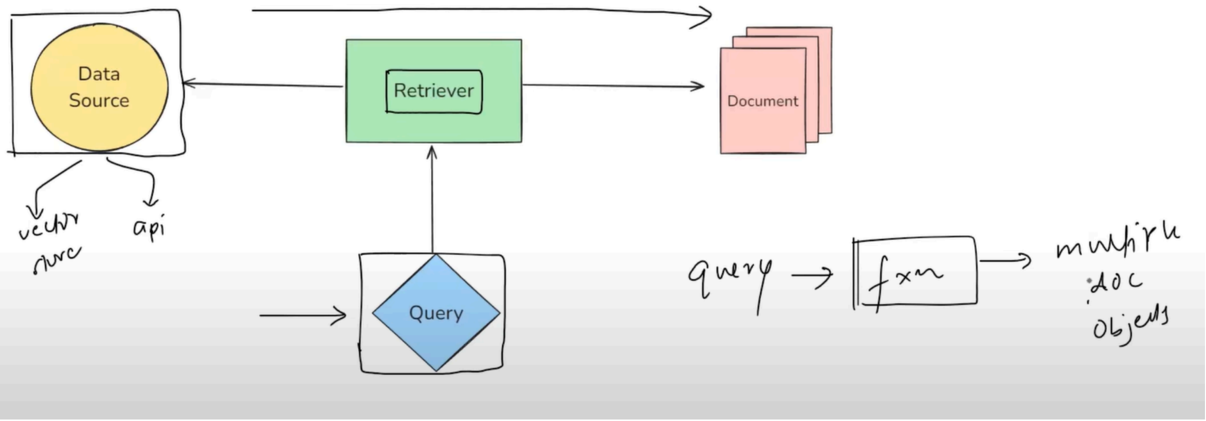
**A Retriever is a component of Langchain that fetches relevant documents from a data source in response to users query**



- **Data Source : api, vector store**
- **Retrievers receives a user query and enters the data source and scans the data source**
- **Finds out the most similar document and fetches them in form of document object**

**There are multiple types of retrievers**

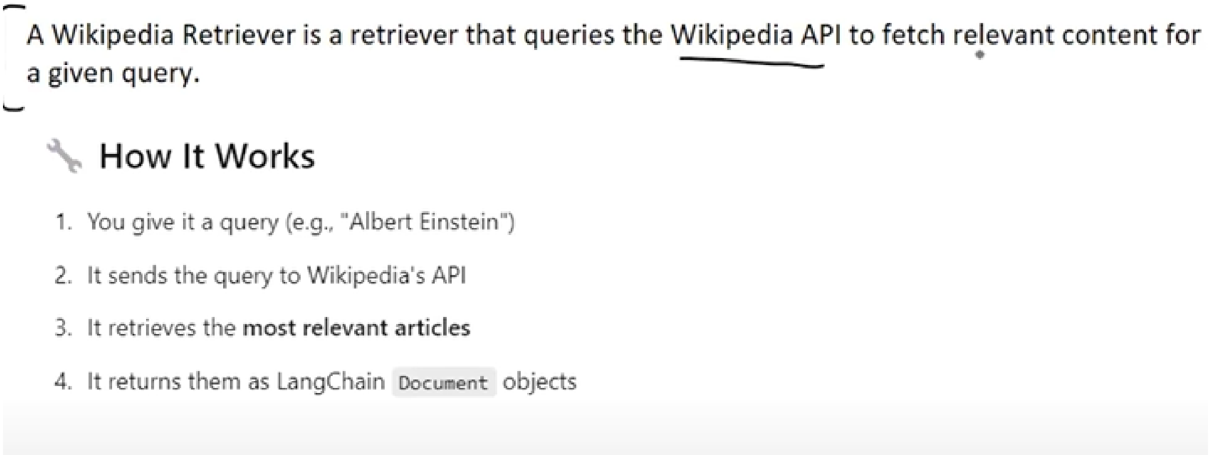
1. **Based on Data Source**

a. **Wikipedia Retrievers:** searches on wikipedia by taking the user query
b. **Vector Stores**: data is in vs and find out relevant document in the vs
c. **Archive Retrievers:** scans the research paper on archive website

2. **Based on Search Strategy of Retrievers:**

a. **Maximum Marginal Relevance**
b. **Multi query Retriever**
c. **Contextual Compression Retriever**

# 1. Wikipedia Retriever:

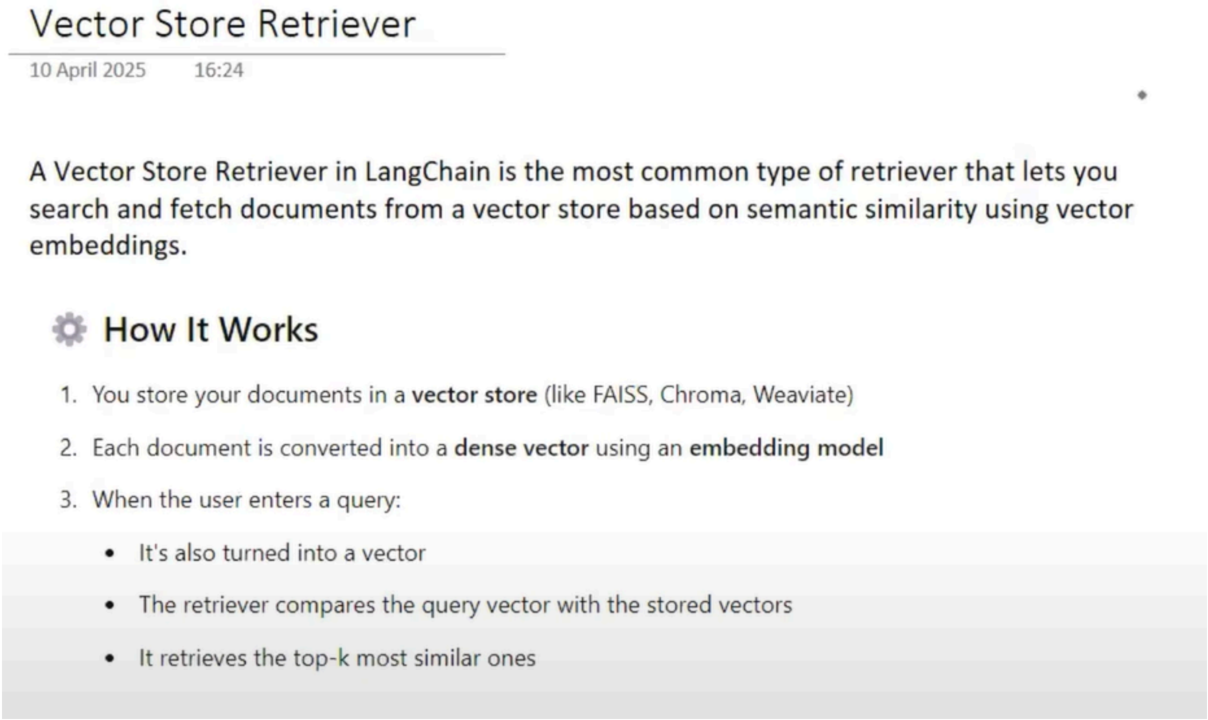


A Wikipedia Retriever is a retriever that queries the Wikipedia API to fetch relevant content for a given query.

### 🔧 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

# 2. Vector Store Retriever:



## Vector Store Retriever

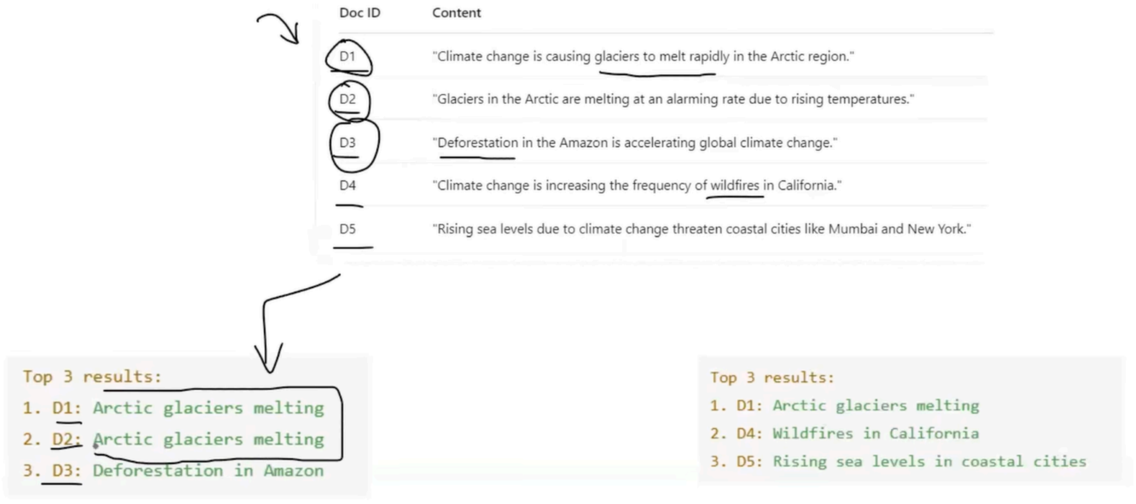
10 April 2025 16:24

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

# 3. Maximal Marginal Retriever:

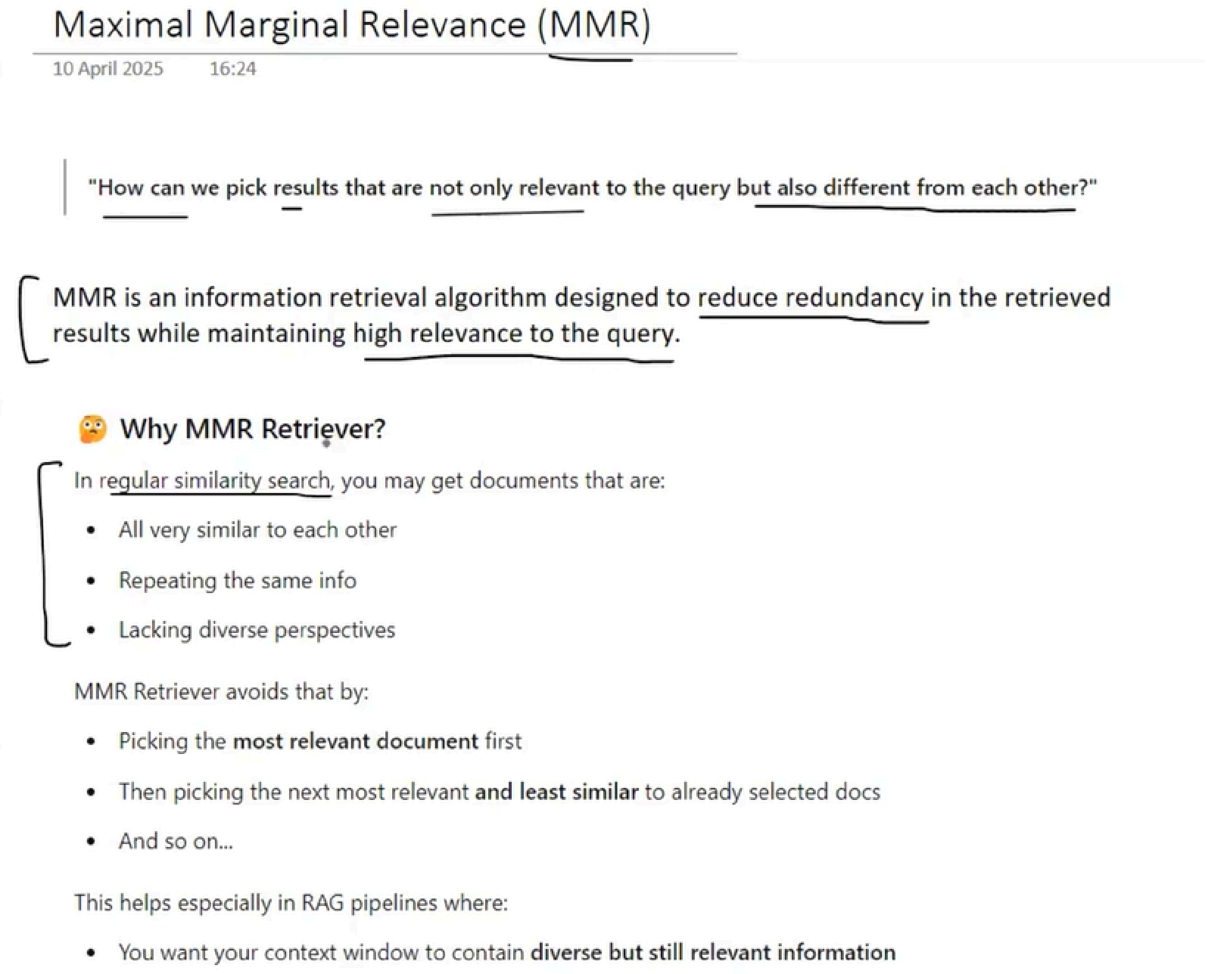


| Doc ID | Content |
|---|---|
| D1 | "Climate change is causing glaciers to melt rapidly in the Arctic region." |
| D2 | "Glaciers in the Arctic are melting at an alarming rate due to rising temperatures." |
| D3 | "Deforestation in the Amazon is accelerating global climate change." |
| D4 | "Climate change is increasing the frequency of wildfires in California." |
| D5 | "Rising sea levels due to climate change threaten coastal cities like Mumbai and New York." |

```
Top 3 results:
1. D1: Arctic glaciers melting
2. D2: Arctic glaciers melting
3. D3: Deforestation in Amazon
```

```
Top 3 results:
1. D1: Arctic glaciers melting
2. D4: Wildfires in California
3. D5: Rising sea levels in coastal cities
```

**Normal Retriever** **Ideally**



## 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.

### 😳 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:

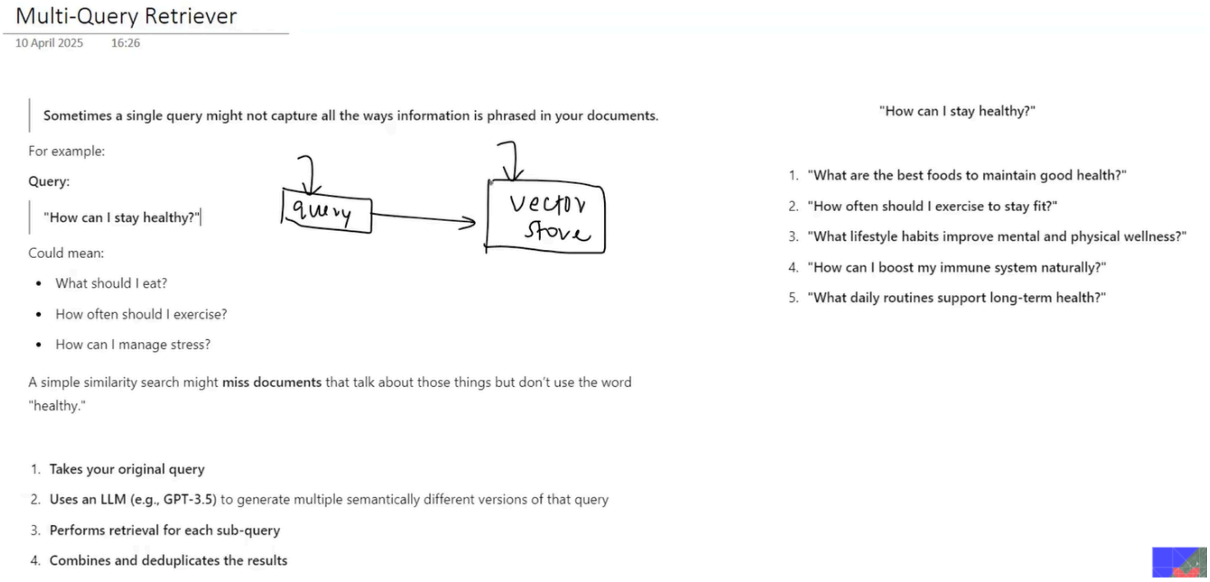
- Picking the **most relevant document** first
- Then picking the next most relevant **and least similar** to already selected docs
- And so on...

This helps especially in RAG pipelines where:

- You want your context window to contain **diverse but still relevant information**

# 4. Multi-Query Retriever:

- Handles ambiguous queries
- First query is sent to llm
- LLM generates multiple related queries
- And they are individually given to their retrievers and fetch relevant document based on their individual ques
- Merged finally



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."

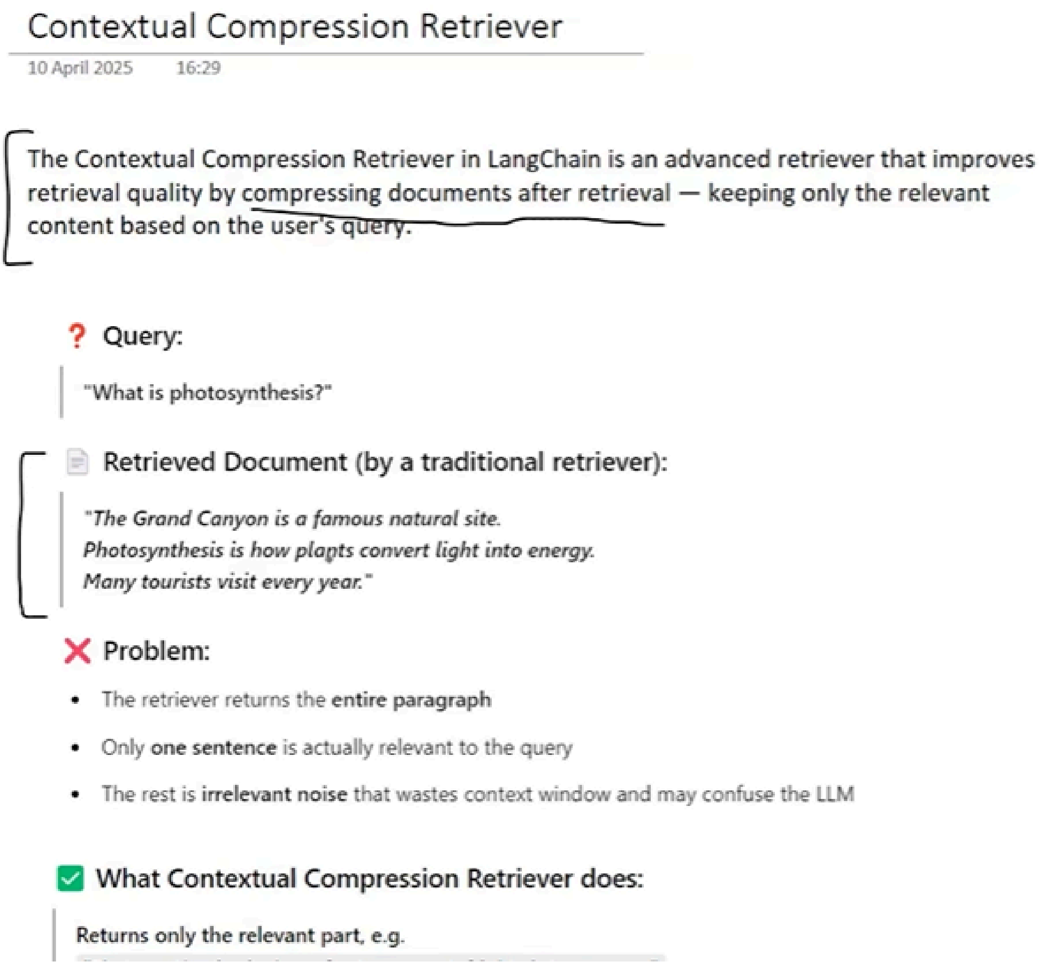
1. **Takes your original query**
2. **Uses an LLM (e.g., GPT-3.5)** to generate multiple semantically different versions of that query
3. **Performs retrieval for each sub-query**
4. **Combines and deduplicates the results**

"How can I stay healthy?"

1. "What are the best foods to maintain good health?"
2. "How often should I exercise to stay fit?"
3. "What lifestyle habits improve mental and physical wellness?"
4. "How can I boost my immune system naturally?"
5. "What daily routines support long-term health?"

# 5. Contextual Compression Retriever:

- Keeps only relevant content based on user query
- Vector store : grand canyon, photosynthesis, tourists
- Diff context
- Query : Photosynthesis
- But in Vector Store : photosynthesis diff content is also there
- User might get ans about grand canyon—-- bad ux
- CC Retriever : gives only about photosynthesis



## 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):

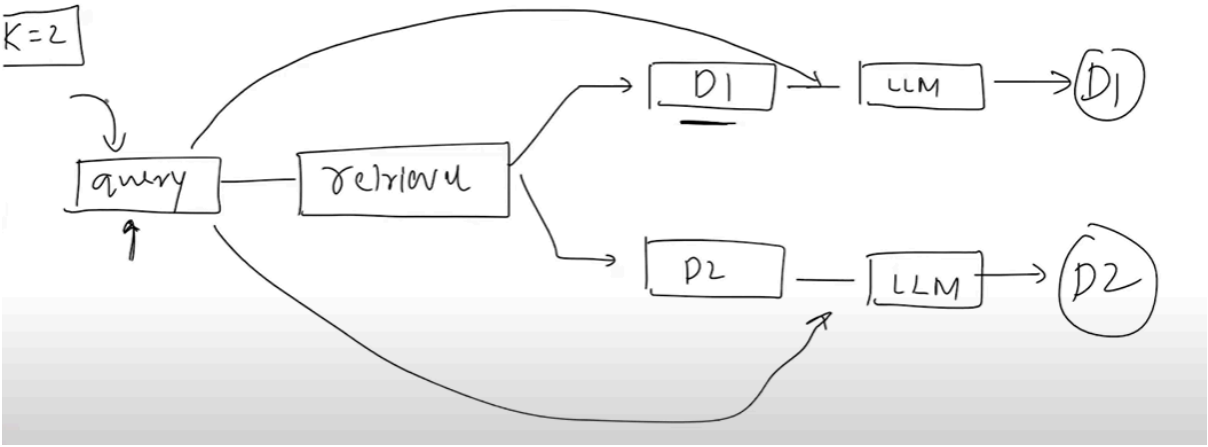
> *"The Grand Canyon is a famous natural site.*
> *Photosynthesis is how plants convert light into energy.*
> *Many tourists visit every year."*

### ❌ 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

### ✅ What Contextual Compression Retriever does:

> Returns only the relevant part, e.g.



Based on prompt trim the unwanted part. Here llm is the compressor