

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 give 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.

Some Famous Open Source Models

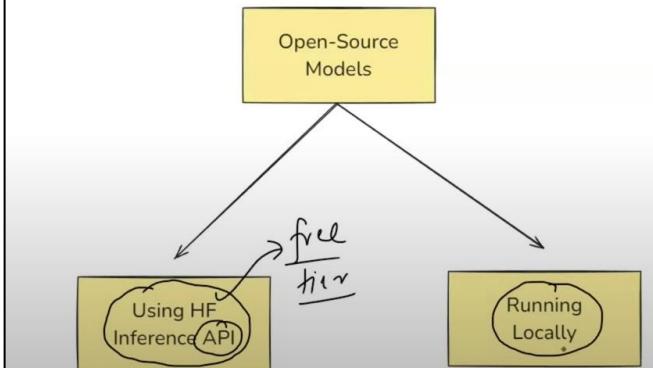
Model	Developer	Parameters	Best Use Case
LLaMA-2-7B/13B/70B	Meta AI	7B - 70B	General-purpose text generation
Mixtral-8x7B	Mistral AI	8x7B (MoE)	Efficient & fast responses
Mistral-7B	Mistral AI	7B	Best small-scale model (outperforms LLaMA-2-13B)
Falcon-7B/40B	TII UAE	7B - 40B	High-speed inference
BLOOM-176B	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

3. Open Source Models

Where to find them?

HuggingFace - The largest repository of open-source LLMs

Ways to use Open-source Models



4. Where to find the Open Source Models

5. Structured Output and Unstructured Output

Structured Output

28 February 2025 00:13

In LangChain, structured output refers to the practice of having language models return responses in a well-defined data format (for example, JSON), rather than free-form text. This makes the model output easier to parse and work with programmatically.

[Prompt] - [Can you create a one-day travel itinerary for Paris?]

[LLM's Unstructured Response]

Here's a suggested itinerary: Morning: Visit the Eiffel Tower.
Afternoon: Walk through the Louvre Museum.
Evening: Enjoy dinner at a Seine riverside café.

[JSON enforced output]

→ unstructured →
→ structured →

["time": "Morning", "activity": "Visit the Eiffel Tower"]

Ways to get Structured Output

28 February 2025 00:14

LLMs

[can]
[can't]
with_structured_output
Output parsers

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.

with_structured_output → data-format

28 February 2025 00:15

(model-inference)

You are an AI assistant that extracts structured insights from text. Given a product review, extract: - Summary: A brief overview of the main points. - Sentiment: Overall tone of the review (positive, neutral, negative). Return the response in JSON format.

TypedDict

Pydantic

json-schema

See jo `with_structured_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 theae lln ko voh hume output data 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!!

Pydantic

01 March 2025 12:59

Pydantic is a data validation and data parsing library for Python. It ensures that the data you work with is correct, structured, and type-safe.

When to use what?

01 March 2025 12:59

- Use `TypedDict` if:**
 - You only need type hints (basic structure enforcement).
 - You don't need validation (e.g., checking numbers are positive).
 - You trust the LLM to return correct data.
- Use `Pydantic` if:**
 - You need data validation (e.g., sentiment must be "positive", "neutral", or "negative").
 - You need default values if the LLM misses fields.
 - You want automatic type conversion (e.g., "100" → 100).
- Use `JSON Schema` if:**
 - You don't want to import extra Python libraries (Pydantic).
 - You need validation but don't need Python objects.
 - You want to define structure in a standard JSON format.

When to Use What?

Feature	TypedDict ✓	Pydantic ✘	JSON Schema ●
Basic structure	✓	✓	✓
Type enforcement	✓	✓	✓
Data validation	✗	✓	✓
Default values	✗	✓	✗
Automatic conversion	✗	✓	✗
Cross-language compatibility	✗	✗	✓

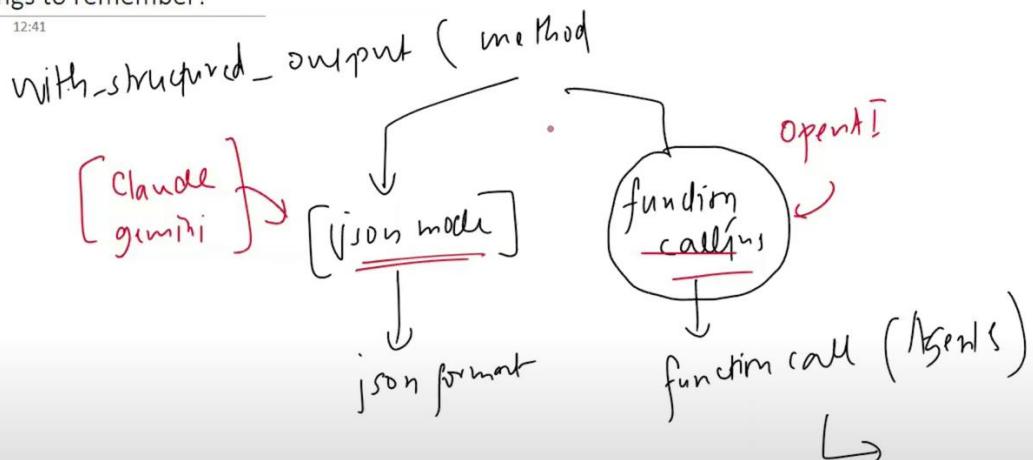
Kabhi kabhi hum Hugging face ke open source models ko use krte hai structure output ke liye, lekin most of the open-source model structure output support nhi krte,

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

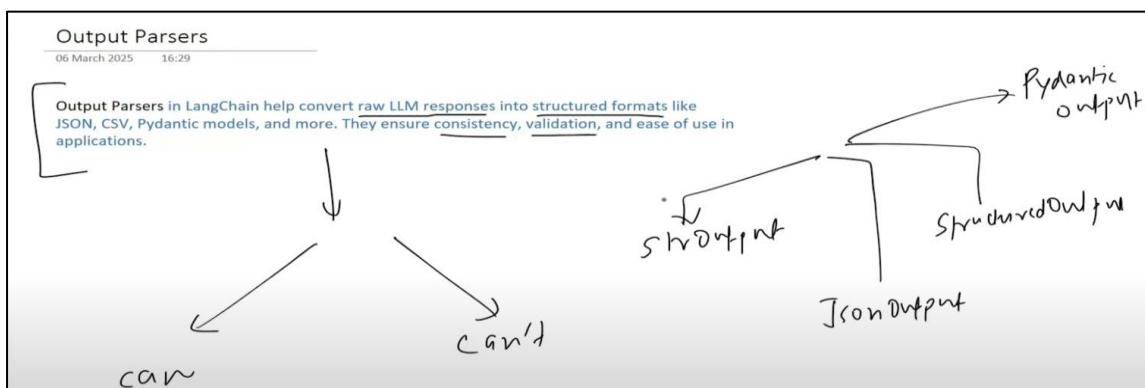
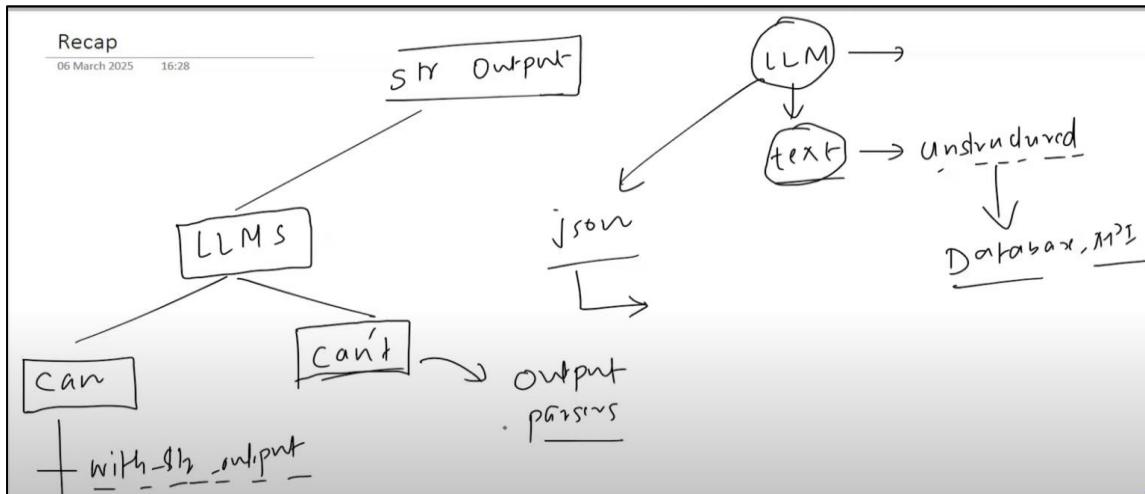
With_structured_output() mei eek attribute hota hai jiski doo values hoti hai one is json mode and another is function calling, inka actual kam toh nhi pta lekin default value fxn calling hota hai jo hum openAI ke model ke saath use kr skte hai

A few things to remember!

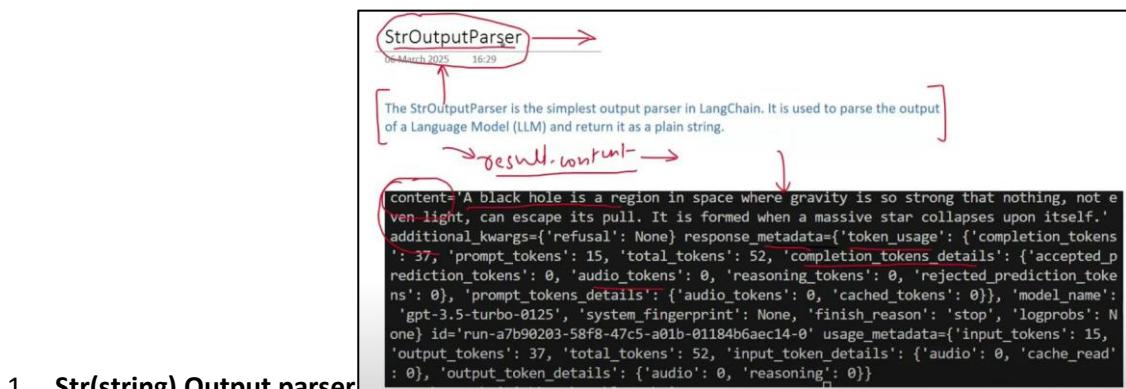
03 March 2025 12:41



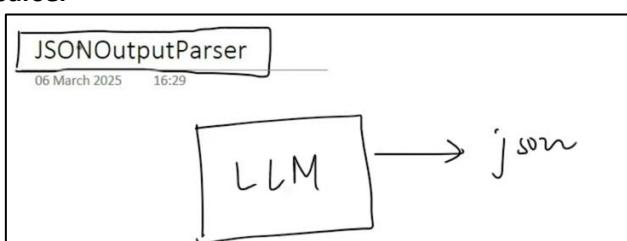
OUTPUT PARSER: are those method with which we can convert the unstructured data into structured data.



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.

StructuredOutputParser

06 March 2025 16:29

StructuredOutputParser is an output parser in LangChain that helps extract structured JSON data from LLM responses based on predefined field schemas.

It works by defining a list of fields (`ResponseSchema`) that the model should return, ensuring the output follows a structured format.

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.

PydanticOutputParser

06 March 2025 16:30

• What is PydanticOutputParser in LangChain?

PydanticOutputParser is a structured output parser in LangChain that uses Pydantic models to enforce schema validation when processing LLM responses.

↗ Why Use PydanticOutputParser?

- Strict Schema Enforcement → Ensures that LLM responses follow a well-defined structure.
- Type Safety → Automatically converts LLM outputs into Python objects.
- Easy Validation → Uses Pydantic's built-in validation to catch incorrect or missing data.
- Seamless Integration → Works well with other LangChain components.

→ Schema ↴
 [Pydantic
Object]

Pydantic 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 terms, 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 fcn mei input jaye, toh toh uss fcn 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.:

`chain = prompt | llm | output_parser`

3. Consistent Interface:

Every Runnable implements methods like:

- o `invoke()` – for synchronous execution
- o `stream()` – for streaming output
- o `batch()` – for batch processing
- o `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.



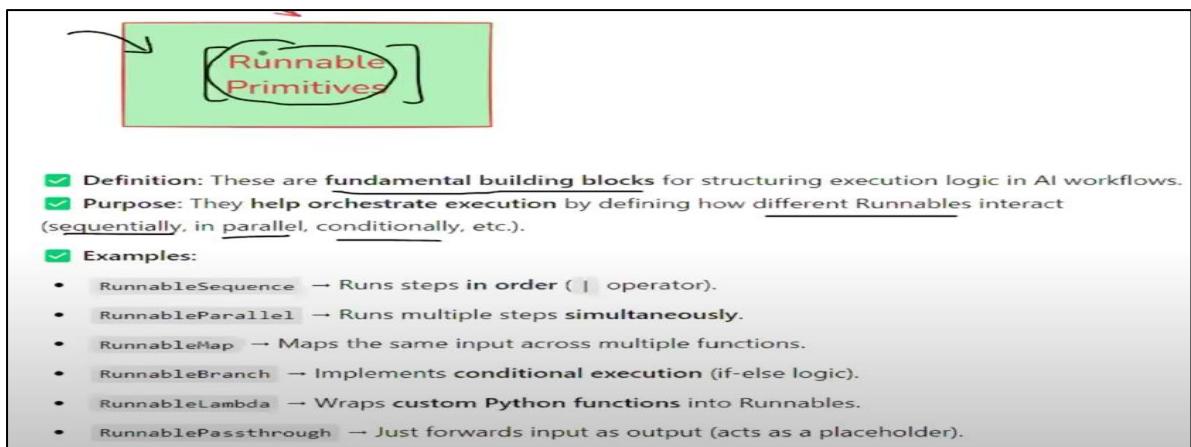
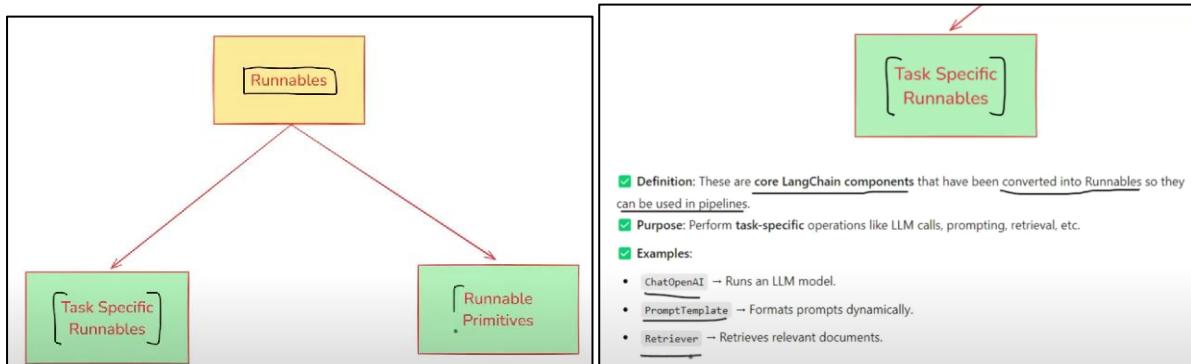
Examples of Built-in Runnables

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

- RunnableLambda (wraps a custom Python function)
-

Summary

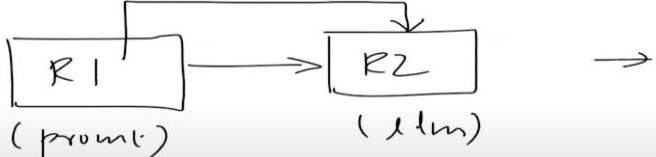
- Runnable is a **core concept** in LangChain's design to make LLM workflows modular, composable, and production-ready.
- It's like a standardized interface for any "step" in a processing pipeline.



1. 1. RunnableSequence

20 March 2025 12:10

RunnableSequence is a sequential chain of runnables in LangChain that executes each step one after another, passing the output of one step as the input to the next. It is useful when you need to compose multiple runnables together in a structured workflow.



1. RunnableSequence is a sequential chain of runnables in LangChain that executes each step one after another, passing the output of one step as the input to the next. It is useful when you need to compose multiple runnables together in a structured workflow.

2. RunnableParallel is a runnable primitive that allows multiple runnables to execute in parallel. Each runnable receives the same input and processes it independently, producing a dictionary of outputs.

4. RunnableLambda

20 March 2025 23:18



RunnableLambda is a runnable primitive that allows you to apply custom Python functions within an AI pipeline.

It acts as a middleware between different AI components, enabling preprocessing, transformation, API calls, filtering, and post-processing in a LangChain workflow.

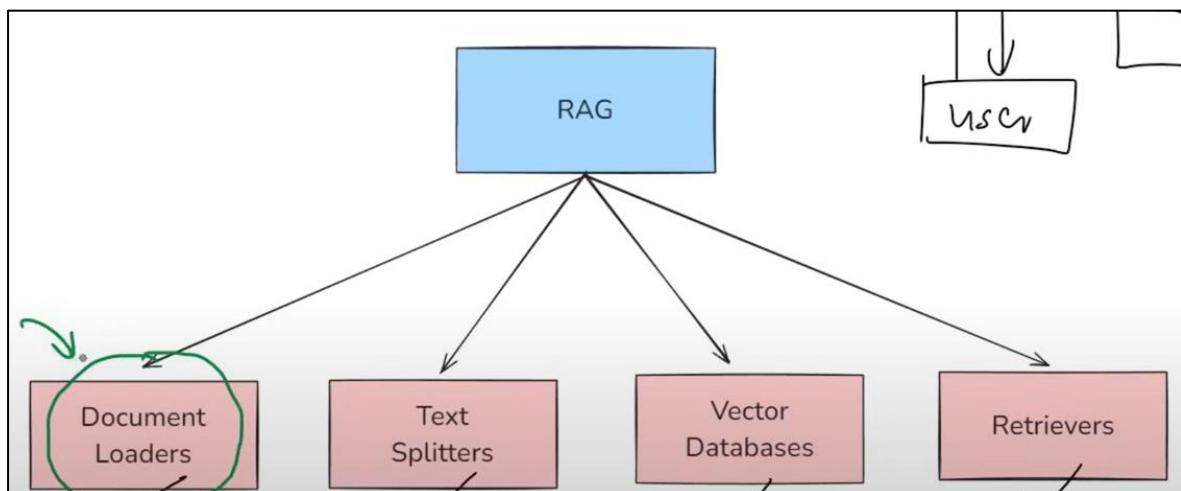
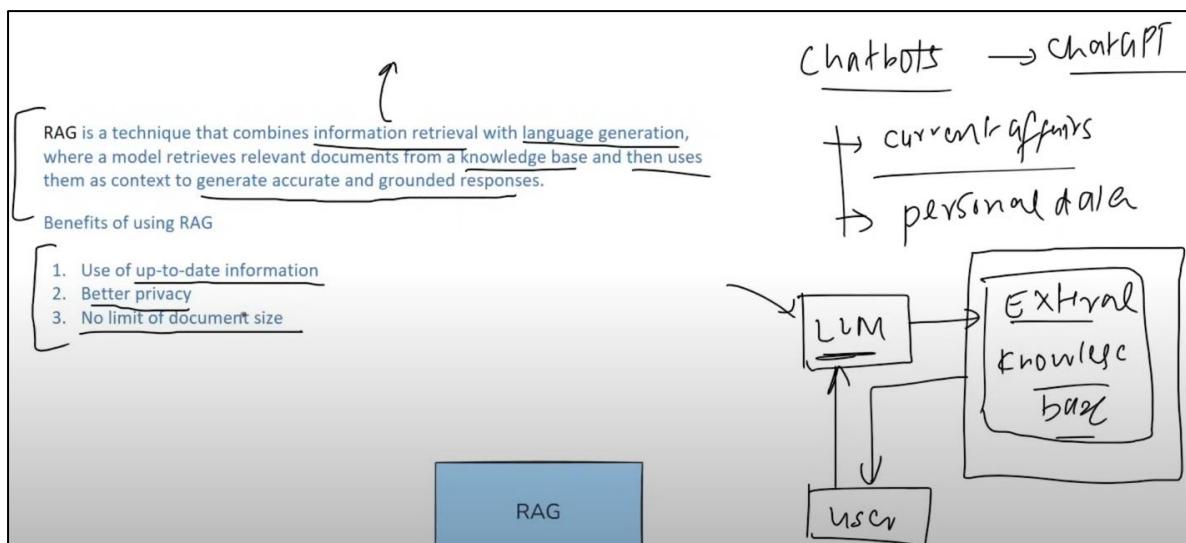
3. RunnableLambda ki help sei hum python ke kisi bhi simple function ko runnable me convert kr skte hai.

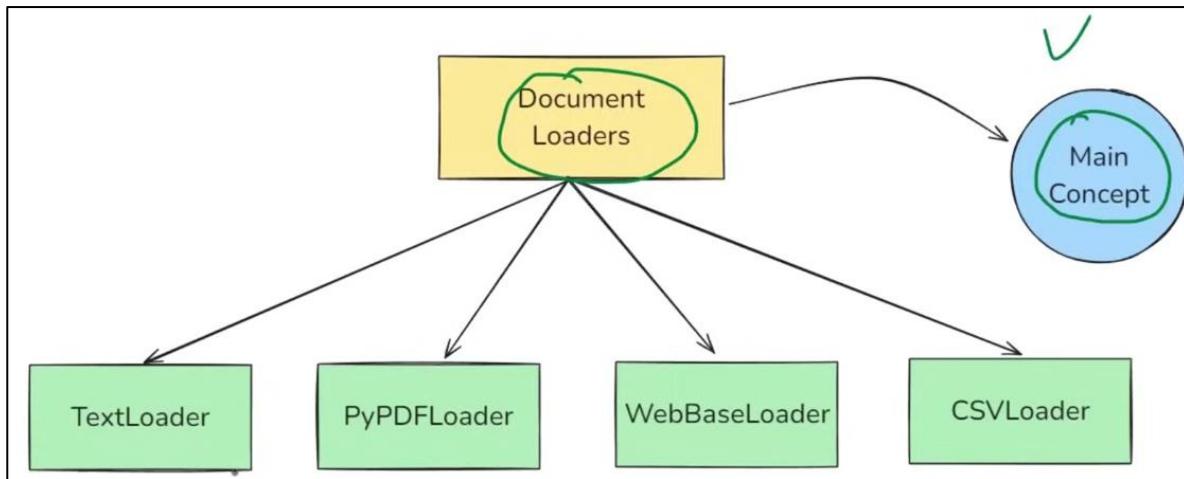
RAG: stands for **Retrieval-Augmented Generation**. It's a technique that combines two key components:

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.





Document Loaders in LangChain

27 March 2025 16:20

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", ...}
)
  
```

1. Text Loader

TextLoader

27 March 2025 16:50

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

2. PyPDFLoader

PyPDFLoader

27 March 2025 17:50

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.

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

DirectoryLoader

27 March 2025 18:11

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()

Load vs Lazy load → 500 pdf → generator of docs →

Eager loading

- ✓ 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.

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

4. WebBaseLoader

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.

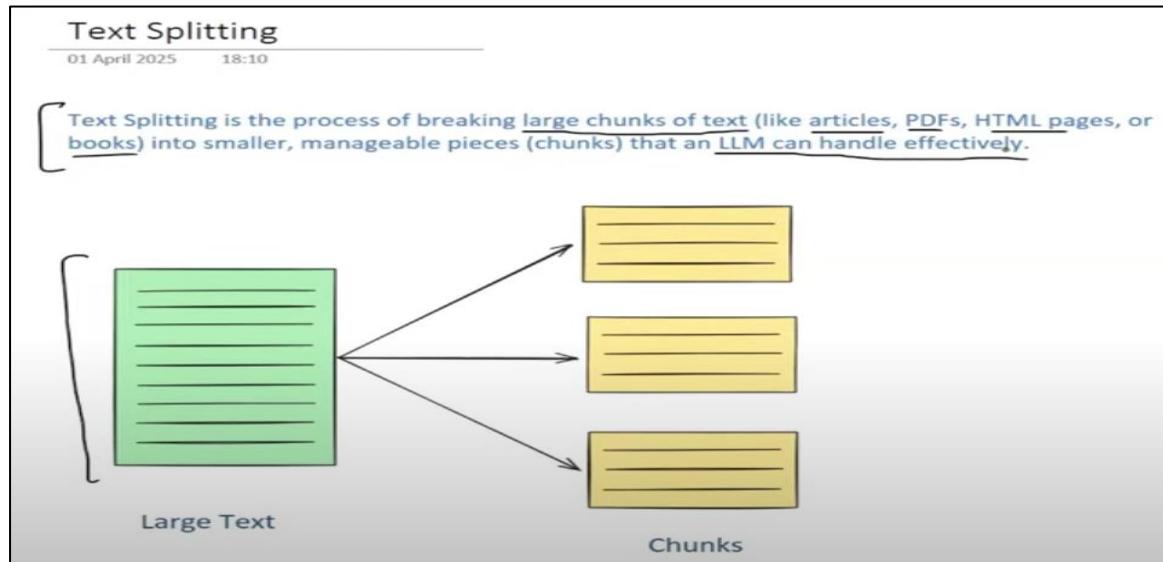
When to Use:

- 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).

Text Splitters in Langchain



Why Text Splitting is Better?

Chunks

- Overcoming model limitations: Many embedding models and language models have maximum input size constraints. Splitting allows us to process documents that would otherwise exceed these limits.
- Downstream tasks - Text Splitting improves nearly every LLM powered task

Task	Why Splitting Helps
Embedding	Short chunks yield more accurate vectors
Semantic Search	Search results point to focused info, not noise
Summarization	Prevents hallucination and topic drift

- Optimizing computational resources: Working with smaller chunks of text can be more memory-efficient and allow for better parallelization of processing tasks.

LLM → context length
 $(50K)$ tokens → words
 ↗ af → summarization
 ↘ ll > embedding

1. Length Based Text Splitting:

1. Length Based Text Splitting

01 April 2025 18:10

Space exploration has led to incredible scientific discoveries. From landing on the Moon to exploring Mars, humanity continues to push the boundaries of what's possible beyond our planet.

These missions have not only expanded our knowledge of the universe but have also contributed to advancements in technology here on Earth. Satellite communications, GPS, and even certain medical imaging techniques trace their roots back to innovations driven by space programs.

chunks → size chunk → 100 charach.

tokens

Space exploration has led to incredible scientific discoveries. From landing on the Moon to exploring Mars. → c1

g Mars, humanity continues to push the boundaries of what's possible beyond our planet. These missi) c2

ons have not only expanded our knowledge of the universe but have also contributed to advancements in c3

n technology here on Earth. Satellite communications, GPS, and even certain medical imaging techniqu c4

es trace their roots back to innovations driven by space programs. c5

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 thaek hum bache skte hai

The screenshot shows the 'Character Splitter' settings with a 'Chunk Size' of 100 and a 'Chunk Overlap' of 22. The total characters are 578, resulting in 6 chunks with an average chunk size of 96.3. The text input is a paragraph about space exploration, and the output shows the text split into 6 chunks with some overlap.

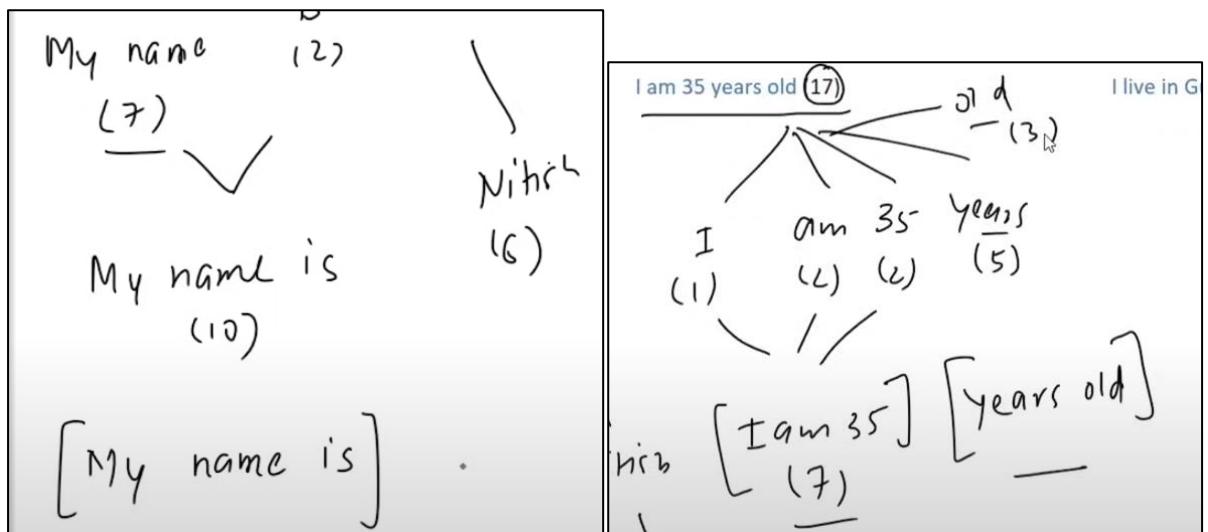
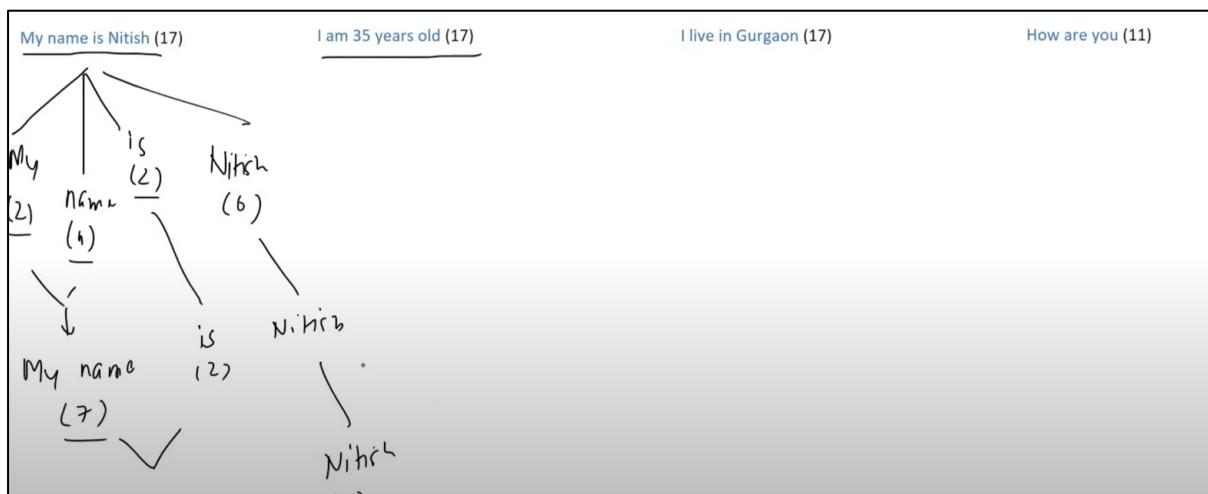
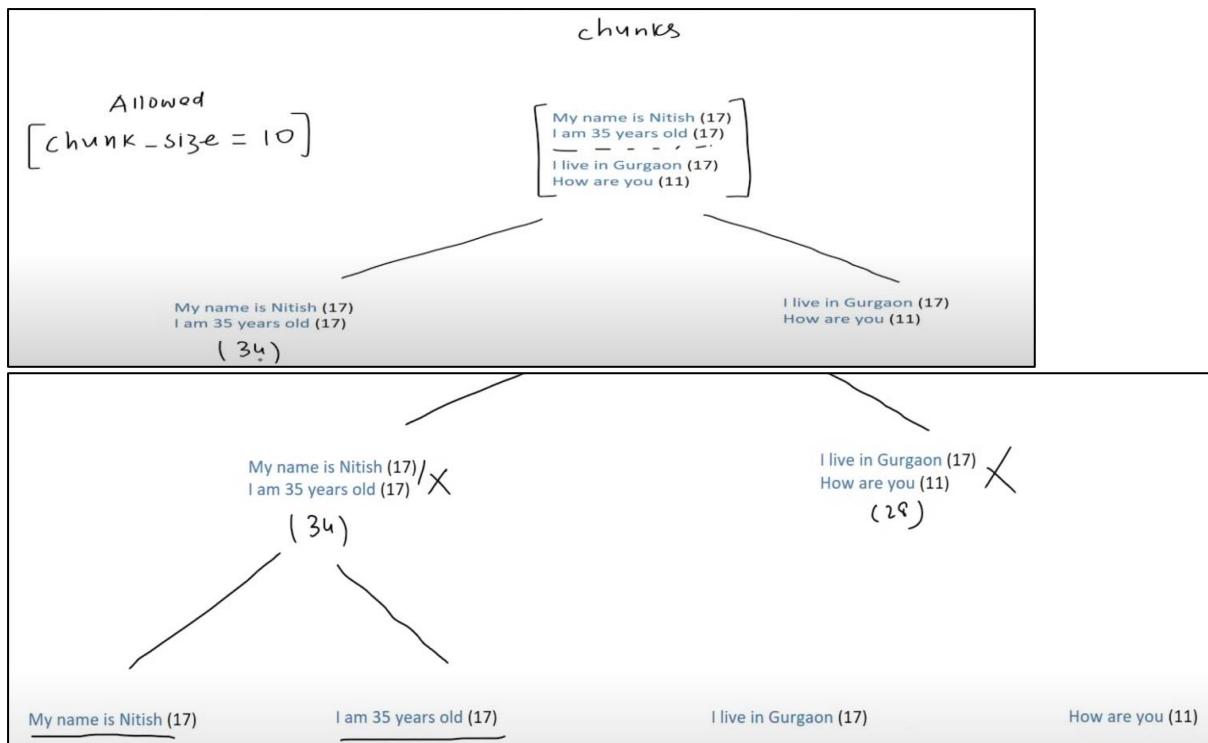
Tool to test different splitter: <https://chunkviz.up.railway.app/>

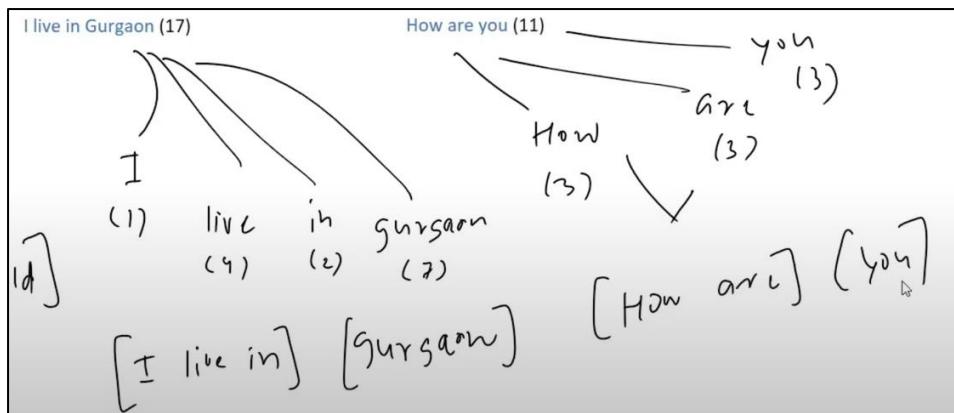
2. **Text Structure based Splitter** : This is the second splitter jisme algo. Neeche wala structure follow krta hai text ko split krne ke liye, pehle **paragraph** ke bases pe split krega agr chunk size chota hua toh phir **line** ke bases pe split krega, agr phir bhi chunk size chota ho, thab word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

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

The diagram shows a text input: "My name is Nitish I am 35 years old I live in Gurgaon How are you". To its right, a bracket labeled "structure" groups the first two sentences. Below the input, arrows point from specific characters to labels: '\n\n' points to "para", ',\n,' points to "line", '-' points to "word", and '''' points to "charact.". The word "charact." has an arrow pointing to a character in the input text.

Example of Text Structure based Splitter:





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),

3. Document-Structured Based

01 April 2025 18:11

```
# Project Name: Smart Student Tracker
A simple Python-based project to manage and track student data,
...
## Features
- Add new students with relevant info
- View student details
- Check if a student is passing
- Easily extendable class-based design
...

```

code →

```
class Student:
    def __init__(self, name, age, grade):
        self.name = name
        self.age = age
        self.grade = grade # Grade is a float (Like 8.5 or 9.2)

    def get_details(self):
        return f"Name: {self.name}, Age: {self.age}, Grade: {self.grade}"

    def is_passing(self):
        return self.grade >= 6.0

# Example usage
```

```
# First, try to split along class definitions
"\nclass",
"\ndef",
"\n\tdef",
# Now split by the normal type of lines
"\n\n",
"\n",
" ",
"" ,
```

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

4. Semantic Meaning Based

01 April 2025 18:11

2 chunks

(len) (structure)

2 para X

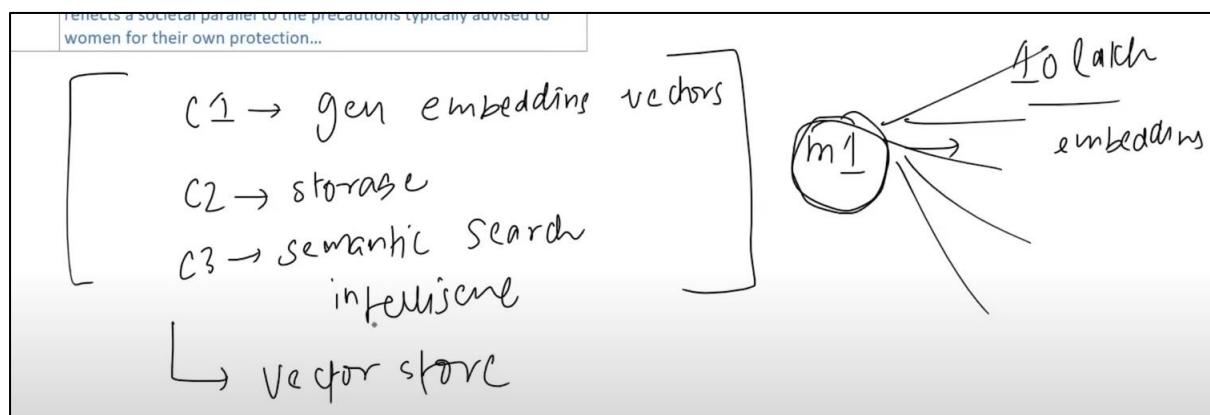
Farmers were working hard in the fields, preparing the soil and planting seeds for the next season. The sun was bright, and the air smelled of earth and fresh grass. The Indian Premier League (IPL) is the biggest cricket league in the world. People all over the world watch the matches and cheer for their 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.

4.

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**.

What are Vector Stores
05 April 2025 17:38

A vector store is a system designed to store and retrieve data represented as numerical vectors.

Key Features

1. Storage - Ensures that vectors and their associated metadata are retained, whether in-memory for quick lookups or on-disk for durability and large-scale use.
2. Similarity Search - Helps retrieve the vectors most similar to a query vector.
3. Indexing - Provide a data structure or method that enables fast similarity searches on high-dimensional vectors (e.g., approximate nearest neighbor lookups).
4. CRUD Operations - Manage the lifecycle of data—adding new vectors, reading them, updating existing entries, removing outdated vectors.

Use-cases

1. Semantic Search ✓
2. RAG → ✓
3. Recommender Systems ✓
4. Image/Multimedia search

RAM

in-memory

on-disk

vector store

Harddrive

Vector Store vs Vector DB

Vector Store Vs Vector Database

05 April 2025 17:40

- Vector Store**
 - Typically refers to a lightweight library or service that focuses on storing vectors (embeddings) and performing similarity search.
 - May not include many traditional database features like transactions, rich query languages, or role-based access control.
 - Ideal for prototyping, smaller-scale applications
 - Examples: FAISS (where you store vectors and can query them by similarity, but you handle persistence and scaling separately).
- Vector Database**
 - A full-fledged database system designed to store and query vectors.
 - Offers additional "database-like" features:
 - Distributed architecture for horizontal scaling
 - Durability and persistence (replication, backup/restore)
 - Metadata handling (schemas, filters)
 - Potential for ACID or near-ACID guarantees
 - Authentication/authorization and more advanced security
 - Geared for production environments with significant scaling, large datasets

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.

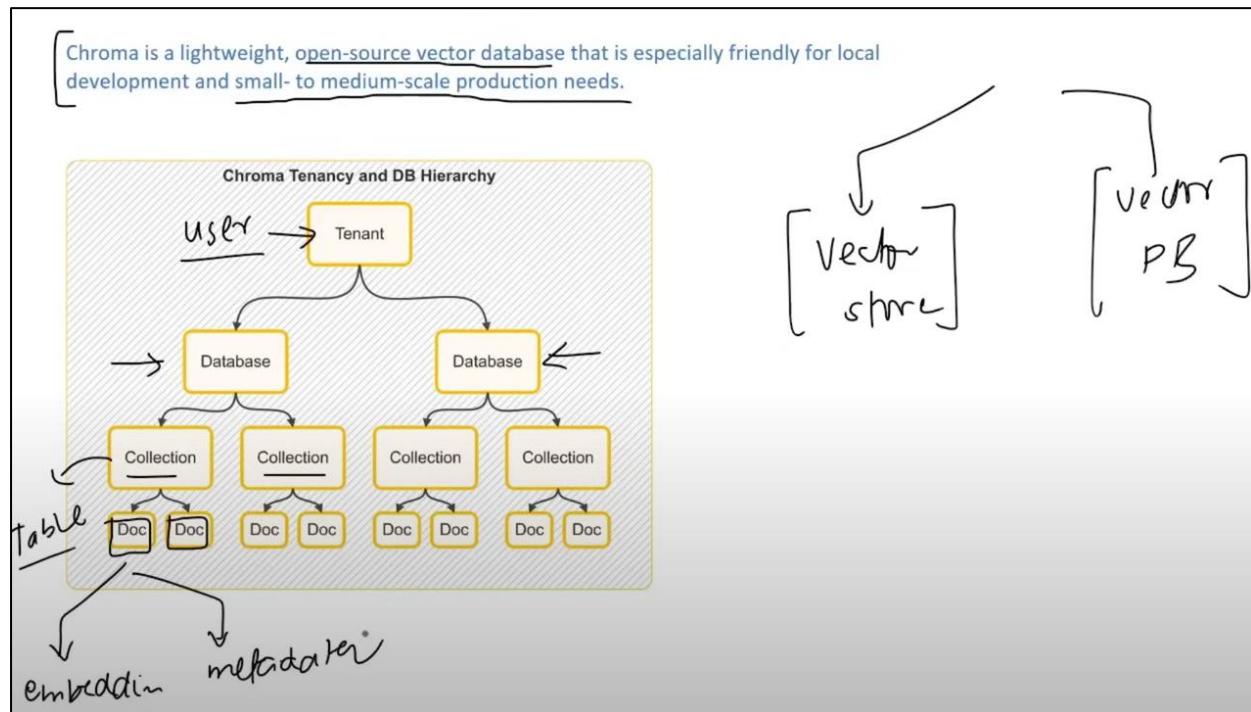
Vector Store voh system hote hai jisme STORAGE aur RETRIEVAL ka features hota hai and Vector DB voh hota hai jisme yeh doo feature toh hote hee hai, but lekin saath mei jo normal DB ke feature hote hai voh bhi isme hote hai. That's why all Vector DB are Vector Store , but all Vector Store are not Vector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

Vector Stores in LangChain

05 April 2025 17:41

- Supported Stores:** LangChain integrates with multiple vector stores (FAISS, Pinecone, Chroma, Qdrant, Weaviate, etc.), giving you flexibility in scale, features, and deployment.
- Common Interface:** A uniform Vector Store API lets you swap out one backend (e.g., FAISS) for another (e.g., Pinecone) with minimal code changes.
- Metadata Handling:** Most vector stores in LangChain allow you to attach metadata (e.g., timestamps, authors) to each document, enabling filter-based retrieval.

Chroma DB:



Retriever

What are Retrievers

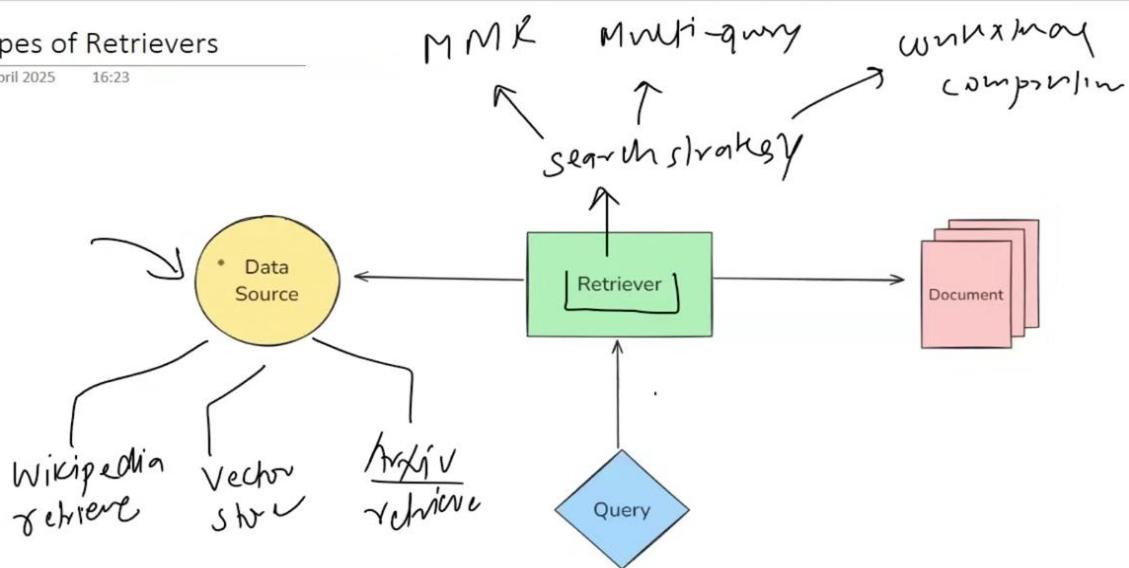
10 April 2025 07:56

A retriever is a component in LangChain that fetches relevant documents from a data source in response to a user's query.

- There are multiple types of retrievers
- All retrievers in LangChain are runnables

Types of Retrievers

10 April 2025 16:23



Wikipedia Retriever

Wikipedia Retriever

10 April 2025 16:23

A Wikipedia Retriever is a retriever that queries the Wikipedia API to fetch relevant content for a given query.

How It Works

1. You give it a query (e.g., "Albert Einstein")
2. It sends the query to Wikipedia's API
3. It retrieves the most relevant articles
4. It returns them as LangChain Document objects



Vector Retriever

Vector Store Retriever

10 April 2025 16:24

A Vector Store Retriever in LangChain is the most common type of retriever that lets you search and fetch documents from a vector store based on semantic similarity using vector embeddings.

⚙️ How It Works

1. You store your documents in a vector store (like FAISS, Chroma, Weaviate)
2. Each document is converted into a dense vector using an embedding model
3. When the user enters a query:
 - It's also turned into a vector
 - The retriever compares the query vector with the stored vectors
 - It retrieves the top-k most similar ones

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

MMR Retriever

Maximal Marginal Relevance (MMR)

10 April 2025 16:24

"How can we pick results that are not only relevant to the query but also different from each other?"

MMR is an information retrieval algorithm designed to reduce redundancy in the retrieved results while maintaining high relevance to the query.

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.

💡 Why MMR Retriever?

In regular similarity search, you may get documents that are:

- All very similar to each other
- Repeating the same info
- Lacking diverse perspectives

MMR Retriever avoids that by:

- Picking the most relevant document first
- Then picking the next most relevant and least similar to already selected docs
- And so on...

This helps especially in RAG pipelines where:

- You want your context window to contain diverse but still relevant information
- Especially useful when documents are semantically overlapping

Multi-Query Retriever

Multi-Query Retriever

10 April 2025 16:26

Sometimes a single query might not capture all the ways information is phrased in your documents.

For example:

Query: "How can I stay healthy?"

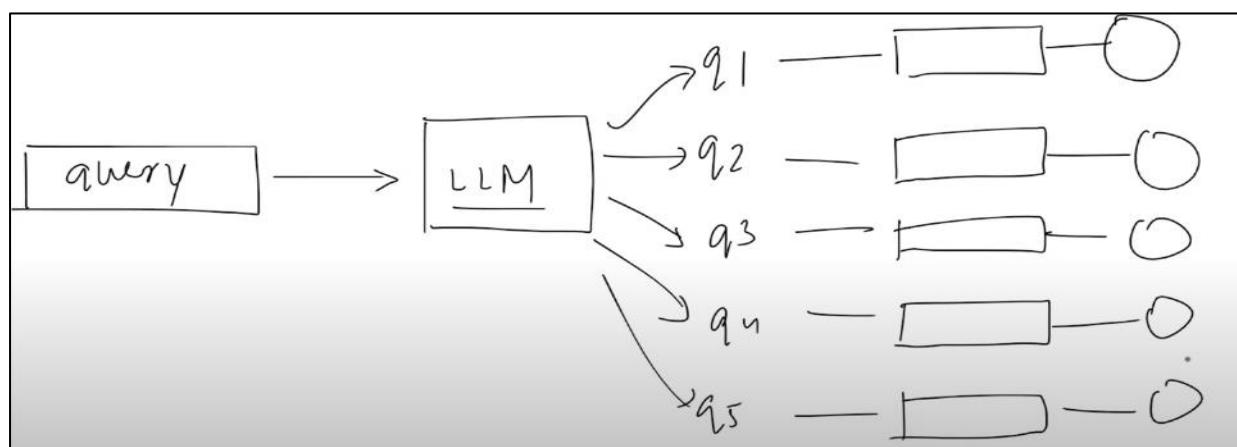
Could mean:

- What should I eat?
- How often should I exercise?
- How can I manage stress?

A simple similarity search might **miss documents** that talk about those things but don't use the word "healthy."

1. Takes your original query
2. Uses an LLM (e.g., GPT-3.5) to generate multiple semantically different versions of that query
3. Performs retrieval for each sub-query
4. Combines and deduplicates the results

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 its own document/output, so at last we remove the redundancy by applying MMR retriever.



Contextual Compression Retriever

Contextual Compression Retriever

10 April 2025 16:29

The Contextual Compression Retriever in LangChain is an advanced retriever that improves retrieval quality by compressing documents after retrieval — keeping only the relevant content based on the user's query.

? Query:

"What is photosynthesis?"

Retrieved Document (by a traditional retriever):

*"The Grand Canyon is a famous natural site.
Photosynthesis is how plants convert light into energy.
Many tourists visit every year."*

text splitter

result

✗ Problem:

- The retriever returns the entire paragraph
- Only one sentence is actually relevant to the query
- The rest is irrelevant noise that wastes context window and may confuse the LLM

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.

What Contextual Compression Retriever does:

Returns only the relevant part, e.g.

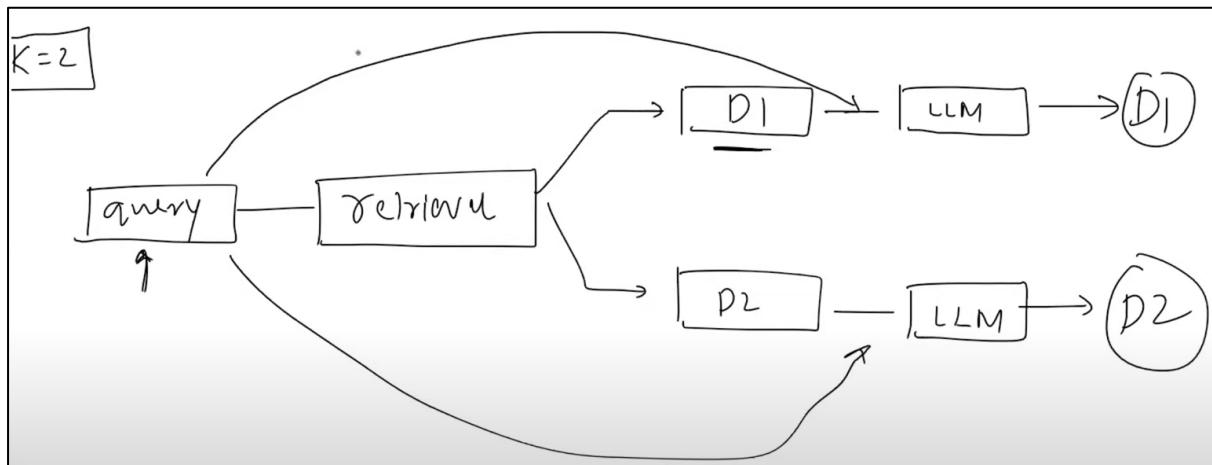
"Photosynthesis is how plants convert light into energy."

How It Works

1. Base Retriever (e.g., FAISS, Chroma) retrieves N documents.
2. A compressor (usually an LLM) is applied to each document.
3. The compressor keeps only the parts relevant to the query.
4. Irrelevant content is discarded.

When to Use

- Your documents are long and contain mixed information
- You want to reduce context length for LLMs
- You need to improve answer accuracy in RAG pipelines



<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.

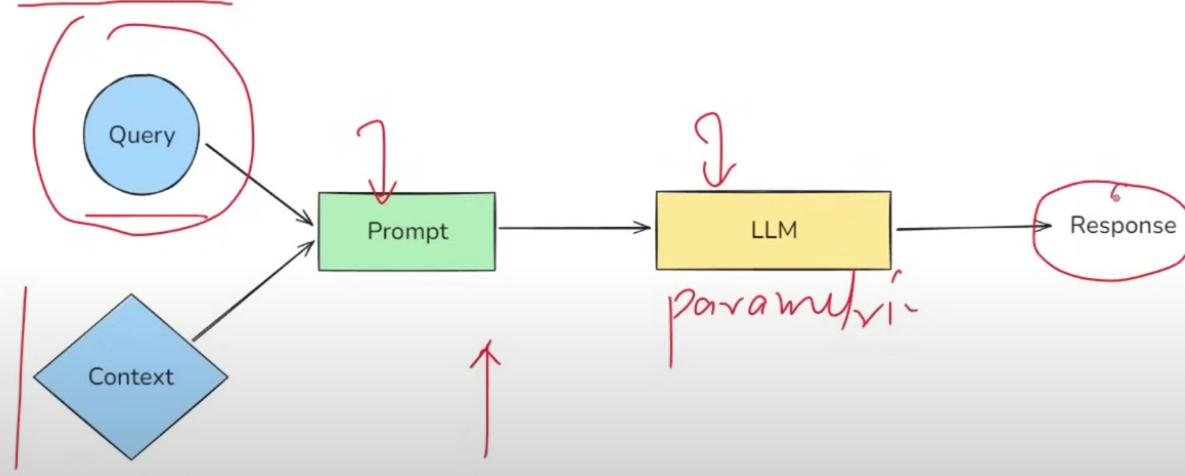
In-Context Learning: incontext learning agr simple words mei bataou toh essa prompt dena llm ko jisme eek doo short example ho apki query se related jo apki query ko solve krne mei help krega, so inhi short examples ki vegah sei apka llm apke query ko ache sei solve kr payega.

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 property

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.

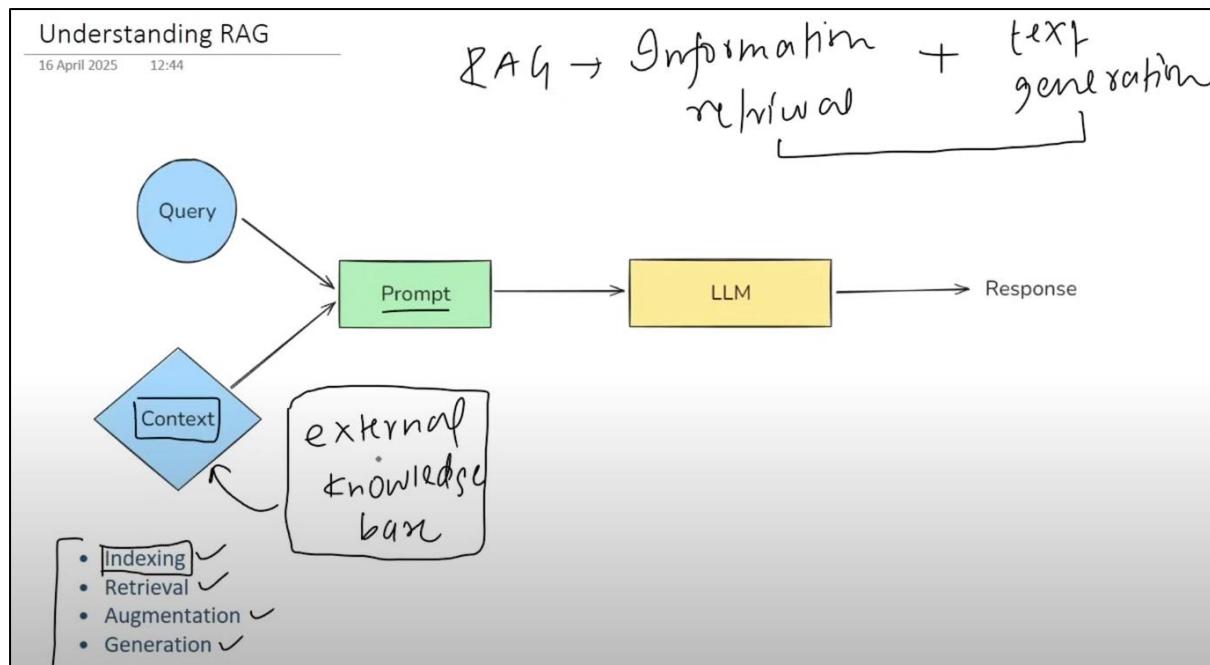


```

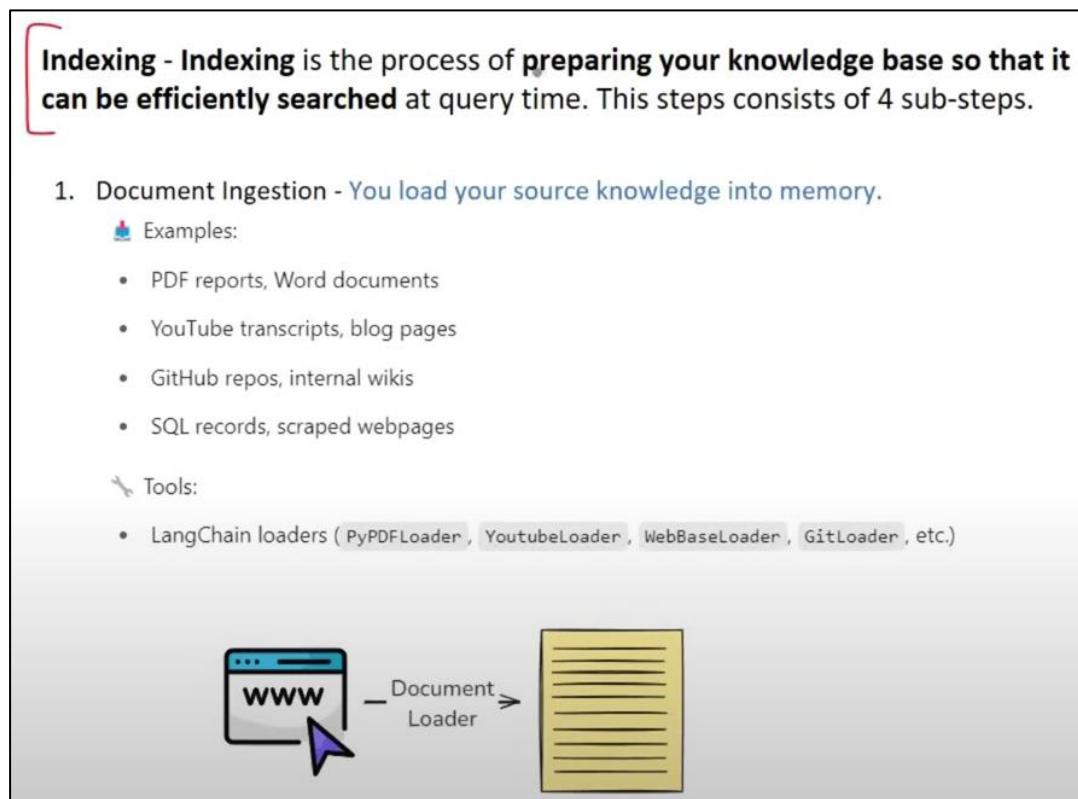
"""
You are a helpful assistant.
Answer the question ONLY from the provided context.
If the context is insufficient, just say you don't know.
  
```

```

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



Full work Flow of Implementing RAG:



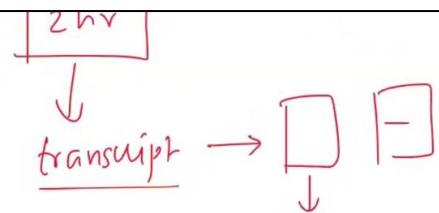
2. Text Chunking - Break large documents into small, semantically meaningful chunks

💡 Why chunk?

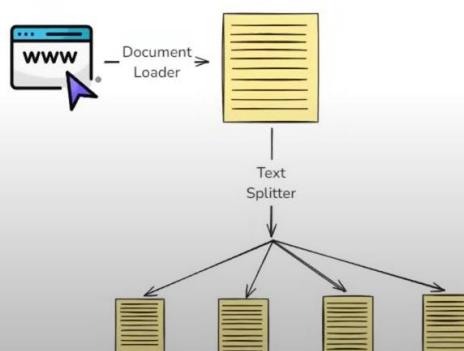
- LLMs have context limits (e.g., 4K–32K tokens)
- Smaller chunks are more focused → better semantic search

🔧 Tools:

- `RecursiveCharacterTextSplitter`, `MarkdownHeaderTextSplitter`, `SemanticChunker`



2.



3. Embedding Generation - Convert each chunk into a **dense vector** (embedding) that captures its meaning.

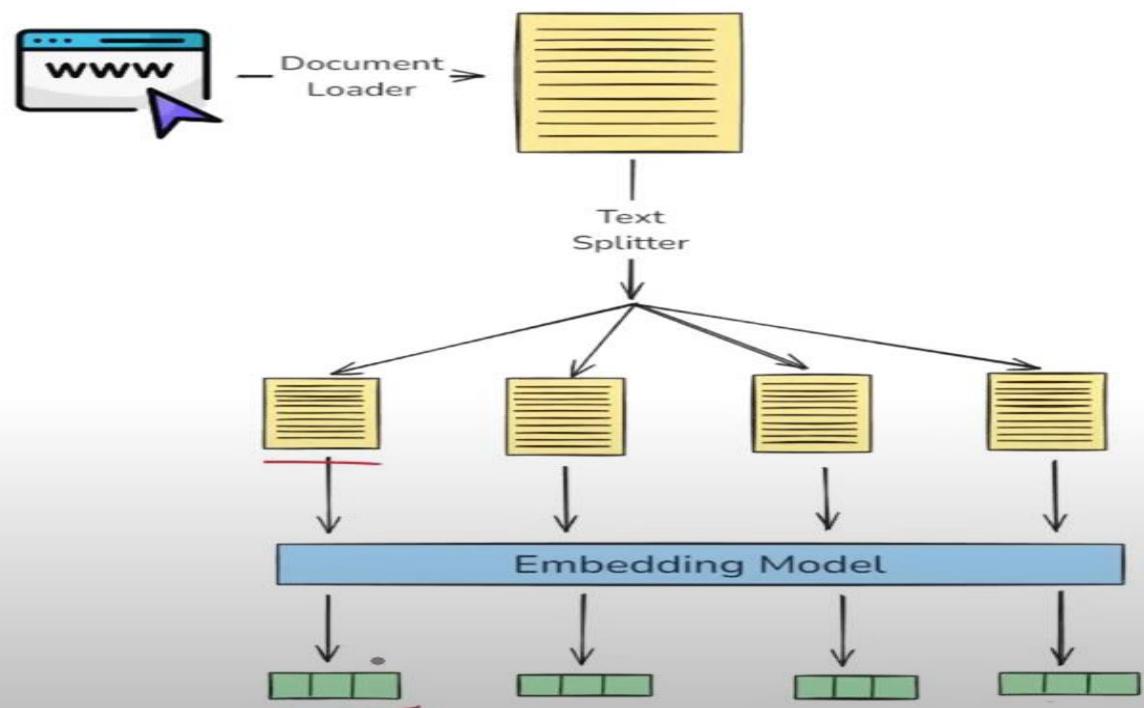
🔍 Why embeddings?

- Similar ideas land close together in vector space
- Allows fast, fuzzy semantic search

🔧 Tools:

- `OpenAIEmbeddings`, `SentenceTransformerEmbeddings`, `InstructorEmbeddings`, etc.

3.

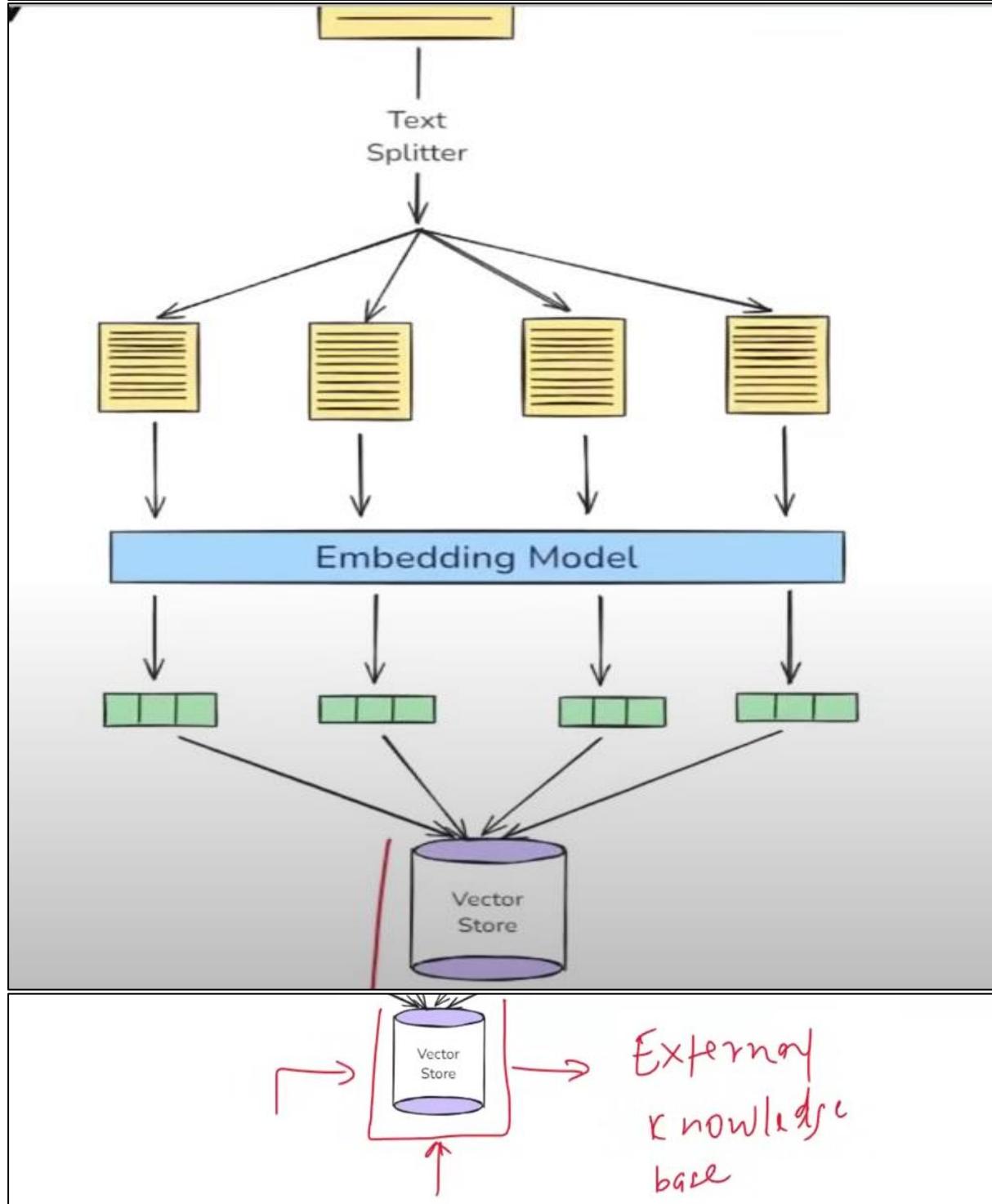


4. Storage in a Vector Store - Store the vectors along with the original chunk text + metadata in a vector database.

↳ Vector DB options:

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

4.

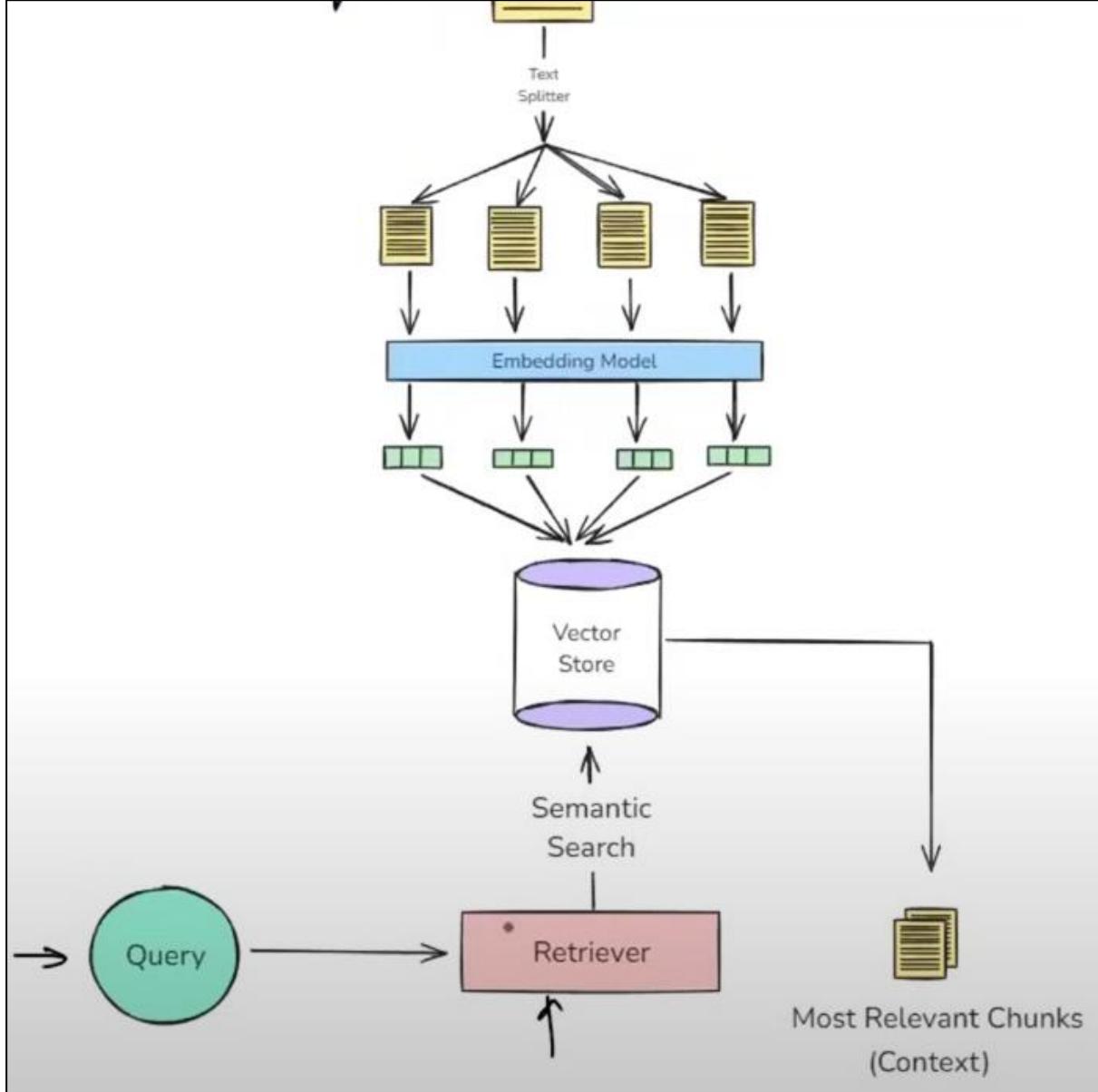


Retrieval - Retrieval is the *real-time* process of finding the most relevant pieces of information from a pre-built index (created during indexing) based on the user's question.

It's like asking:

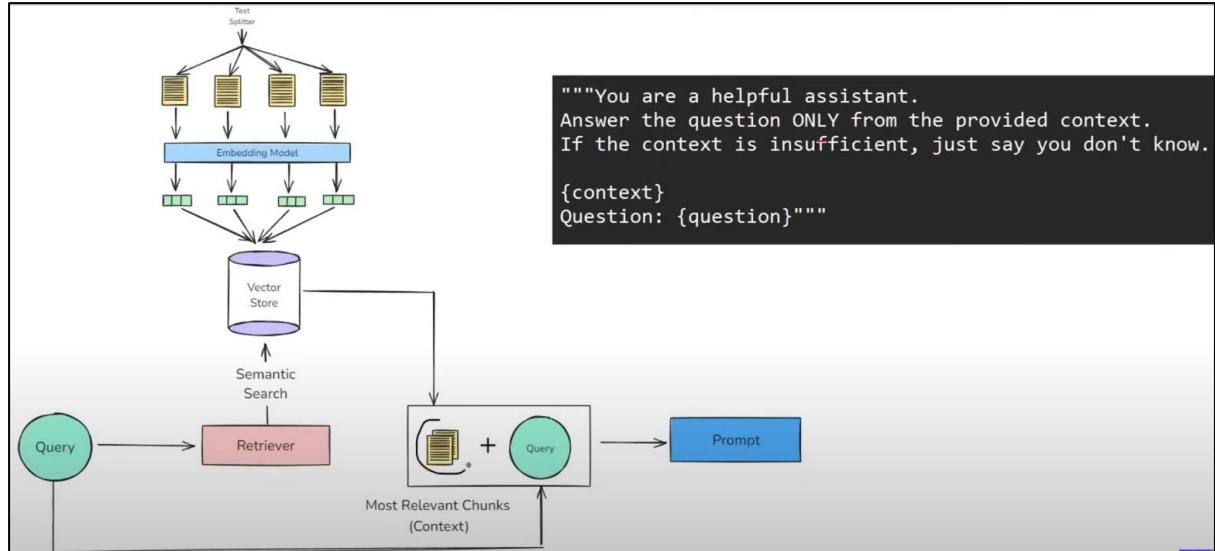
"From all the knowledge I have, which 3–5 chunks are most helpful to answer this query?"

5.



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.

6.



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.

7.

Architecture of RAG:

