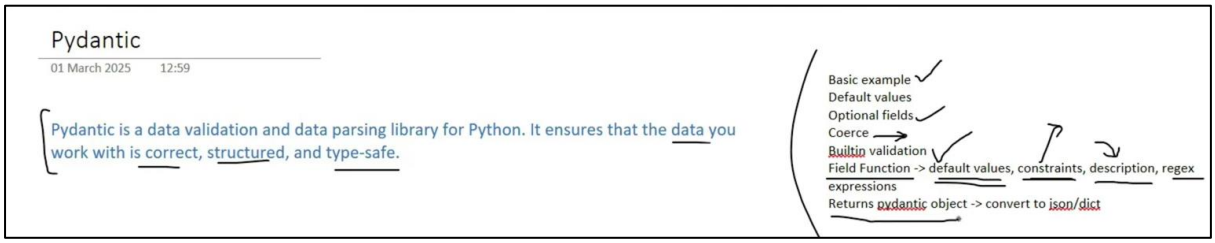


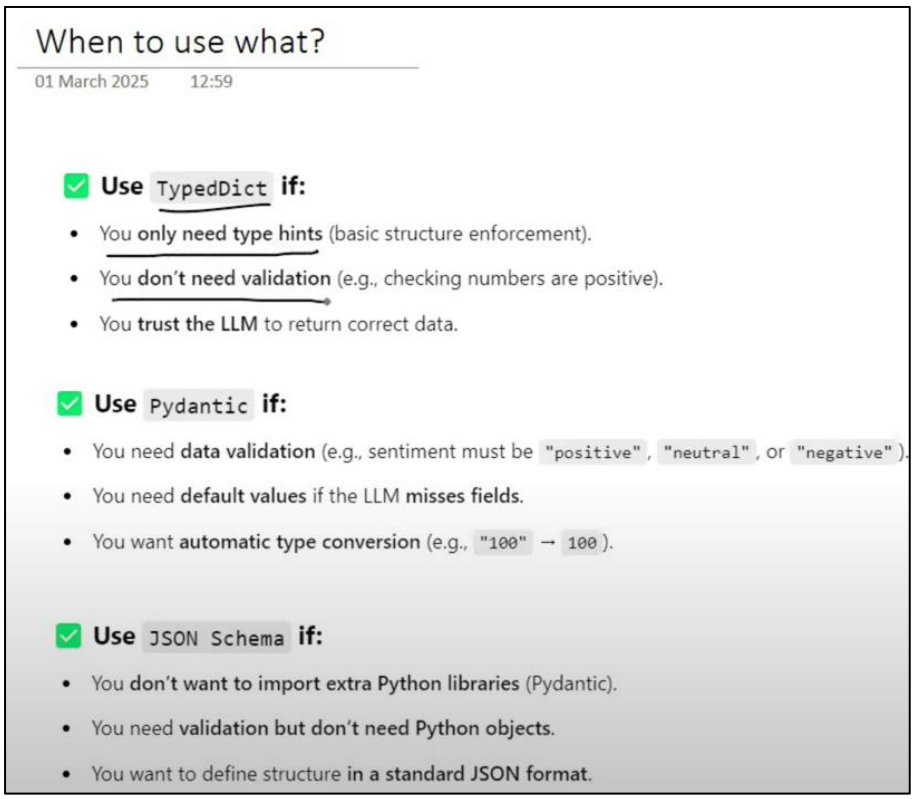
Those LLM's which gives the structured output, for that we use with_structured_output fxn for extracting the output of LLM in structured form, and those LLM which are not able to produce the structured output we use Output parsers for making there output as structured output.

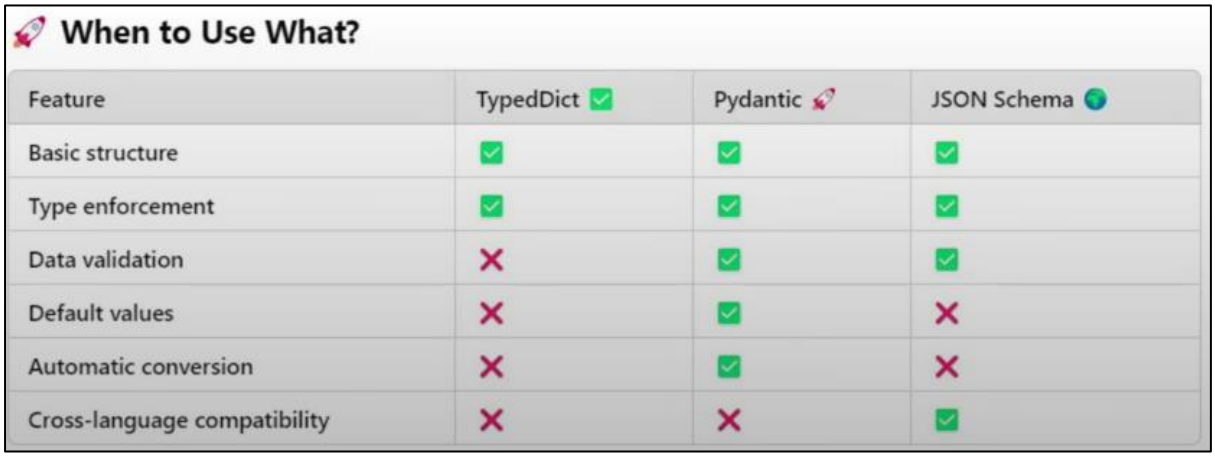


See jo with_structure_output fxn hai voh simply hmare output ko eek particular structure form mei de rha hai, aur jis particular structure me hume output chaiye, voh structure hum upar ke teen mei se kisi bhi eek way sei define kr skte hai (TypedDict, Pydantic and json_schema),

Phele hum input dete thae llm ko voh hume output deta tha text mei (text eek unstructured data format hai) aab ussi output ko eek proper structure mei lane ke liye aab model ko batayenge(TypedDict) ki help se ki mujhe kis particular structure format mei output chaiye, got it!!



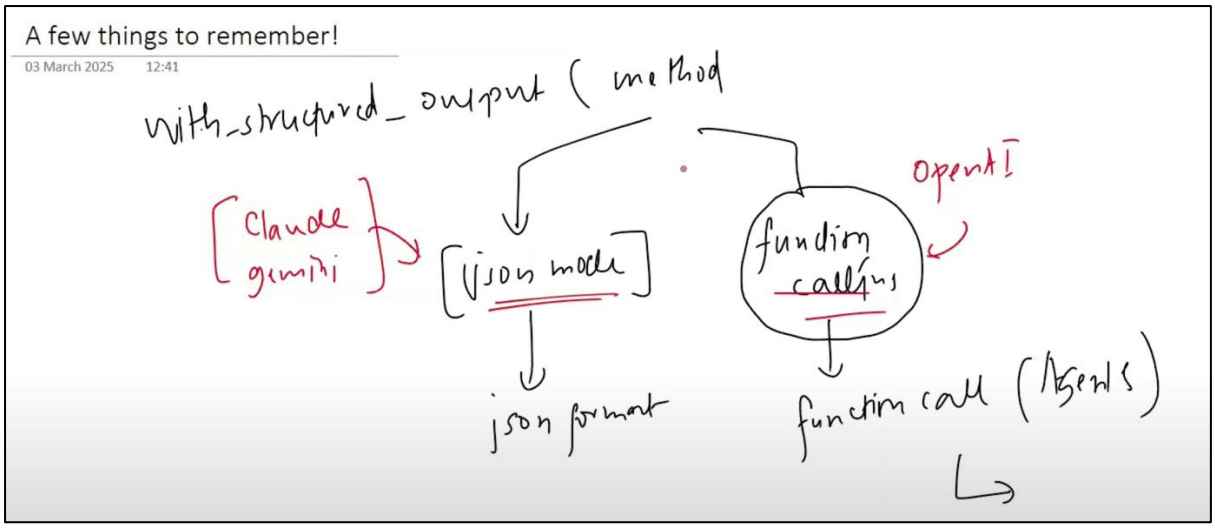




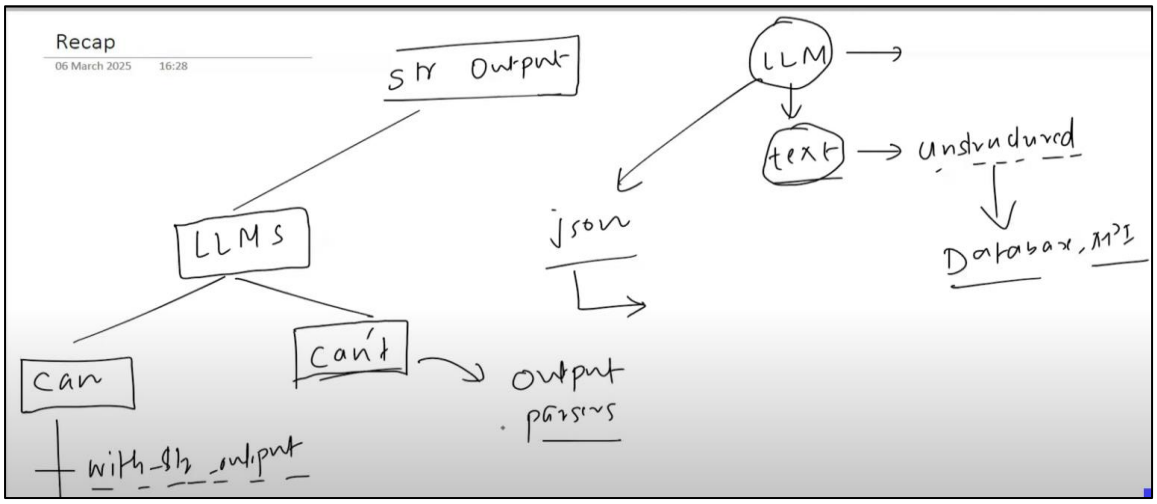
Kabhi kabhi hum Hugging face ke open source models ko use krte hai structure output ke liye, lekin most of the open-source model structure output support nhi krte,

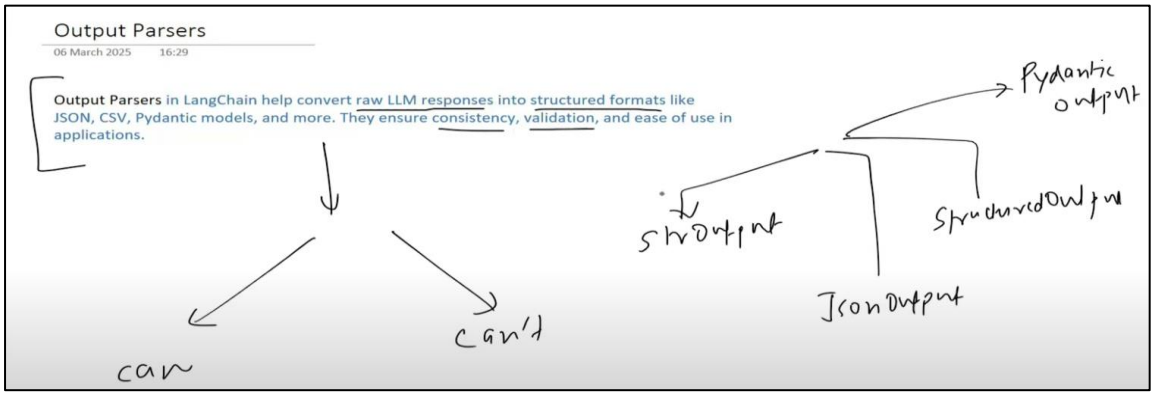
OpenAI support krta hai structure output, Deepseek thoda thoda, llama nhi krta support

With_structure_output() mei eek attribute hota hai jiski doo values hoti hai one is json mode and another is function calling, inka actual kam toh nhi pta lekin default value fxn calling hota hai jo hum openAI ke model ke saath use kr skte hai



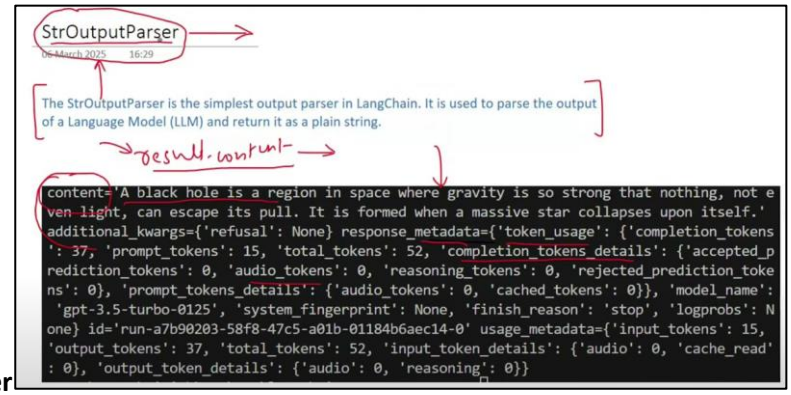
OUTPUT PARSER: are those method with which we can convert the unstructured data into structured data.



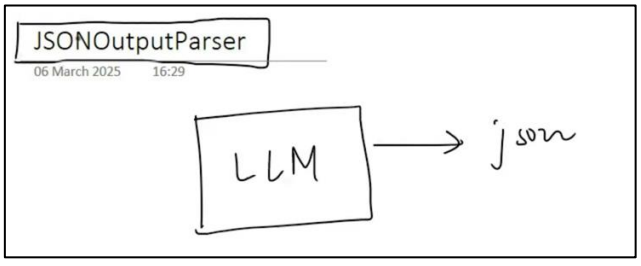


Output parsers can be used both kinds of LLM models which can and can't produce the Structured Output, and four famous output parsers are:

1. **Str(string) Output parser**

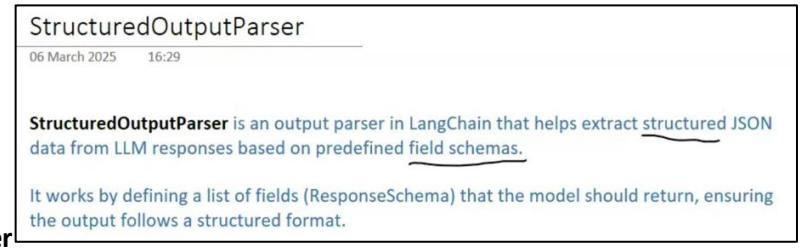


2. **Json output parser**



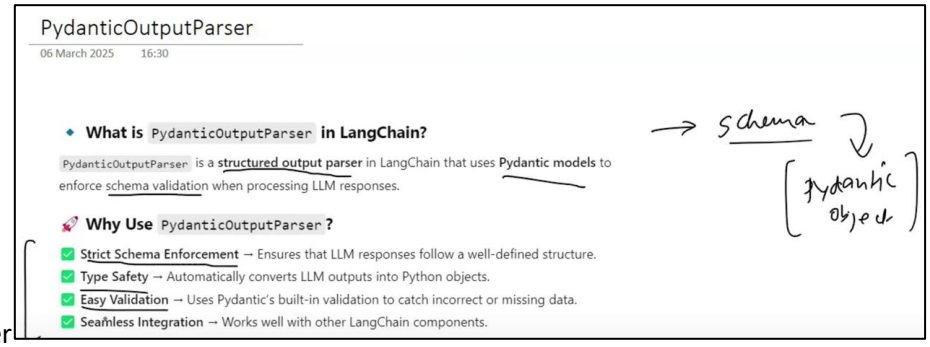
esa parser jo LLM ke output ko JSON format mei de is our JSON output parser. Limitation of JSON output

parser is that ki jo hume JSON output format me milega, voh JSON format fix nhi hota hai aur uss ko control nhi kr skte, and issi cheez ko solve krne ke liye STRUCTURED output parser hai.



3. **Structured output parser**
   this parser forces the LLM to produce the Structured output which is predefined to it (i.e. ResponseSchema). And the main disadvantage is that we cannot do the DATA VALIDATION thing in output like if we want the 'age' value in integer, but if our LLM is giving it in 'str' form, so we cannot control that thing.

Pydantict output parser



Runnables:

In **LangChain**, a **Runnable** is a core abstraction that represents a **composable unit of computation** — something that can be executed. This could be a language model call, a prompt template, a chain, a retriever, or any other operation that can be "run." In simple term runnables vo hota hai jo chains form krne mei use hota hai, basically agr apke paas output hai aur aap chahte ho ki voh kiski aur fxn mei input jaye, toh toh uss fxn ko runnable bna do, than yeah output to input wala kaam boht easy ho jayega.

Think of a `Runnable` as:

Something that takes in input, does some processing, and returns output — possibly asynchronously or in a streaming fashion.

🧱 Why Runnables Matter in LangChain

Runnables are part of LangChain's new **Runnable protocol**, introduced in the **LangChain Expression Language (LCEL)**. This provides a standard interface for building and composing components.

This allows developers to **chain together** complex flows using simple, modular blocks.

✅ Core Features of Runnables

1. **Composable**:
   You can chain runnables together using `|`, e.g.:
2. `chain = prompt | llm | output_parser`
3. **Consistent Interface**:
   Every `Runnable` implements methods like:
   - `invoke()` – for synchronous execution
   - `stream()` – for streaming output
   - `batch()` – for batch processing
   - `ainvoke()` – for async execution
4. **Supports Streaming and Async**:
   Runnables are designed to work with both streaming outputs (like OpenAI's streaming API) and async processing, making them suitable for production use.

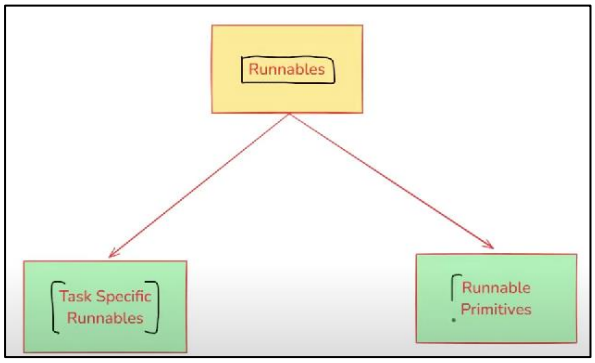
🎁 Examples of Built-in Runnables

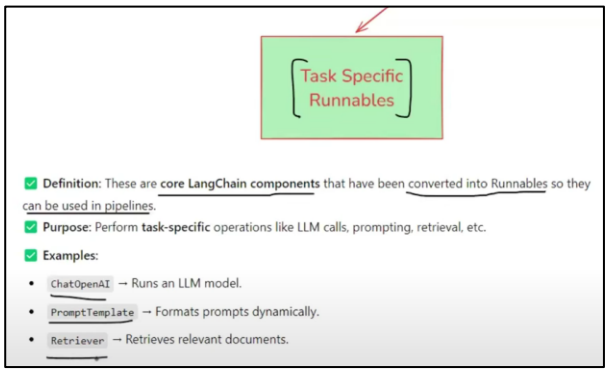
- `PromptTemplate` (turns input dict → prompt string)
- `ChatOpenAI` (prompt → LLM output)
- `RunnableMap` (parallel execution of multiple Runnables)

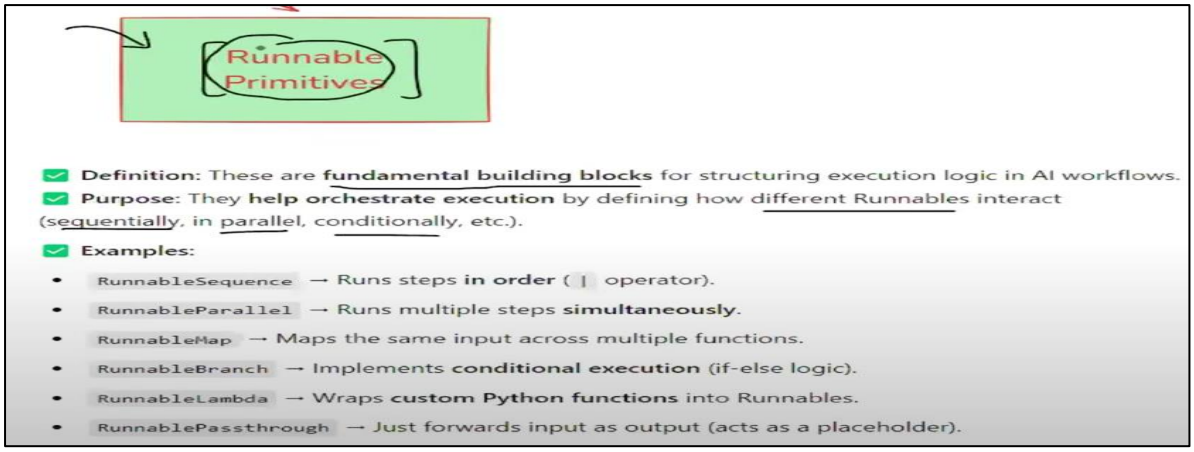
- `RunnableLambda` (wraps a custom Python function)

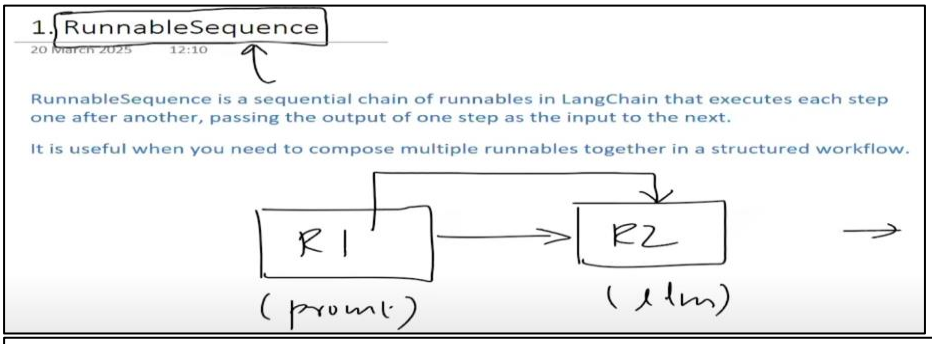
🧠 Summary

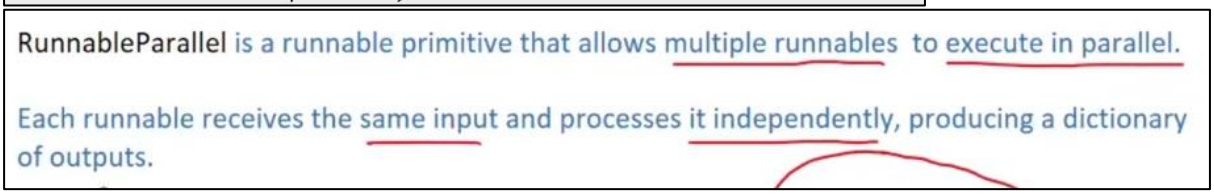
- `Runnable` is a **core concept** in LangChain's design to make LLM workflows modular, composable, and production-ready.
- It's like a standardized interface for any "step" in a processing pipeline.

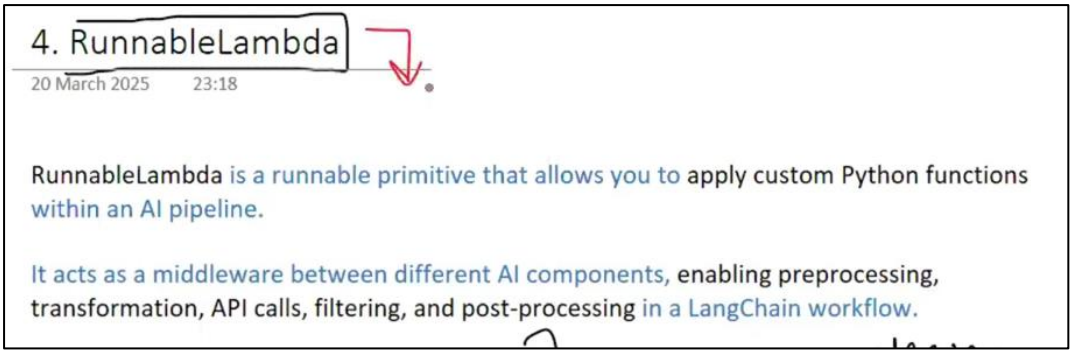






1. 

2. 

3. 

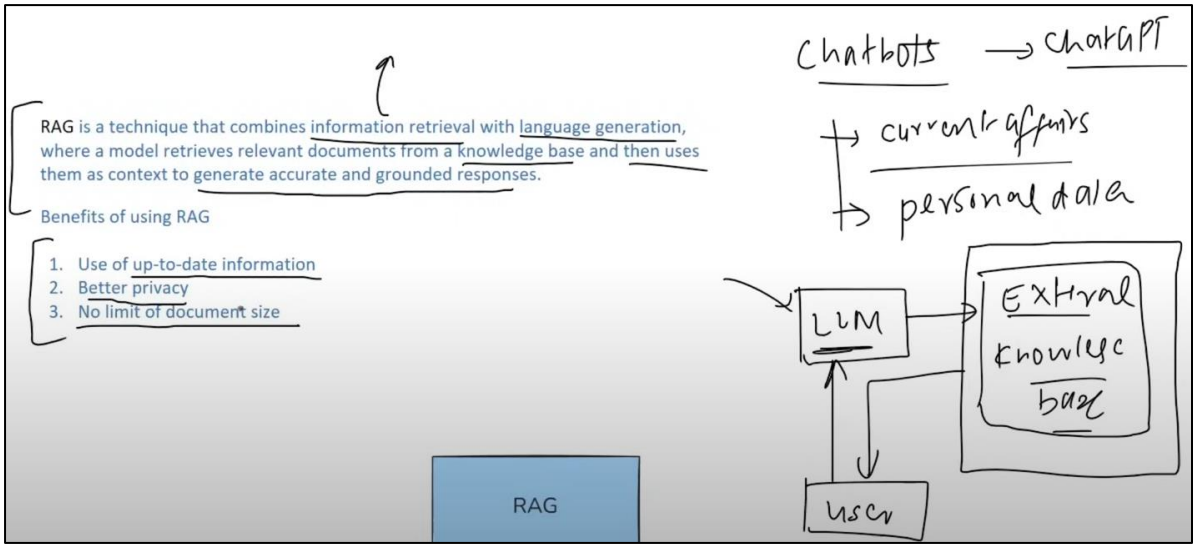
RunnableLambda ki help sei hum python ke kisi bhi simple function ko runnable me convert kr skte hai.

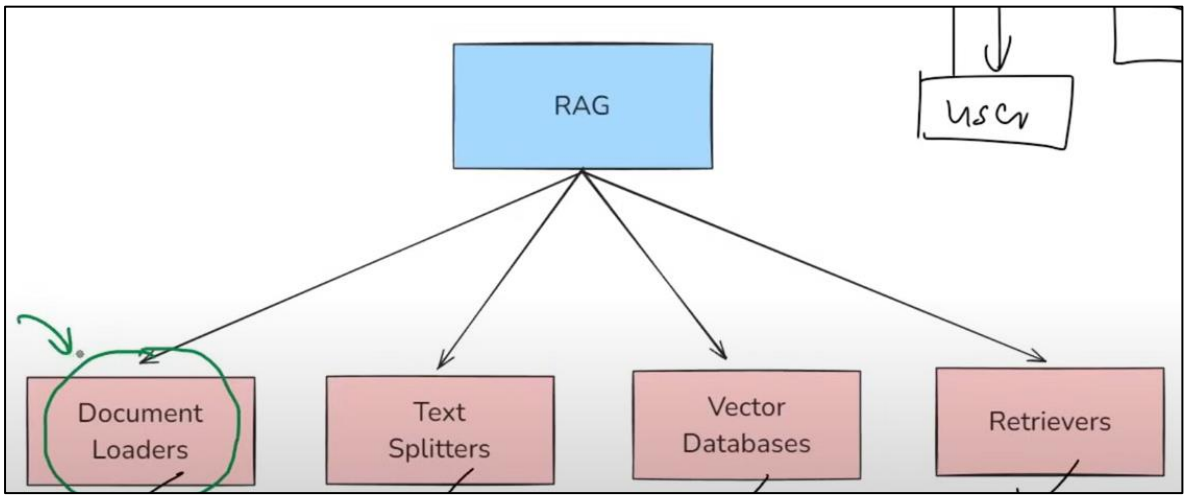
RAG: stands for **Retrieval-Augmented Generation**. It's a technique that combines two key components:

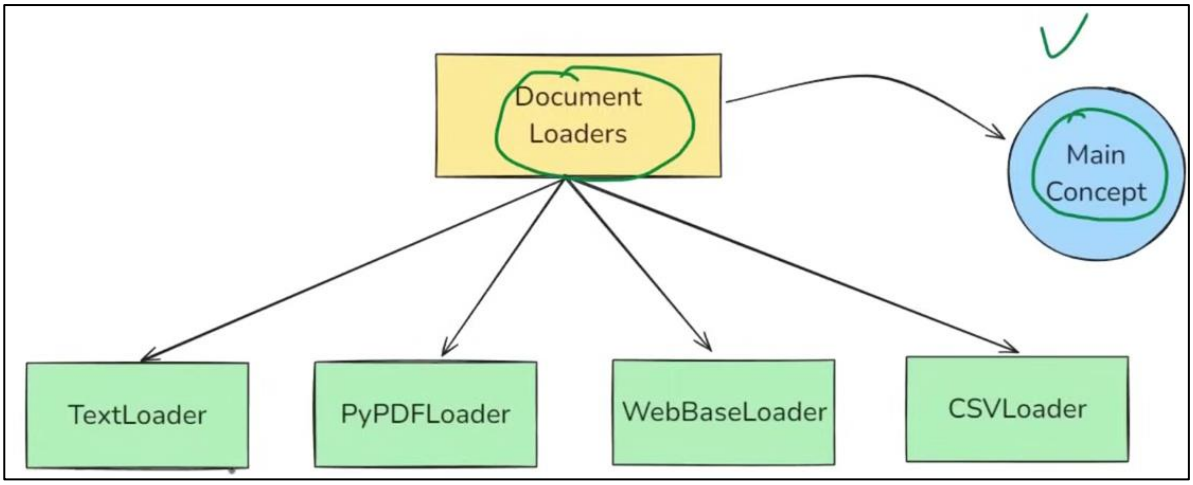
1. **Retrieval** – Pulling in relevant information from a large external knowledge base (like documents, databases, or the web).
2. **Generation** – Using a language model to generate answers based on the retrieved information.

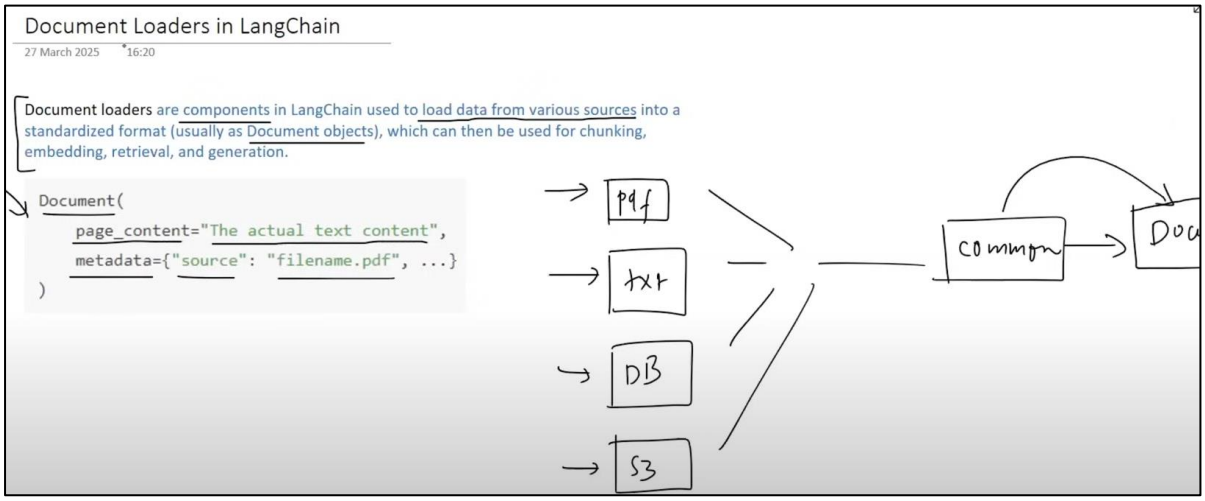
How It Works:

1. **Input Query**: A user asks a question or gives a prompt.
2. **Retriever Module**: Instead of relying only on the model's internal knowledge, the system searches an external corpus (e.g., Wikipedia or a private document store) for relevant documents.
3. **Generator Module**: The retrieved documents, along with the original query, are fed into a language model (like GPT) to generate a final, informed response.

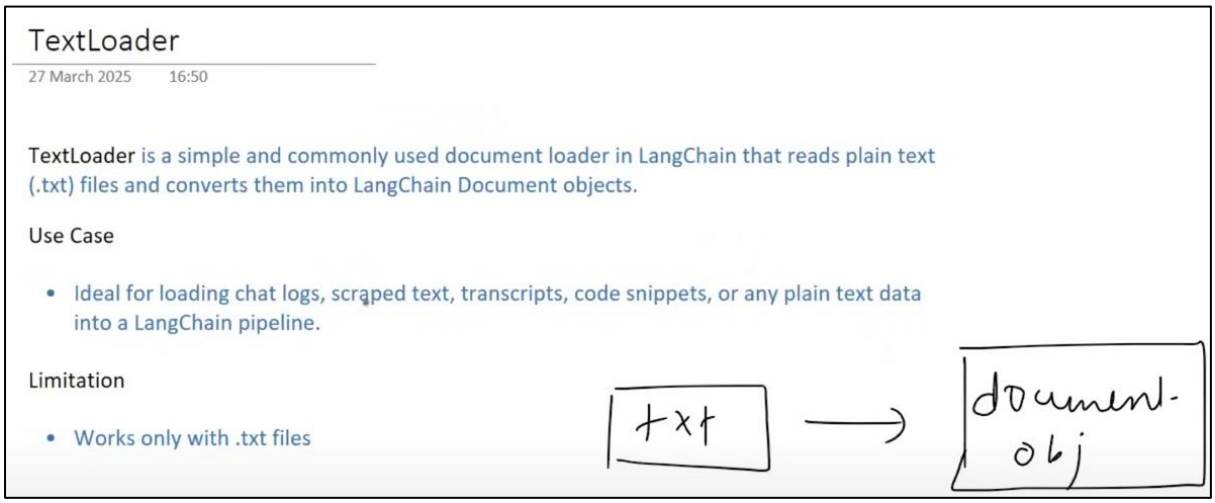








1. Text Loader

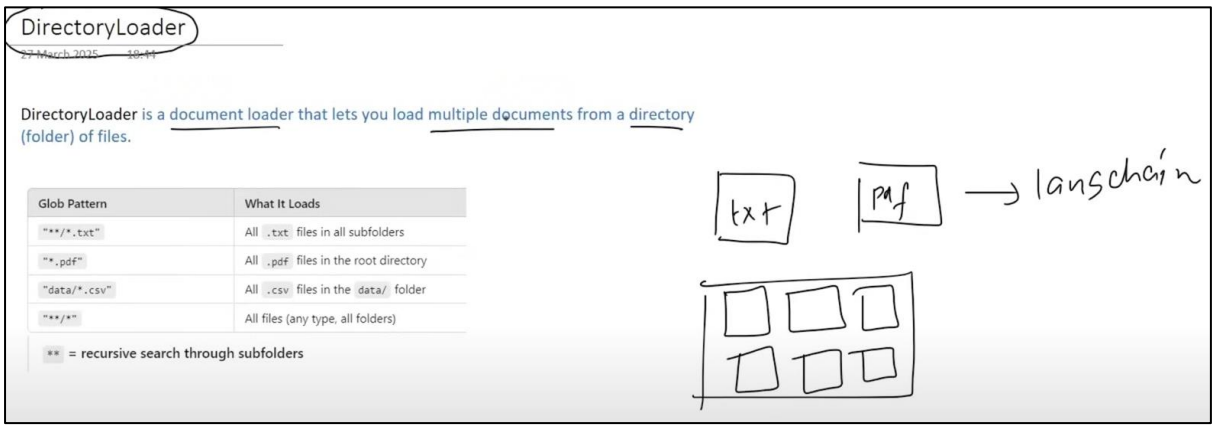


2. PyPDFLoader

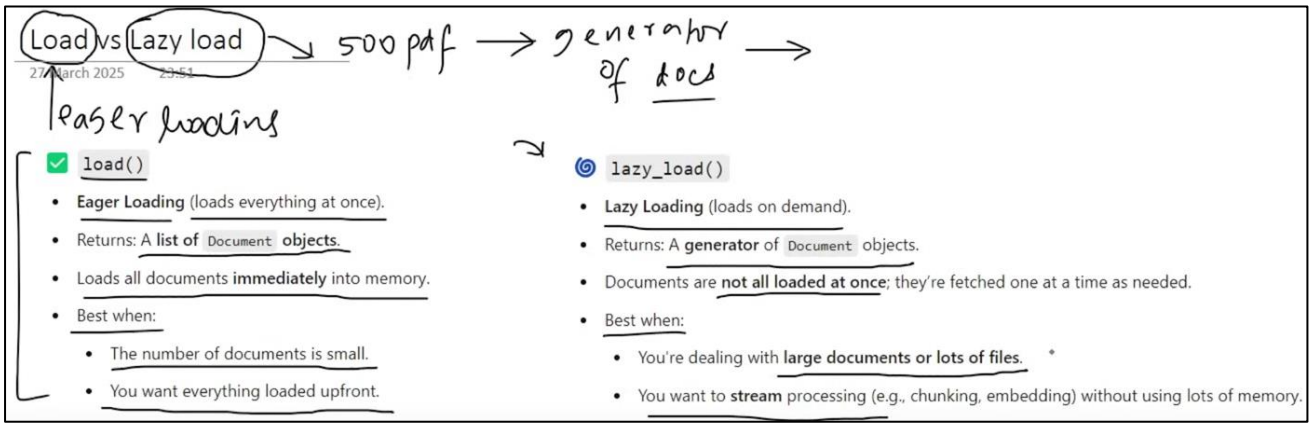
https://python.langchain.com/docs/integrations/document_loaders/#pdfs

Use Case	Recommended Loader
Simple, clean PDFs	`PyPDFLoader`
PDFs with tables/columns	`PDFPlumberLoader`
Scanned/image PDFs	`UnstructuredPDFLoader` or `AmazonTextractPDFLoader`
Need layout and image data	`PyMuPDFLoader`
Want best structure extraction	`UnstructuredPDFLoader`

3. Directory Loader

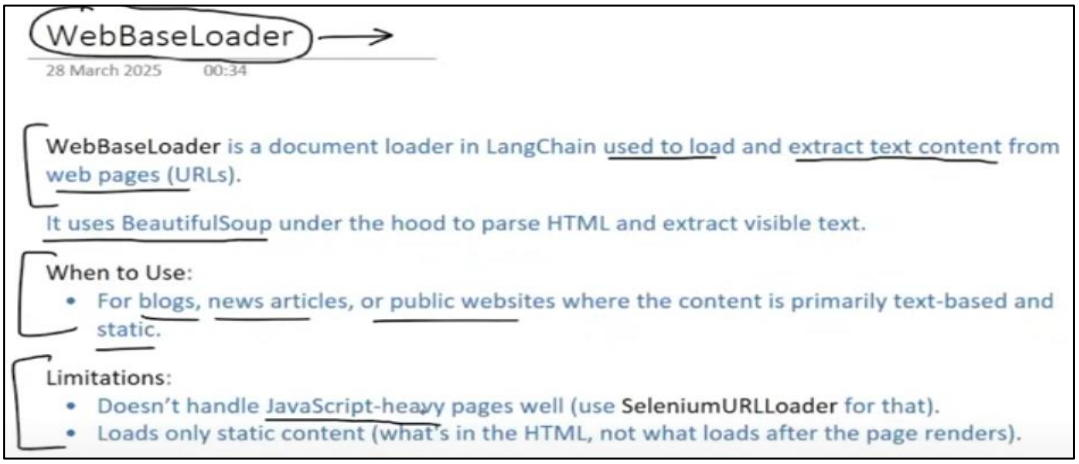


Load() vs lazy_load()

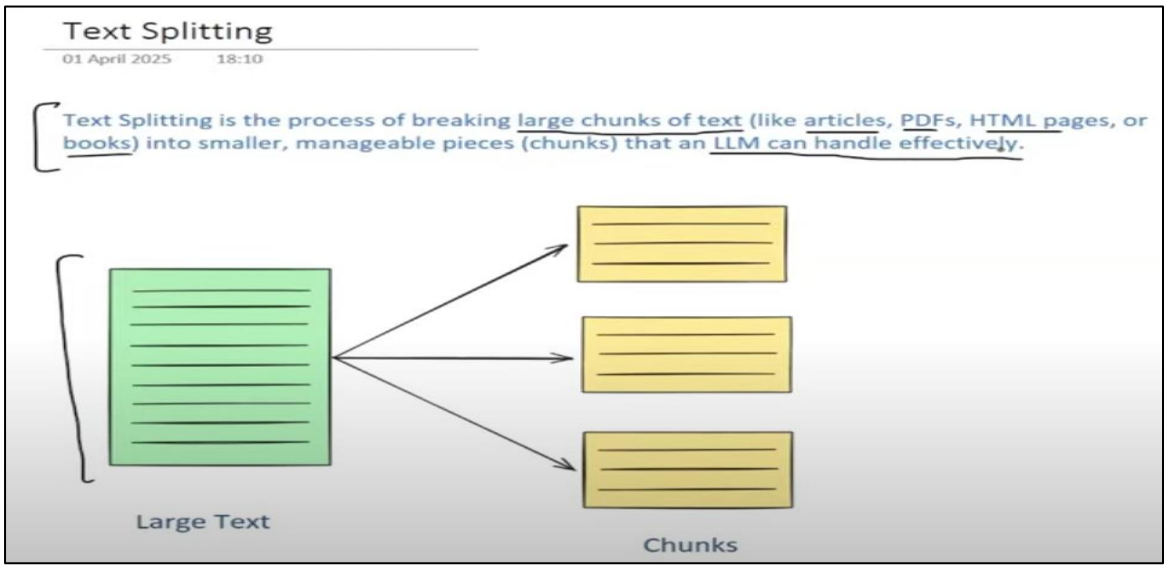


Load fxn saare pages ko eek saath load kr deta hai ram mei aur lazy load fxn one by one page ko load krta hai memory mei

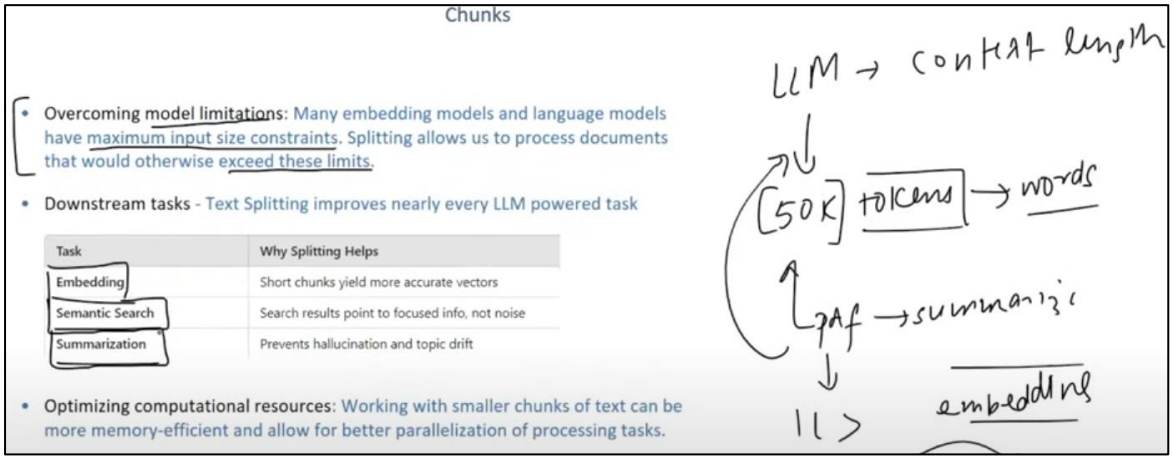
4. WebBaseLoader



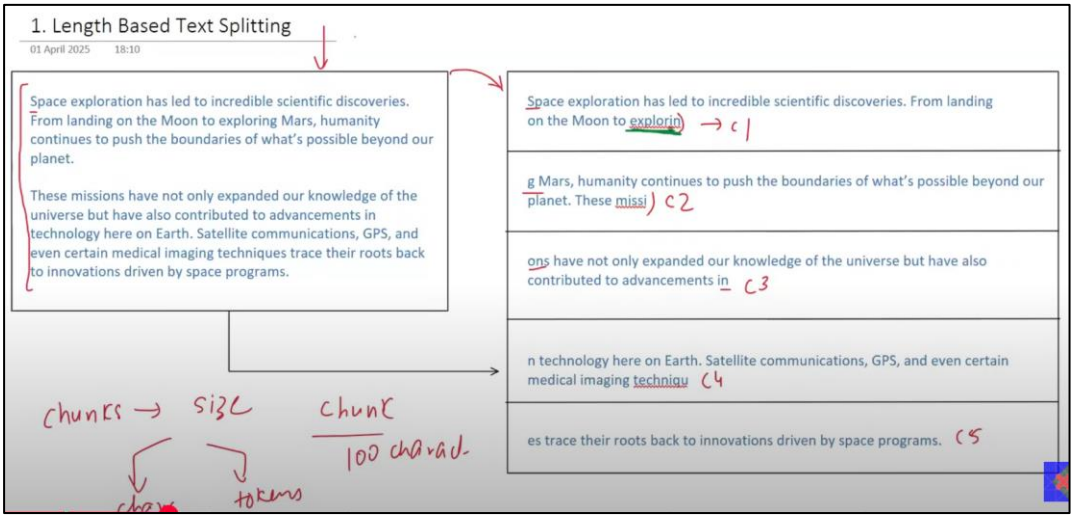
Text Splitters in Langchain



Why Text Splitting is Better?

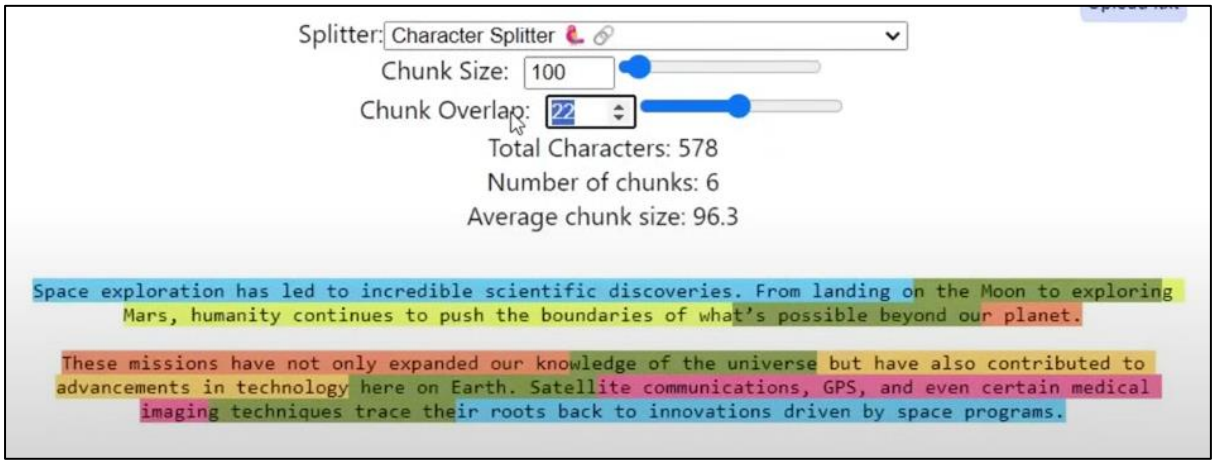


1. Length Based Text Splitting:



It is very **simple** and easy to use text splitting technique, but the only demerit is that it does not that while splitting the text it doesn't see the contextual meaning of any word or sentences, it only have to split the text (that's it).

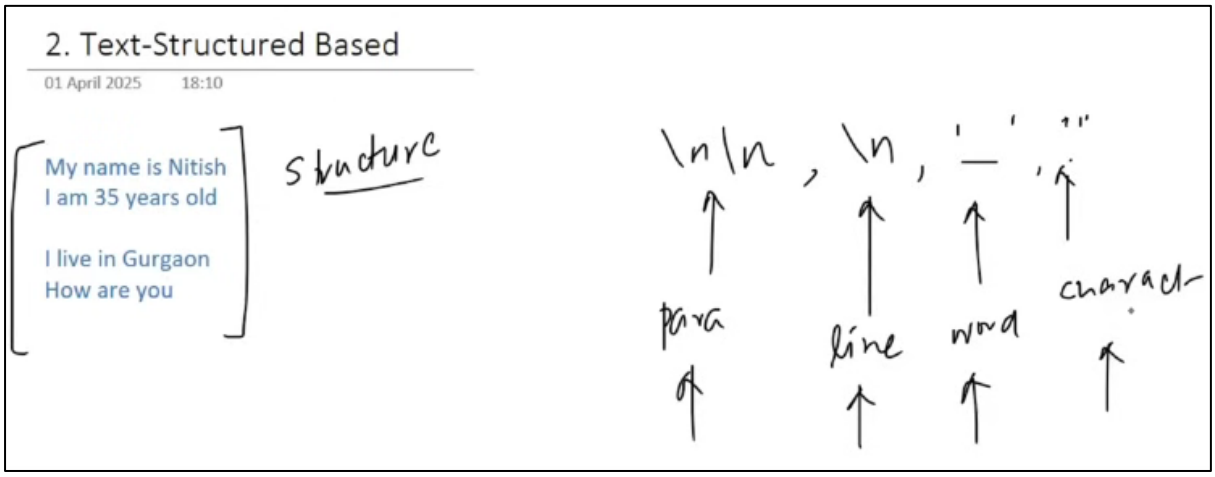
Chunk Overlap is a way in which jis se hum abruptly word jo kt rhe thae usko hum bacha skte hai



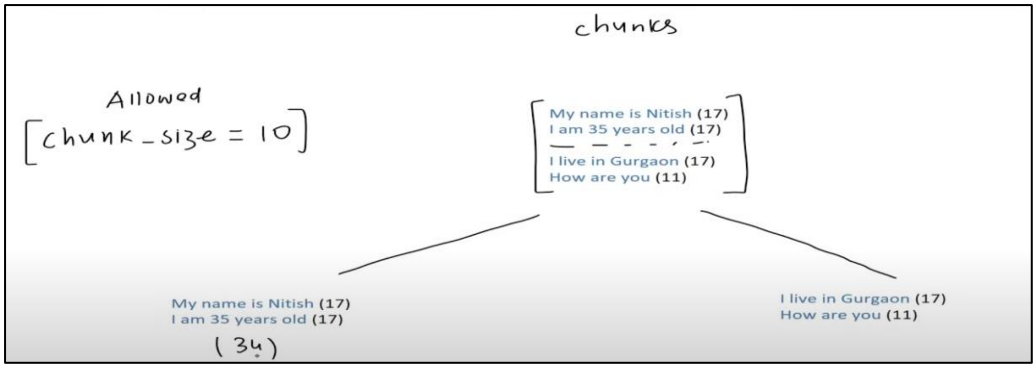
Tool to test different splitter: https://chunkviz.up.railway.app/

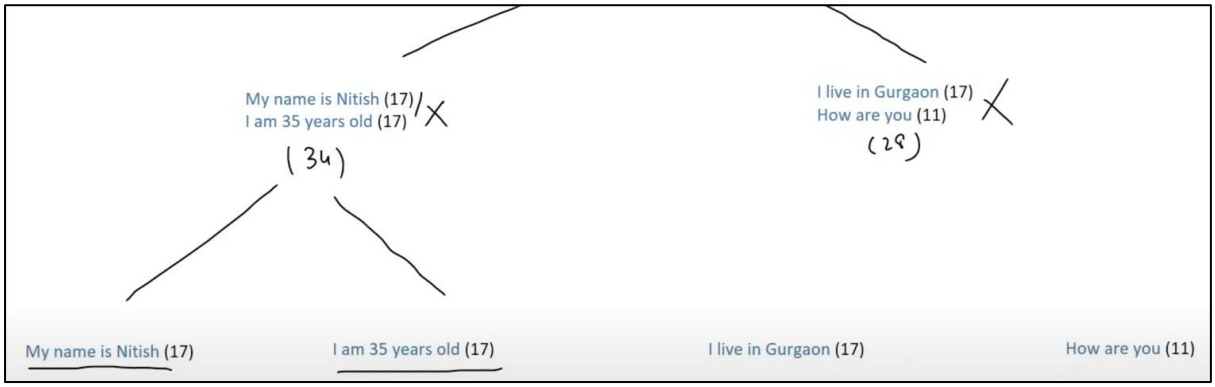
2. **Text Structure based Splitter** : This is the second splitter jisme algo. Neeche wala structure follow krta hai text ko split krne ke liye, pehle **paragraph** ke bases pe split krega agr chunk size chota hua toh phir **line** ke bases pe split krega, agr phir bhi chunk size chota ho, thab word ke basis pe, and so on up to **character**,

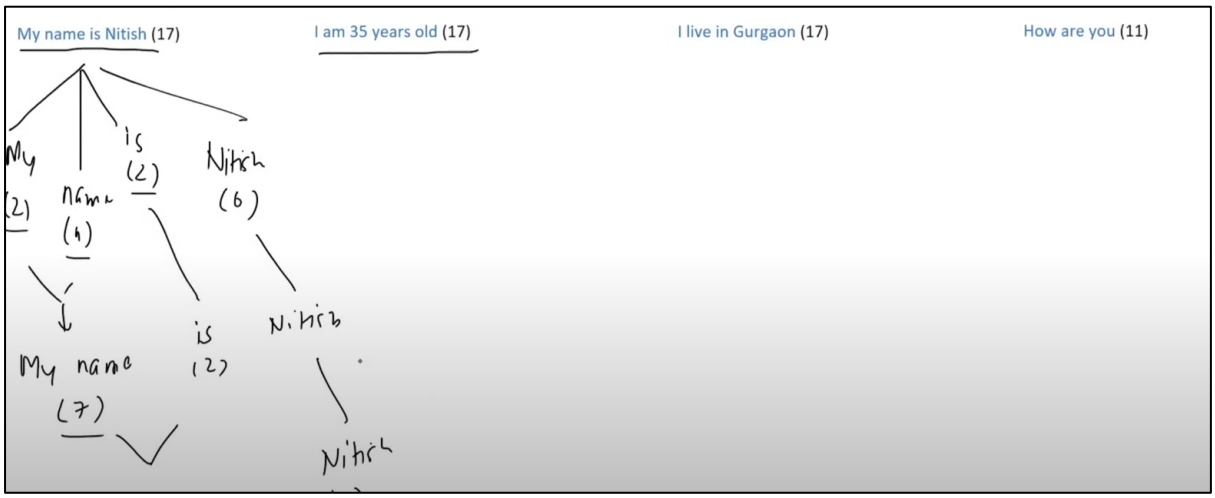
   And chunk size number of words ko bolte hai including spaces.

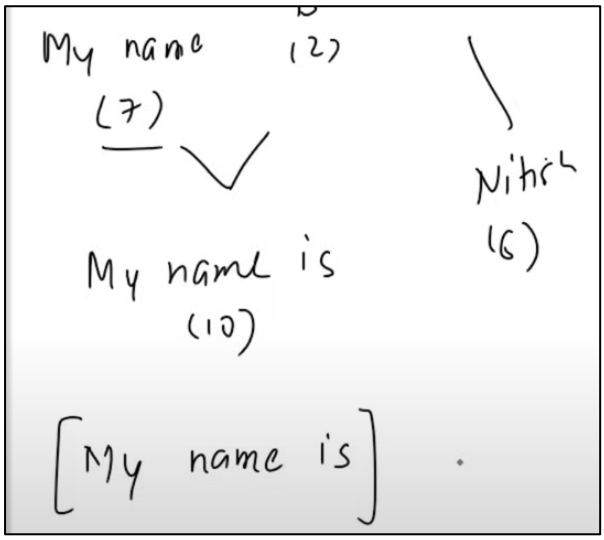


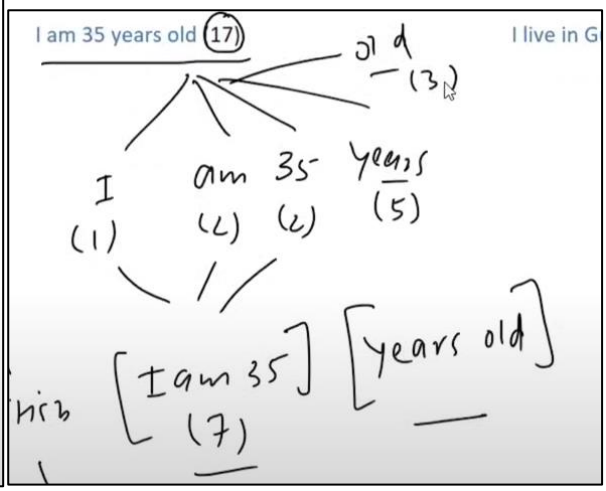
Example of Text Structure based Splitter:

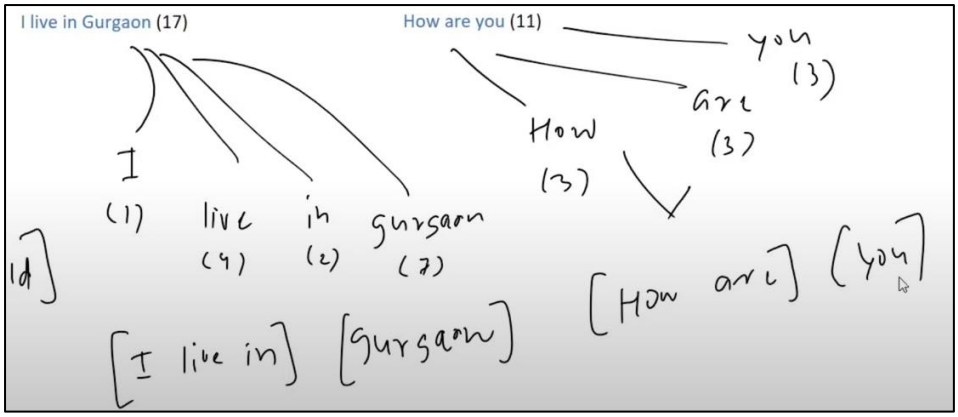

chunks
Allowed
[chunk_size = 10]
My name is Nitish (17)
I am 35 years old (17)
I live in Gurgaon (17)
How are you (11)
My name is Nitish (17)
I am 35 years old (17)
(34)
I live in Gurgaon (17)
How are you (11)


My name is Nitish (17)
I am 35 years old (17)
(34)
I live in Gurgaon (17)
How are you (11)
(29)
My name is Nitish (17)
I am 35 years old (17)
I live in Gurgaon (17)
How are you (11)

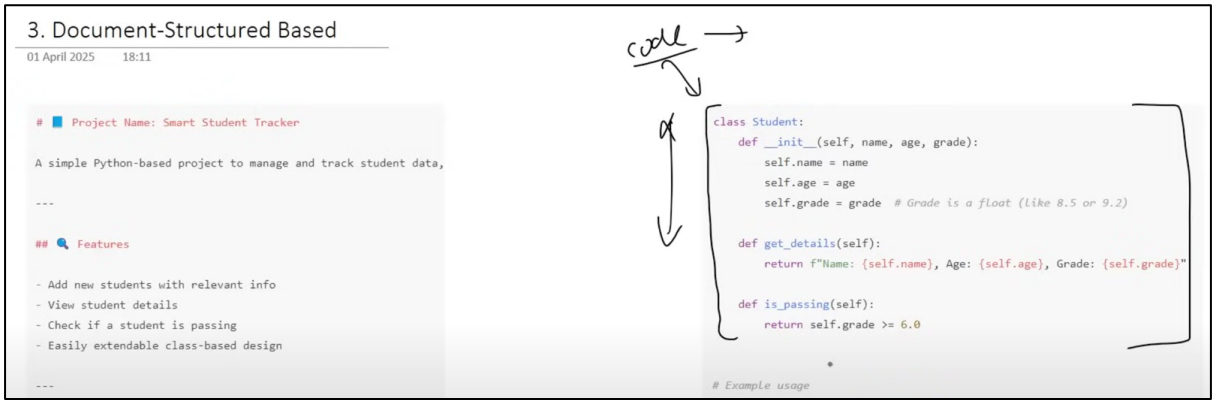

My name is Nitish (17)
I am 35 years old (17)
I live in Gurgaon (17)
How are you (11)
My
(2)
name
(4)
is
(2)
Nitish
(6)
My name
(7)
is
(2)
Nitish
Nitish

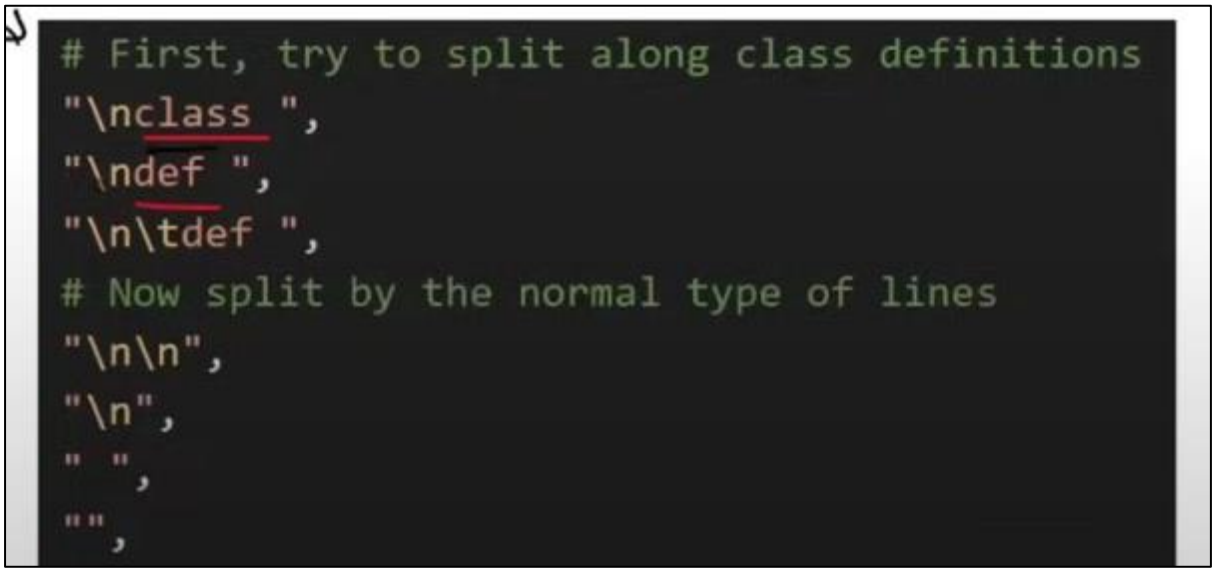

My name
(7)
(2)
Nitish
(6)
My name is
(10)
[My name is


I am 35 years old (17)
I live in G
old
(3)
I
(1)
am
(2)
35
(2)
years
(5)
[I am 35
(7)
[years old
Nitish



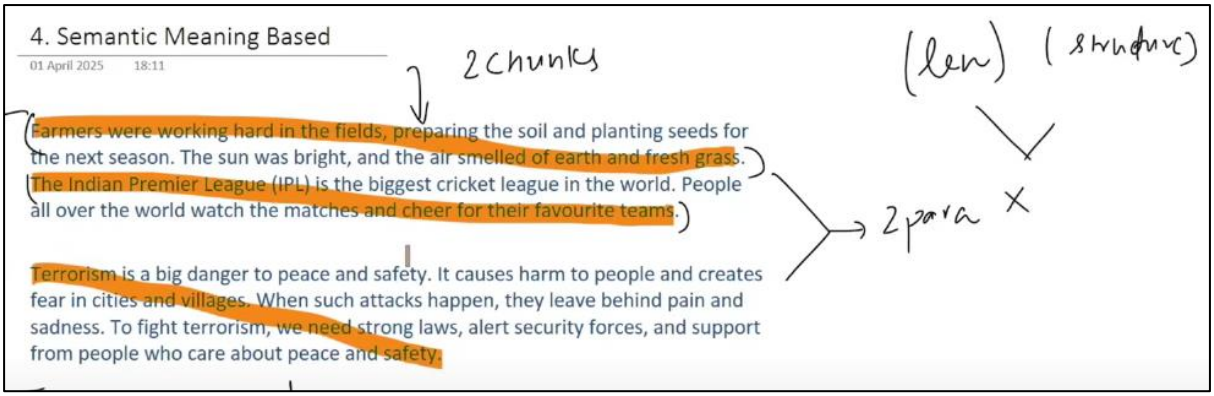
3. **Document Structure Based Splitter:** isme basically essi English text jo normal English se alag hoti hai usko yeh splitter alag krne me apply hota hai jesse Python code, Html code etc yeh sab bhi English lang mei likhe hote hai lekin normal text splitter se hum isko alag nhi kr skte (para -> line -> word -> char),





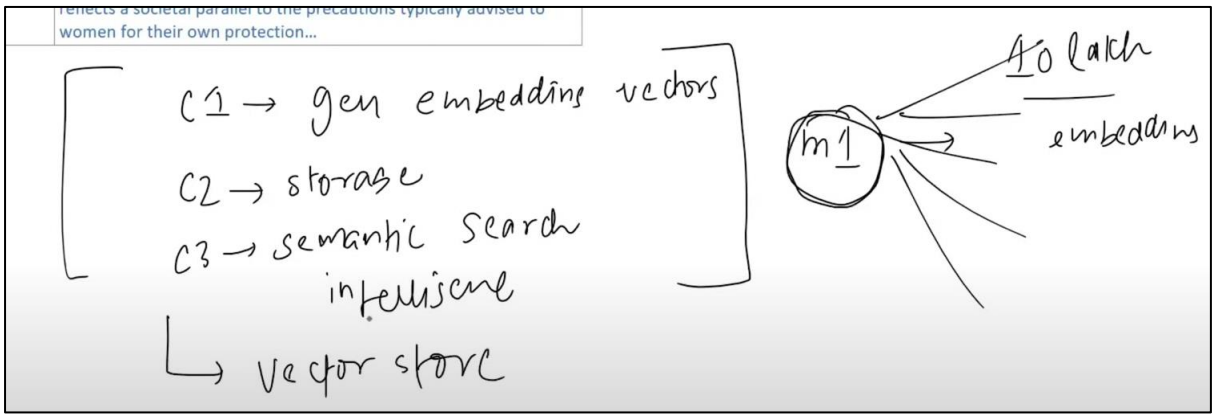
yeh upar wala python ke code ko alag krne ke liye hota hai

18

4. 
4. Semantic Meaning Based
01 April 2025 18:11
2 chunks
(len) (structure)
Farmers were working hard in the fields, preparing the soil and planting seeds for the next season. The sun was bright, and the air smelled of earth and fresh grass.
The Indian Premier League (IPL) is the biggest cricket league in the world. People all over the world watch the matches and cheer for their favourite teams.
Terrorism is a big danger to peace and safety. It causes harm to people and creates fear in cities and villages. When such attacks happen, they leave behind pain and sadness. To fight terrorism, we need strong laws, alert security forces, and support from people who care about peace and safety.
2 para X

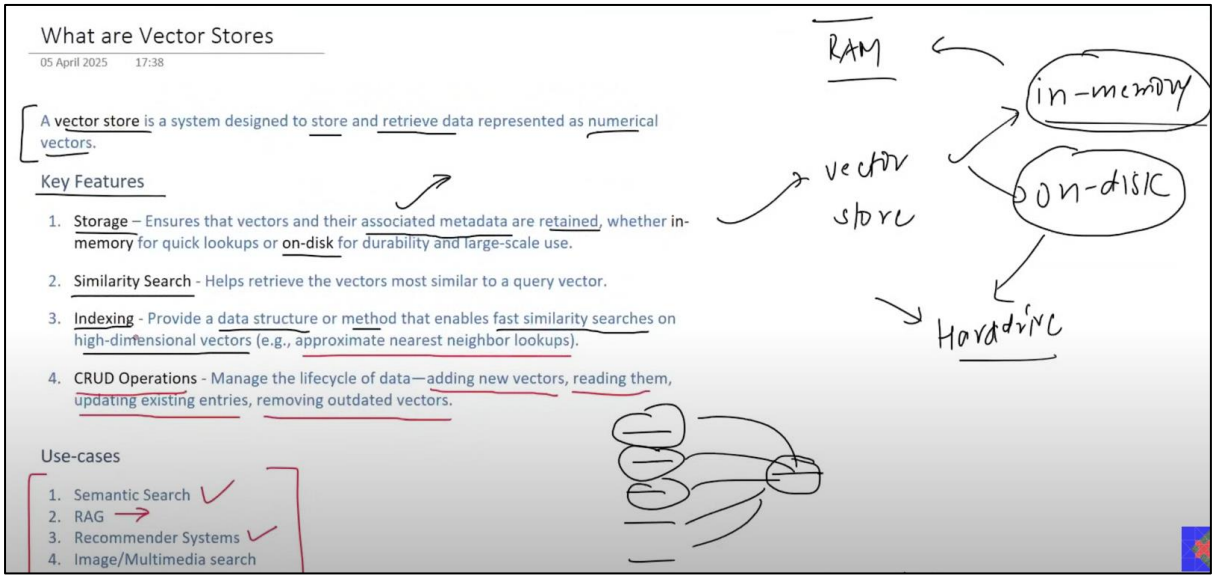
Vector Stores

Various challenges while making a semantic based movie recommendation systems:

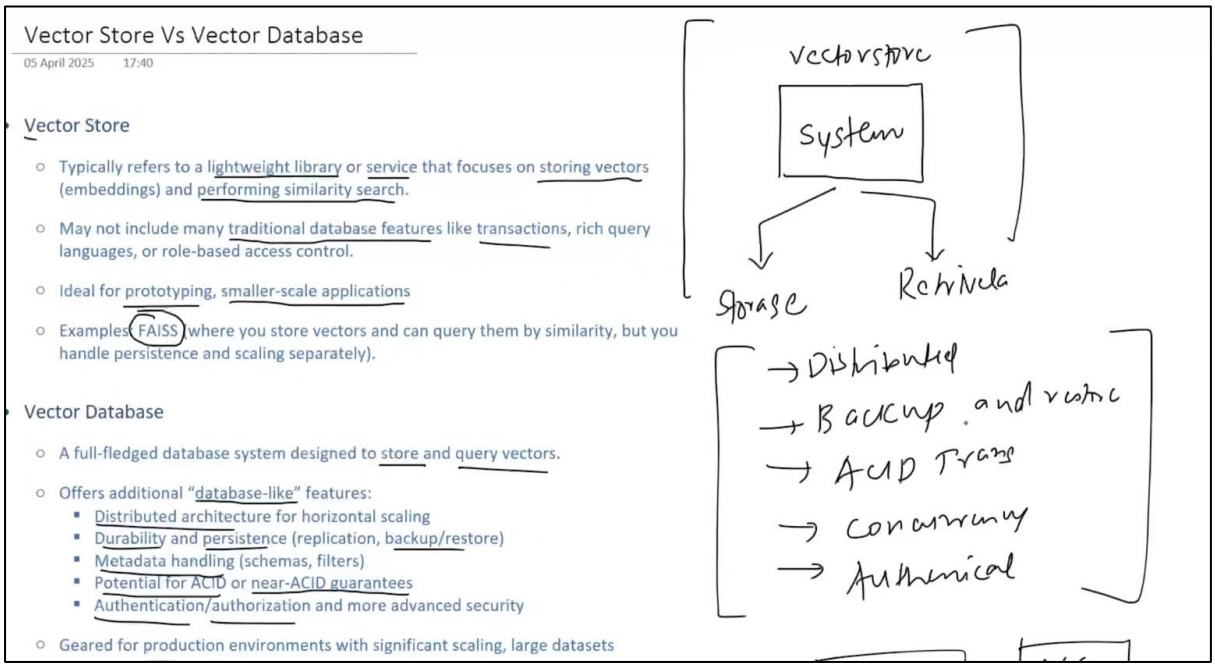


1. First challenge is we have to generate the embedding vectors for all movies.
2. We cannot store embedding in the normal db like oracle, mysql, because in that we can do the semantic search easily.
3. And we want to build intelligent semantic search because for one movie we cannot compare the embedding vector of that one movie with the other 10lakh or crores movies embedding vectors.

So these are the few challenges that we will face while making the semantic based movie recommendation system, and you know what that all these problem are solved by **VECTOR SPACE.**



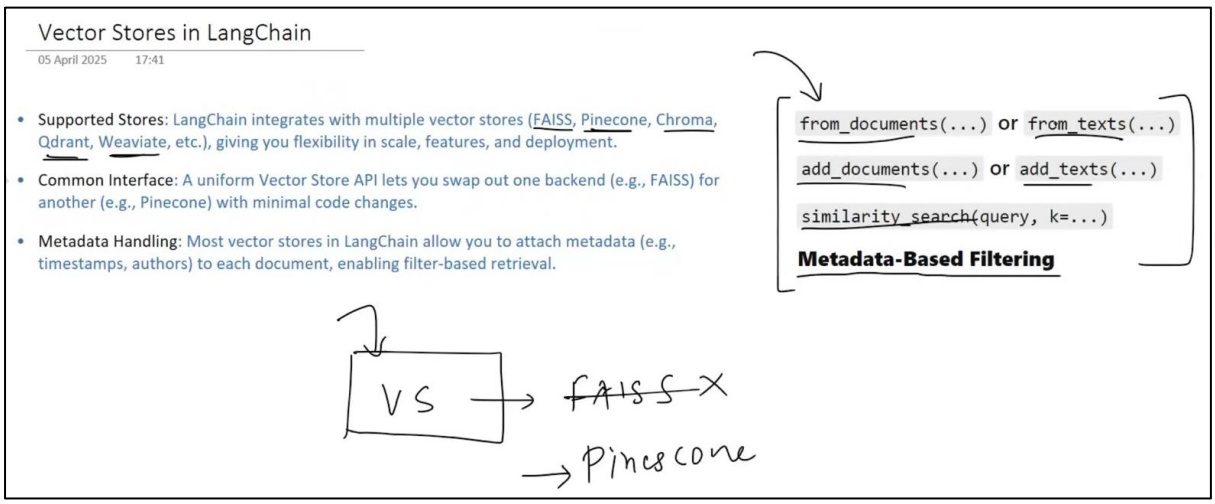
Vector Store vs Vector DB



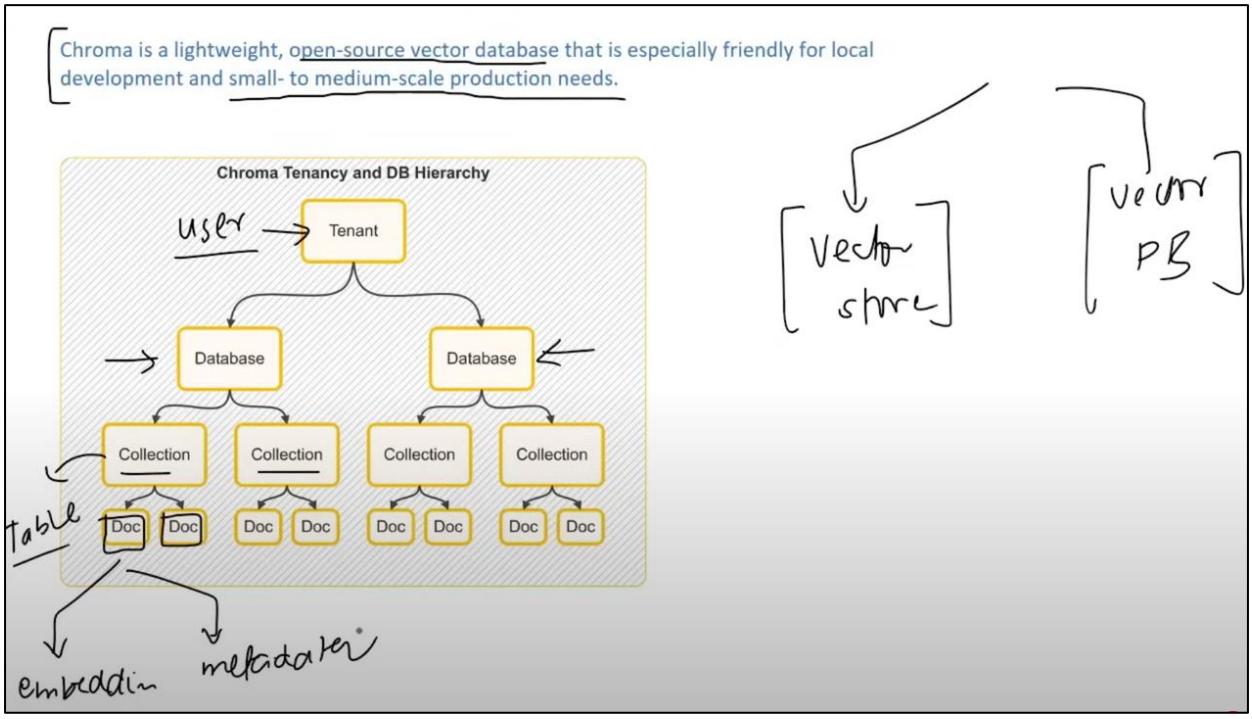
A vector database is effectively a vector store with extra database features (e.g., clustering, scaling, security, metadata filtering, and durability)

All vector DB are vector Store, but vice versa not true.

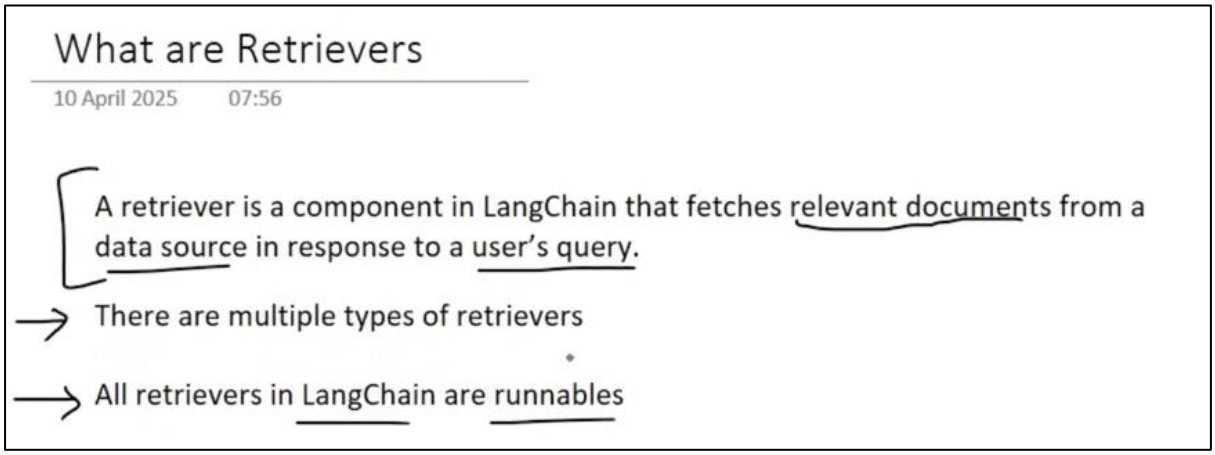
Vector Store voh system hote hai jisme STORAGE aur RETRIEVAL ka features hota hai and Vector DB voh hota hai jisme yeh doo feature toh hote hee hai, but lekin saath mei jo normal DB ke feature hote hai voh bhi isme hote hai. That's why all Vector DB are Vector Store , but all Vector Store are not Vector DB.

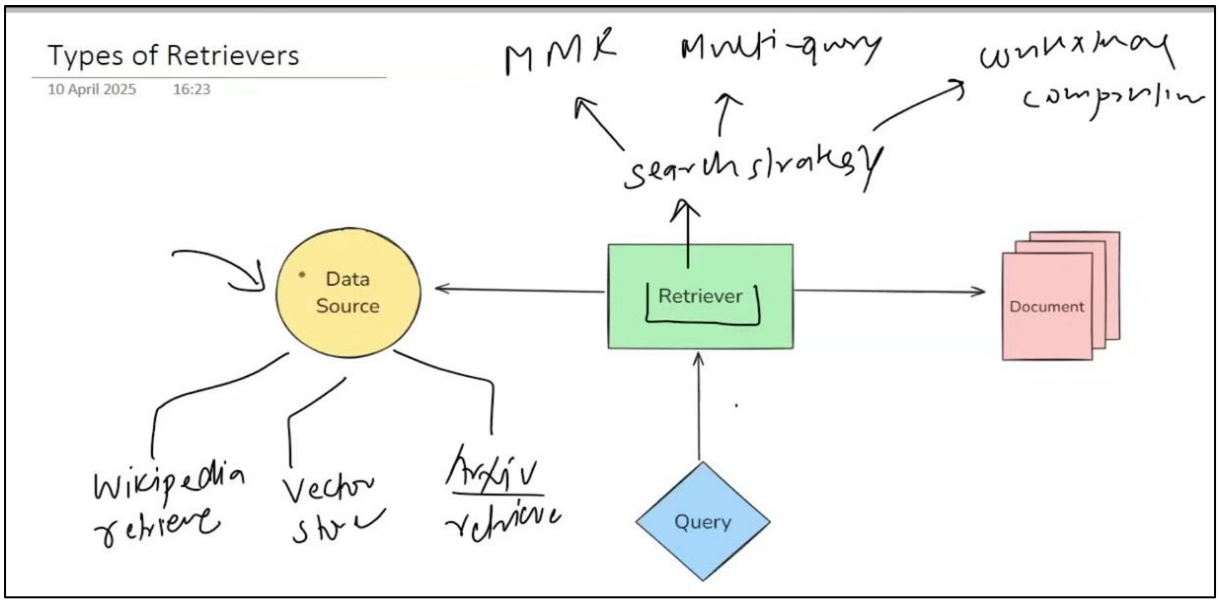


Chroma DB:

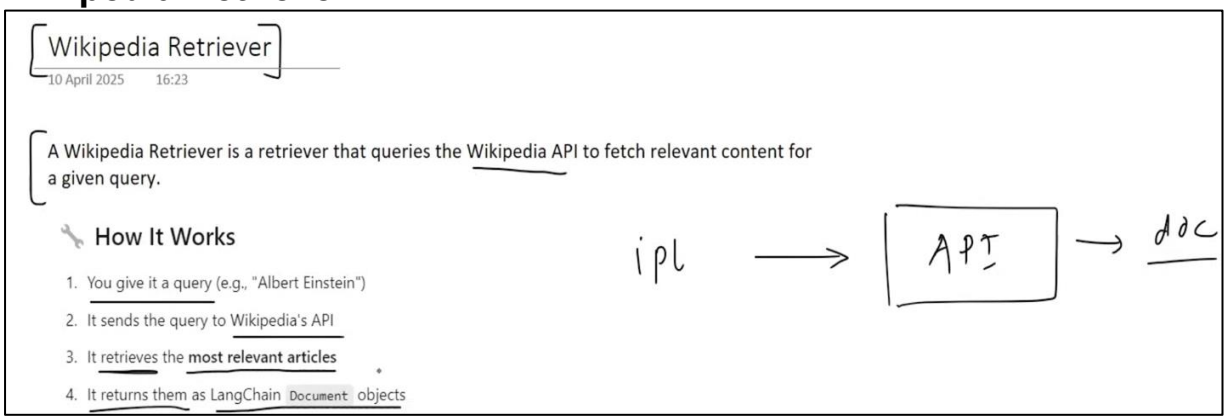

Chroma is a lightweight, open-source vector database that is especially friendly for local development and small- to medium-scale production needs.
Chroma Tenancy and DB Hierarchy
Tenant
Database
Database
Collection
Collection
Collection
Collection
Doc
Doc
Doc
Doc
Doc
Doc
Doc
Doc

Retriever

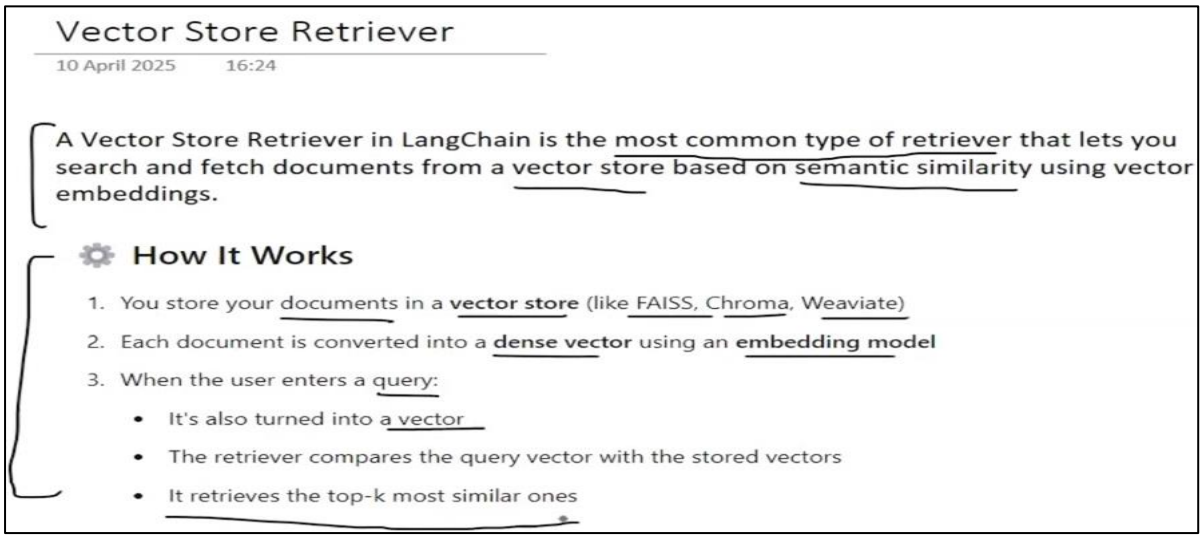




Wikipedia Retriever

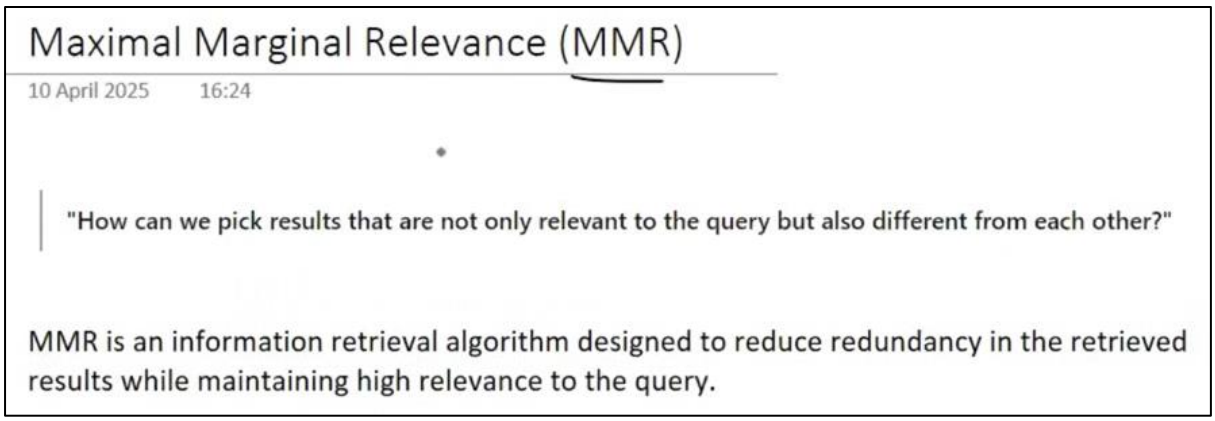


Vector Retriever

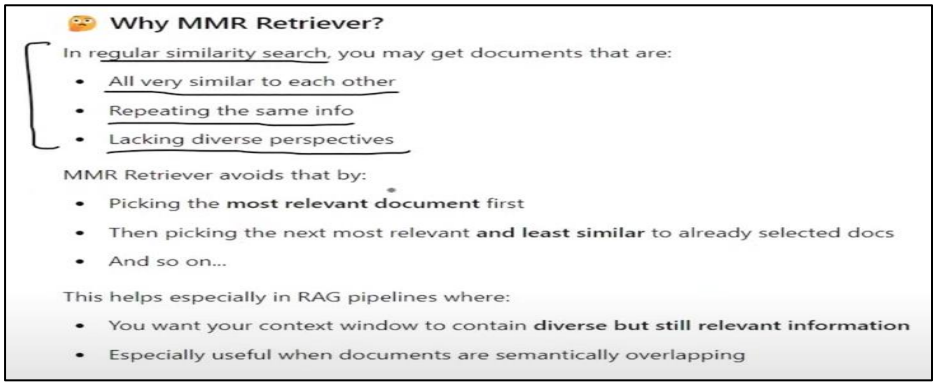


Wikipedia and Vector Store retriever both are data source side retriever means yeah data base wali side pe se hume most relevant result deta hai, bole toh semantic similarity meaning ke base pr user ko output me result deta hai.

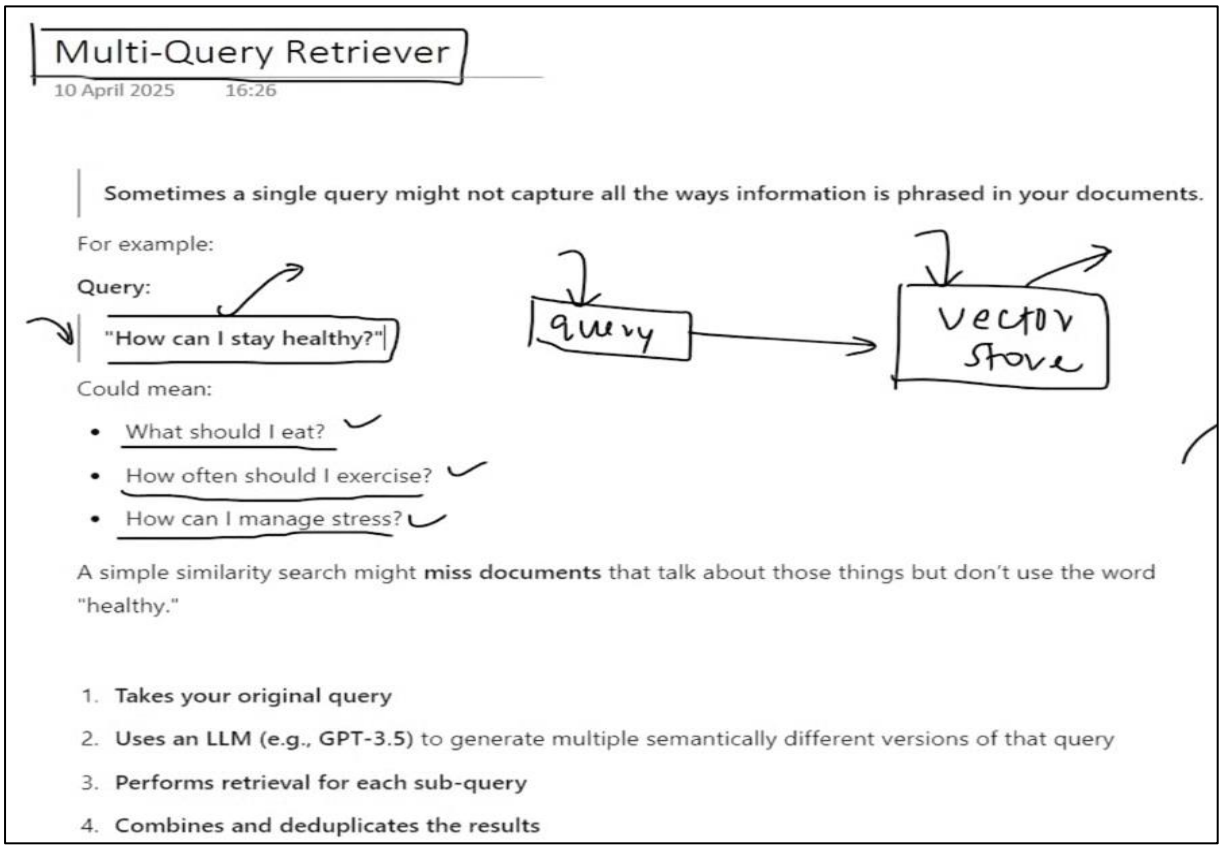
MMR Retriever



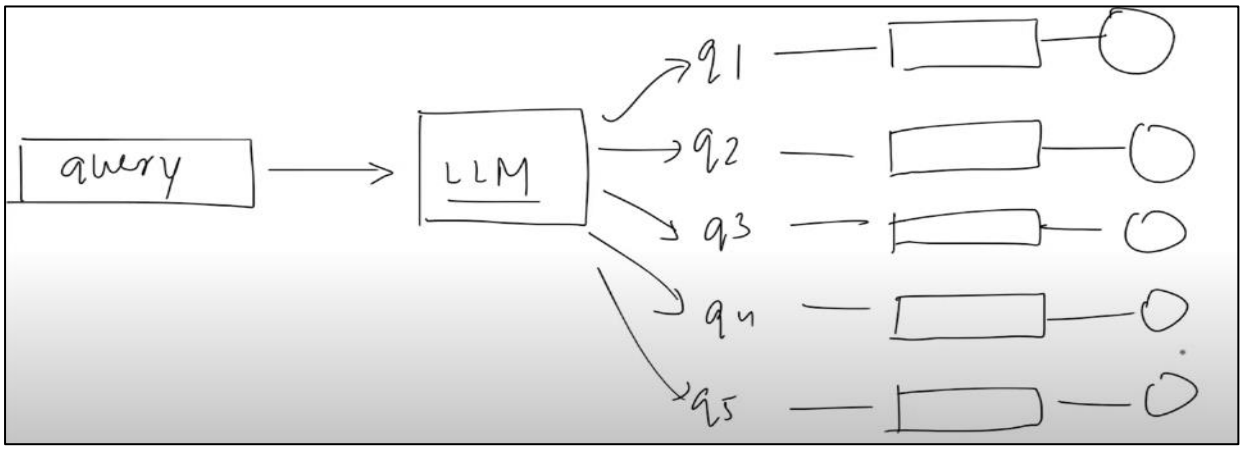
Yeah jo MMR retriever hai voh Retriever ke side wala retriever hai, bole toh yeah redundant result ko hatane ka kaam krta hai aur output mei diversification lane ki kosish krta hai.



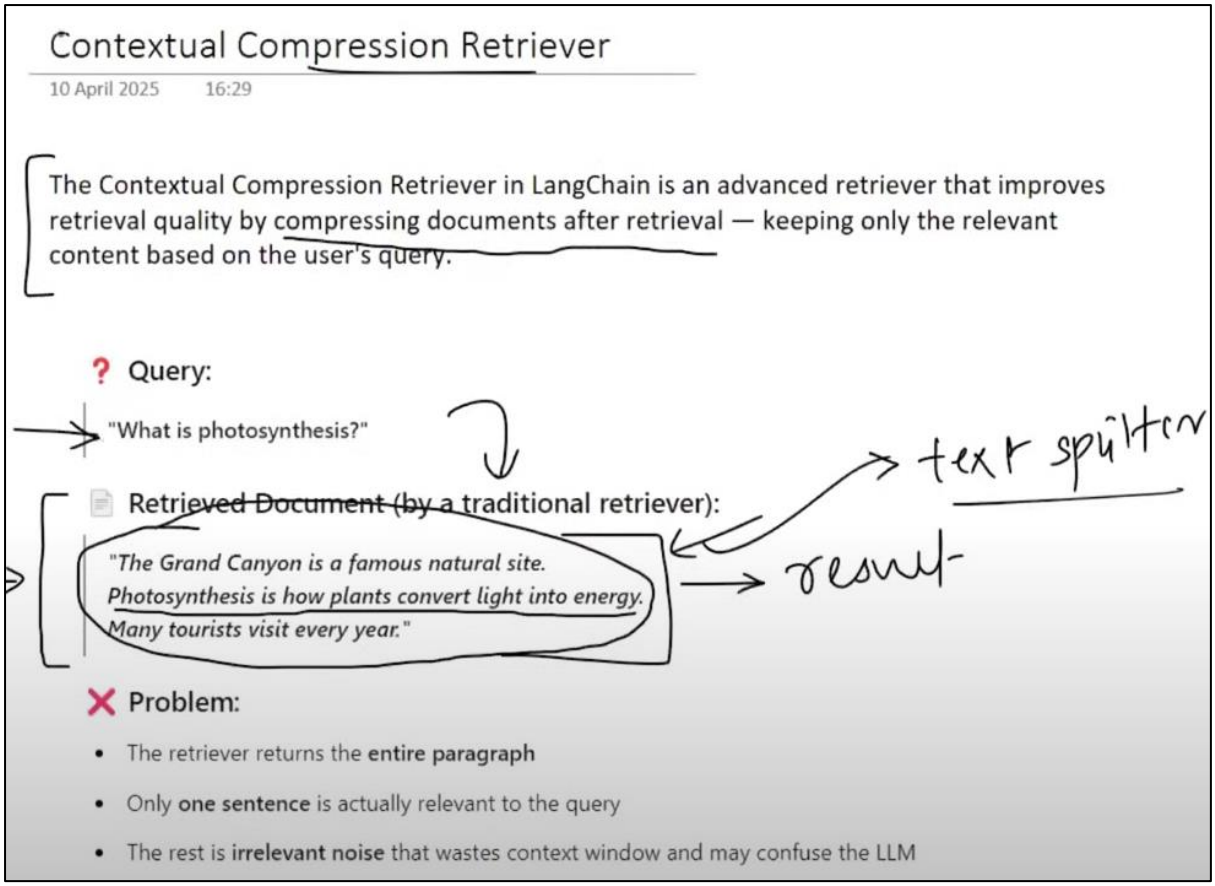
Multi-Query Retriever



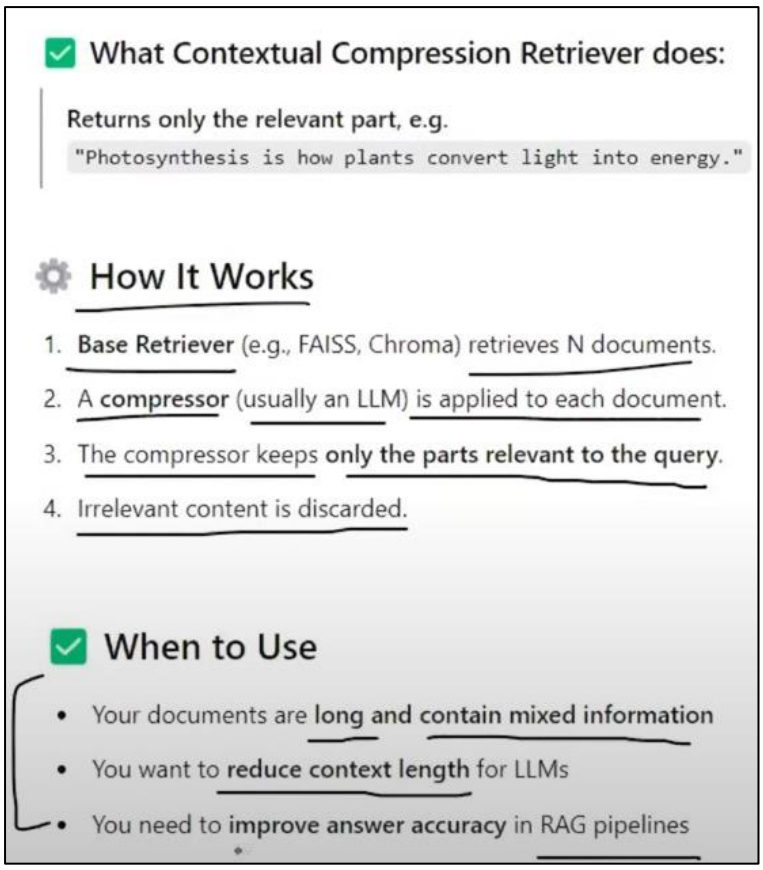
The working of this retriever is very simple that, it take one ambiguity query from the user, give it to LLM, than LLM will generate different semantically version of query and that all queries will go to the retrieval , it found the relevant documents/output and every subquery has it own document/output, so at last we remove the redundancy by applying MMR retriever.

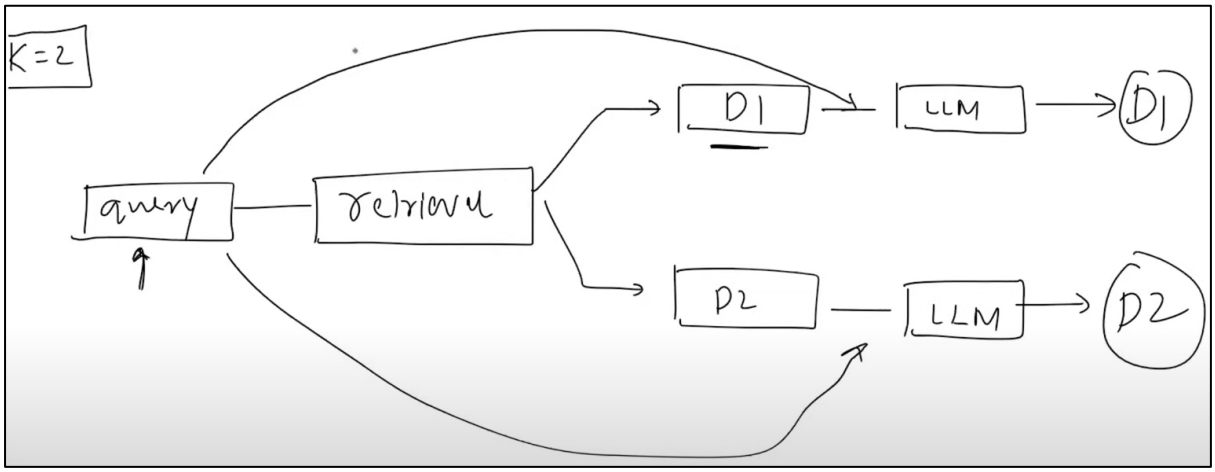


Contextual Compression Retriever



Yeh retriever waha use hota hai, jaha pr eek document mei doo teen cheeze ho aur user ko agr particular eek hee cheez ko dekhna ho toh us case mei baaki ke text ko compress kr dena LLM ki help sei aur phir output dikhana.





https://python.langchain.com/docs/integrations/retrievers/

For more retriever follow the above link, also yaad rhe retrievers are runnables means we can plug it into chains.

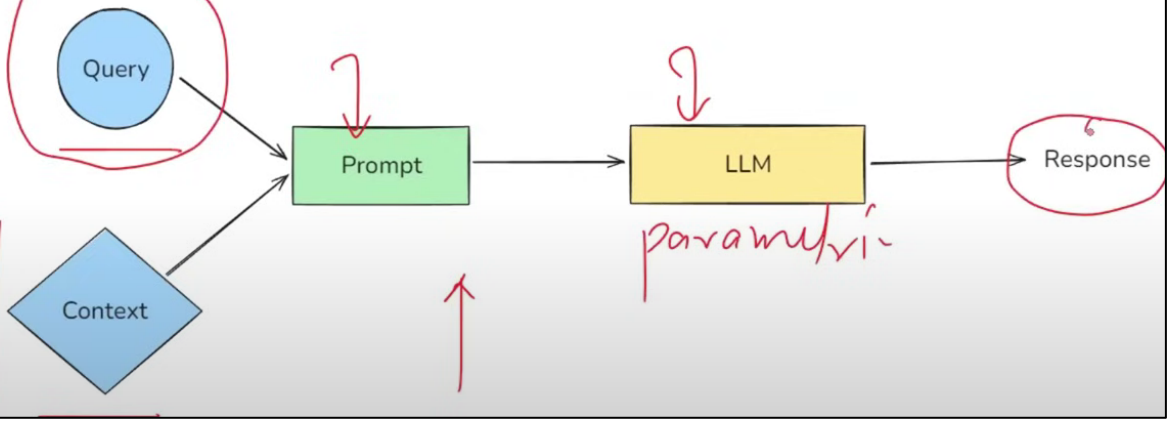
In-Context Learning: incontext learning agr simple words mei bataou toh essa prompt dena llm ko jisme eek doo short example ho apki query se related jo apki query ko solve krne mei help krega, so inhi short examples ki vegah sei apka llm apke query ko ache sei solve kr payega.

In-Context Learning is a core capability of Large Language Models (LLMs) like GPT-3/4, Claude, and Llama, where the model learns to solve a task purely by seeing examples in the prompt—without updating its weights.

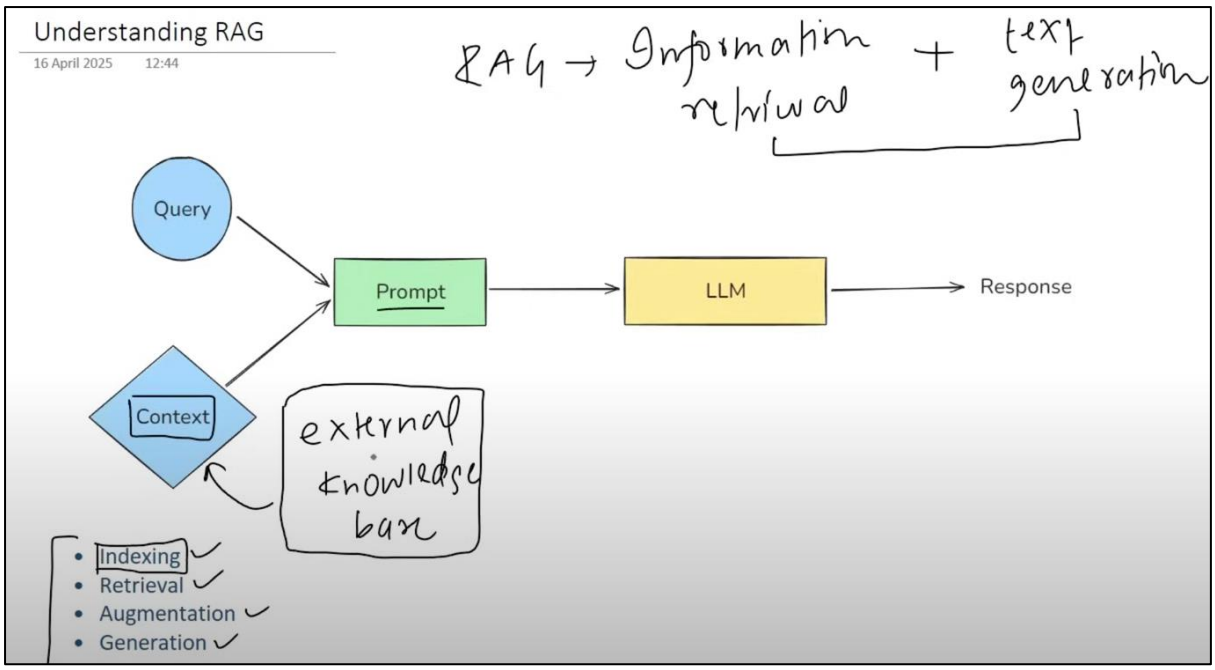
LLM → emergent prop

An **emergent property** is a behaviour or ability that suddenly appears in a system when it reaches a certain scale or complexity—even though it was not explicitly programmed or expected from the individual components.

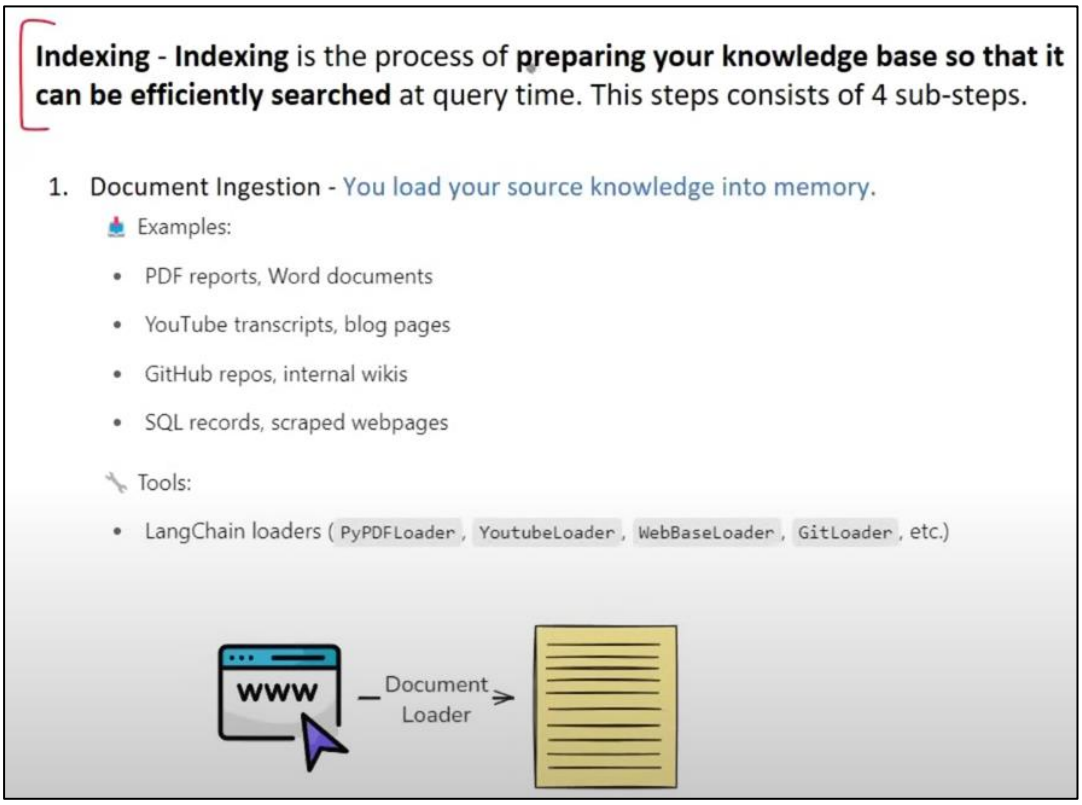
RAG is a way to make a language model (like ChatGPT) smarter by giving it extra information at the time you ask your question.



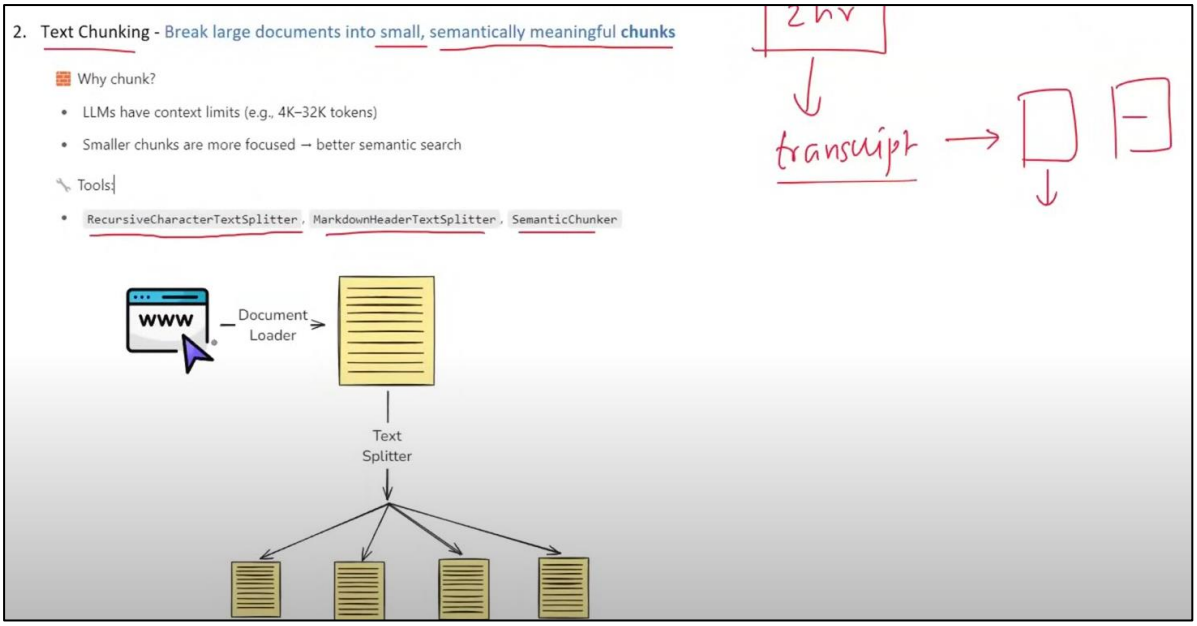

```
"""You are a helpful assistant.
Answer the question ONLY from the provided context.
If the context is insufficient, just say you don't know.

{context}
Question: {question}"""
```



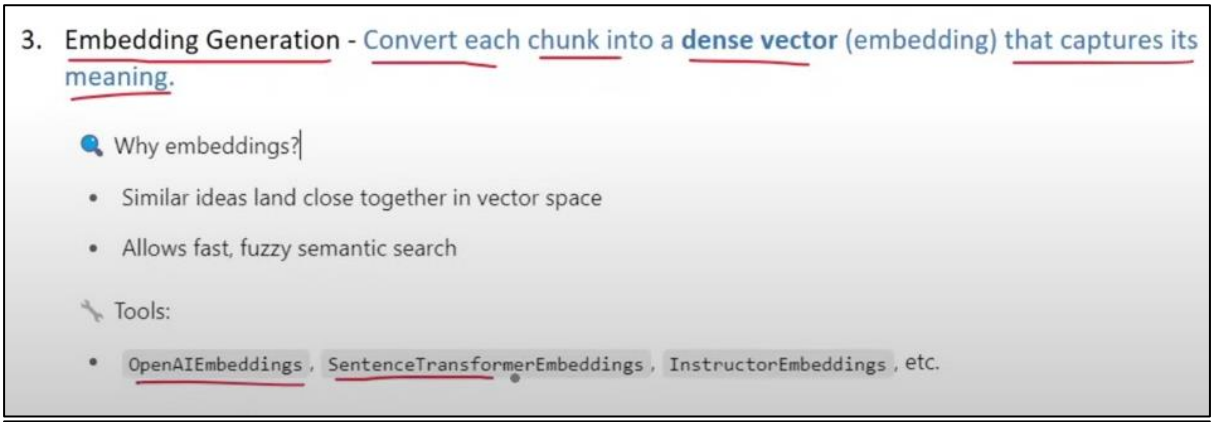
Full work Flow of Implementing RAG:

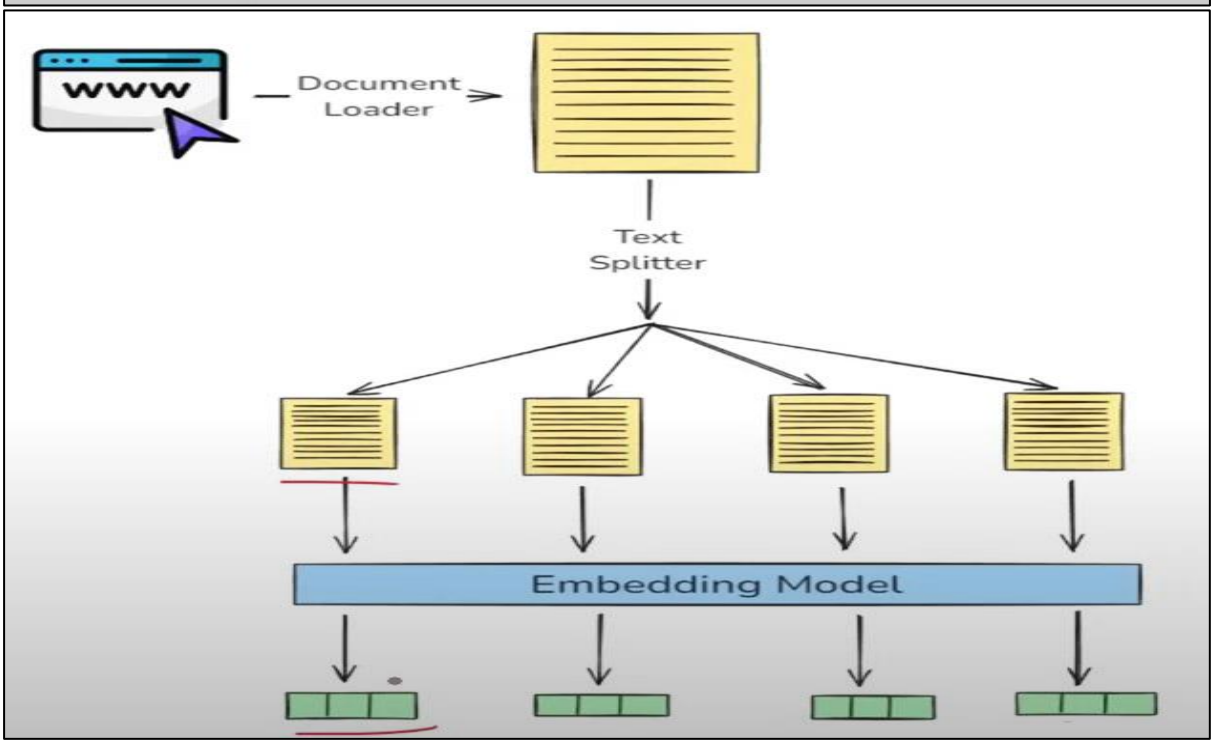
1. 

2.


2. Text Chunking - Break large documents into small, semantically meaningful chunks
Why chunk?
• LLMs have context limits (e.g., 4K–32K tokens)
• Smaller chunks are more focused → better semantic search
Tools:
• RecursiveCharacterTextSplitter, MarkdownHeaderTextSplitter, SemanticChunker
www
Document Loader
Text Splitter
transcript

3.


3. Embedding Generation - Convert each chunk into a dense vector (embedding) that captures its meaning.
Why embeddings?
• Similar ideas land close together in vector space
• Allows fast, fuzzy semantic search
Tools:
• OpenAIEmbeddings, SentenceTransformerEmbeddings, InstructorEmbeddings, etc.

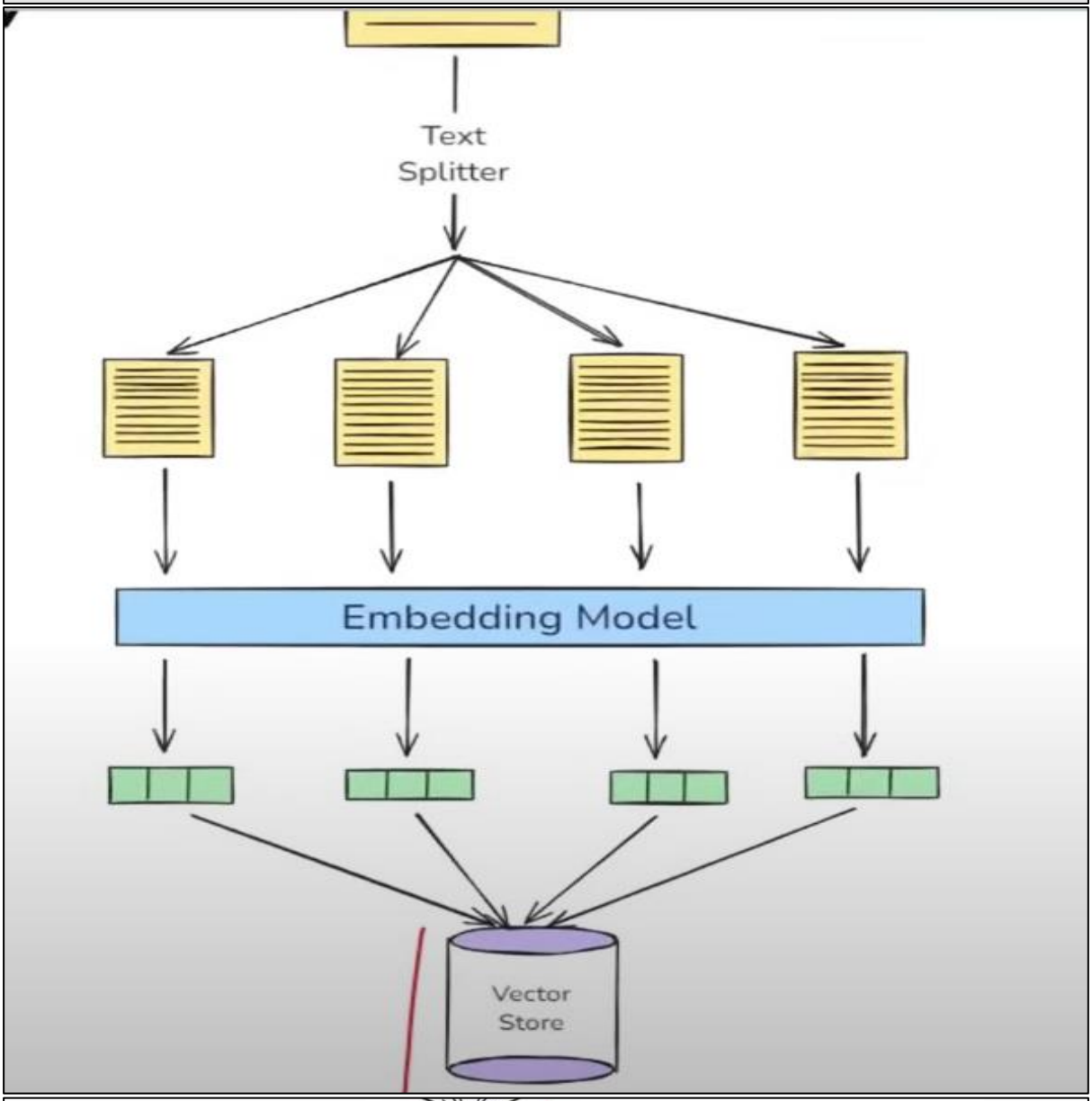

www
Document Loader
Text Splitter
Embedding Model

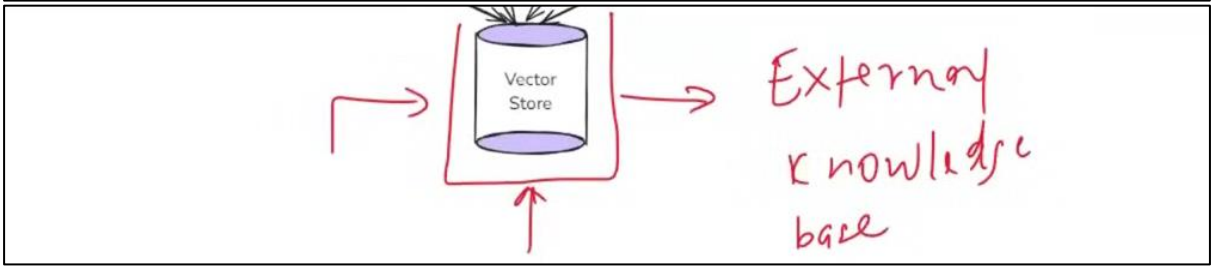
4. Storage in a Vector Store - Store the vectors along with the original chunk text + metadata in a vector database.

   Vector DB options:

   - Local: `FAISS`, `Chroma`
   - Cloud: `Pinecone`, `Weaviate`, `Milvus`, `Qdrant`

4.



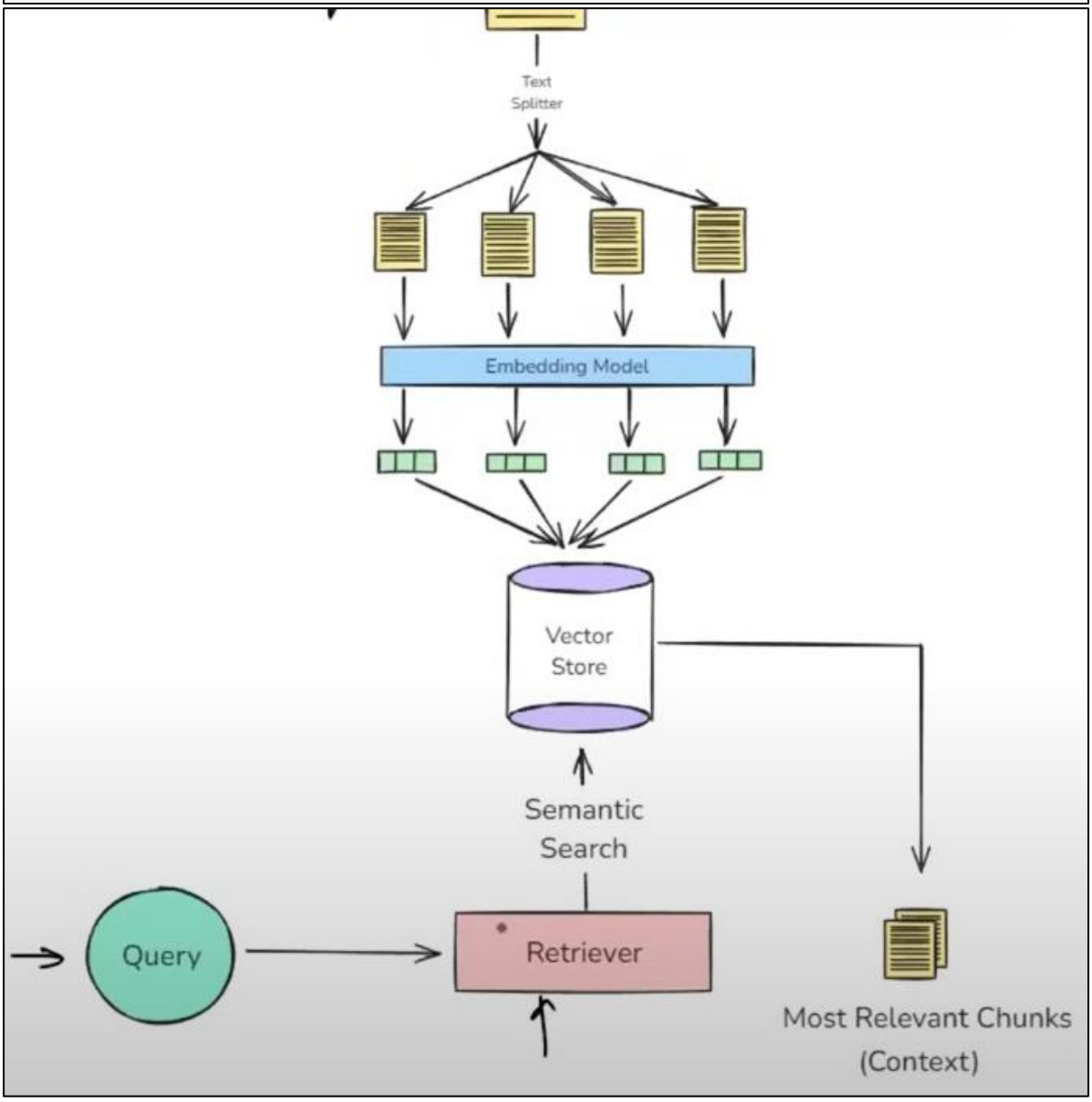


5.

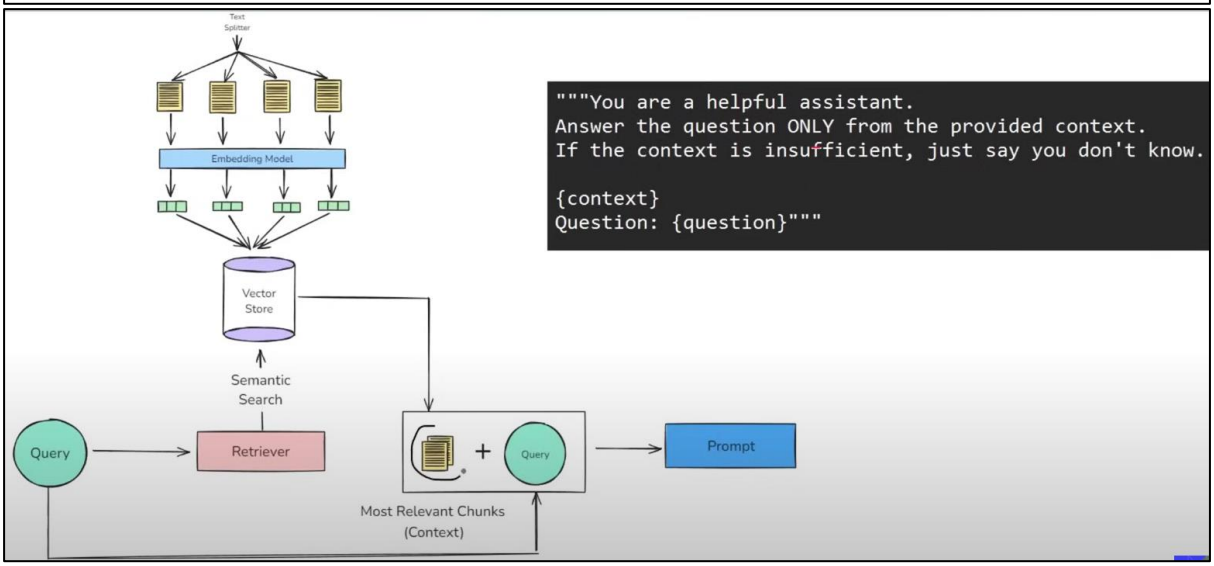
Retrieval - **Retrieval** is the *real-time* process of **finding the most relevant pieces of information** from a pre-built index (created during indexing) based on the user's question.

It's like asking:

> "From all the knowledge I have, which 3–5 chunks are most helpful to answer this query?"



6. **Augmentation** - **Augmentation** refers to the step where the **retrieved documents** (chunks of relevant context) are **combined with the user's query** to form a new, enriched prompt for the LLM.



7. **Generation** - **Generation** is the final step where a **Large Language Model (LLM)** uses the **user's query** and the **retrieved & augmented context** to generate a response.

Architecture of RAG: