Introduction to Large Language Models and LangChain

Overview

Large language models are able to answer questions on topics on which they are trained. But they are not able to answer questions on our personal data or a company's proprietary documents or articles written after the LLM was trained. It will be really great if we are able to have some conversations with our own documents and answer questions from these documents using an LLM.

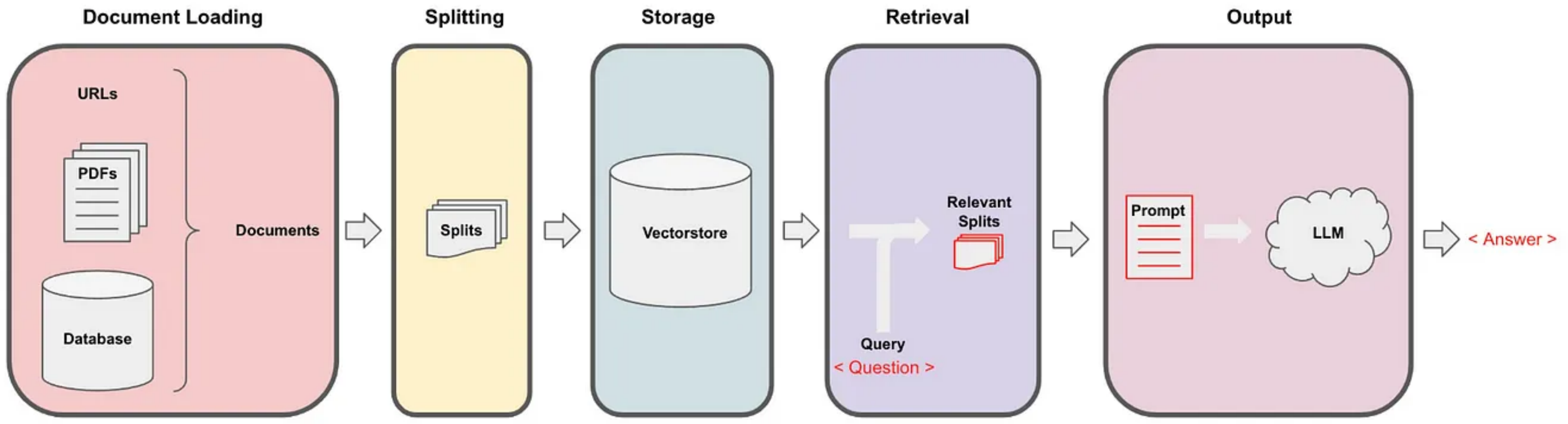
LangChain is an open-source developer framework for building LLM applications. In this article, we will focus on a specific use case of LangChain i.e. how to use LangChain to chat with own data.

Topics Covered

1. Document Loading
2. Document Splitting
3. Vector Store and Embeddings
4. Retrieval
5. Question Answering

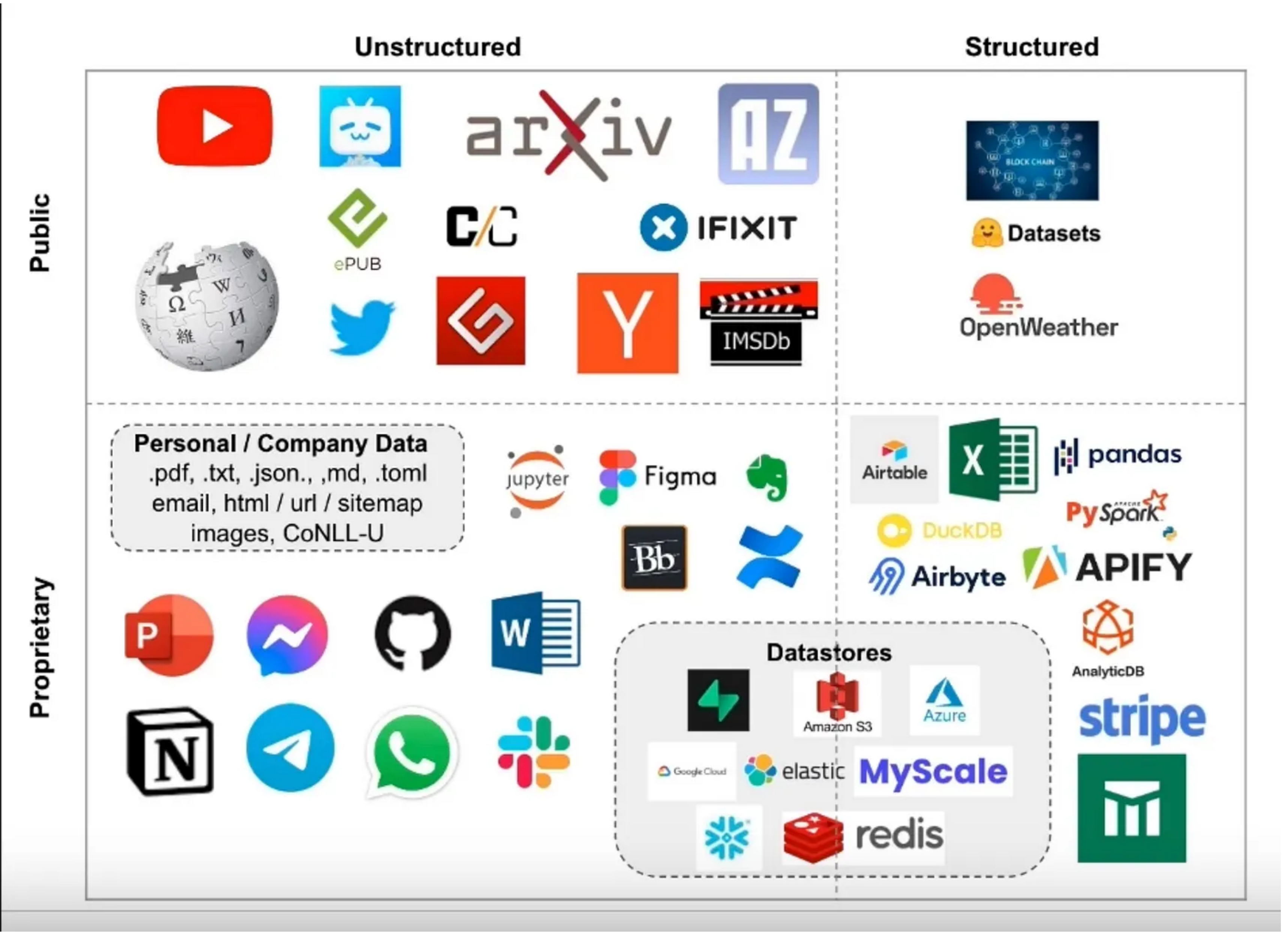
Document Loading

Retrieval Augmented Generation (RAG)



In retrieval augmented generation (RAG) framework, an LLM retrieves contextual documents from an external dataset as part of its execution.

LangChain's Document Loaders



We use LangChain’s document loaders for this purpose. Document loaders deal with the specifics of accessing and converting data from a variety of different formats and sources into a standardized format.

PyPDF DataLoader

Code Example

```python
from langchain.document_loaders import PyPDFLoader
loader = PyPDFLoader("docs/document.pdf")
pages = loader.load()
print(len(pages))
print(pages[0].page_content[0:500])
print(pages[0].metadta)
```

Youtube DataLoader

Code Example

```
from langchain.document_loaders.generic import GenericLoader
from langchain.document_loaders.parsers import OpenAIWhisperParser
from langchain.document_loaders.blob_loaders.youtube_audio import YoutubeAudioLoader

url="https://www.youtube.com/watch?v=jGwO_UgTS7I"
save_dir="docs/youtube/"
loader = GenericLoader(YoutubeAudioLoader([url],save_dir), OpenAIWhisperParser())
docs = loader.load()
print(docs[0].page_content[0:500])
```

WebBaseLoader

Code Example

```python
from langchain.document_loaders import WebBaseLoader
loader = WebBaseLoader("https://github.com/basecamp/handbook/blob/master/37signals-is-you.md")
docs = loader.load()
print(docs[0].page_content[:500])
```

Here, we need to do post-processing on the above output as there is a lot of white space followed by some text.

Chains

Chain

- Chains typically combine Large Language Models (LLMs) with prompts.
- They allow us to perform a series of operations on text or data.

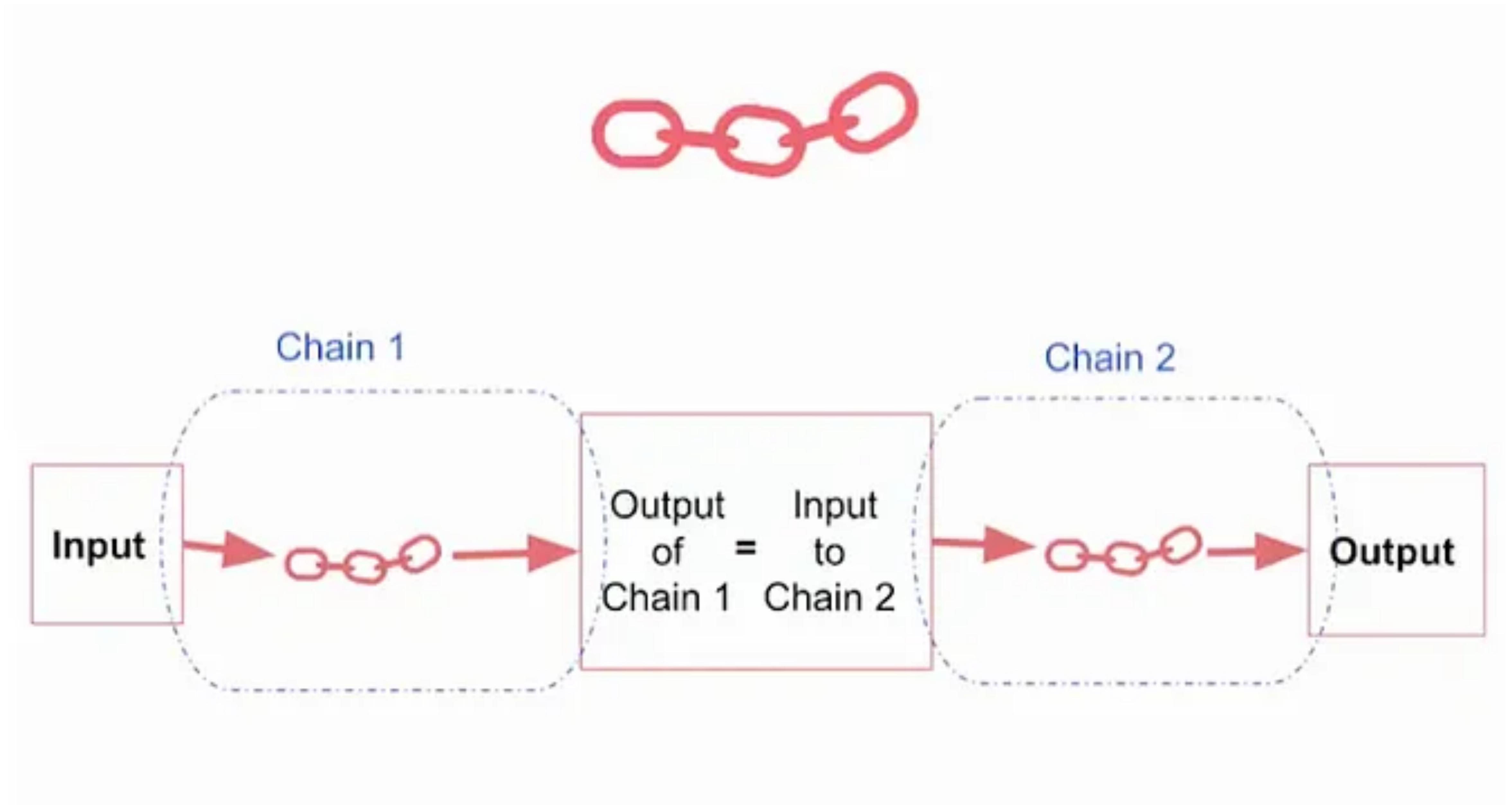
Use Cases

- Aimed for processing pipelines on data
  - Sequential processing line,
  - Branching (Router)

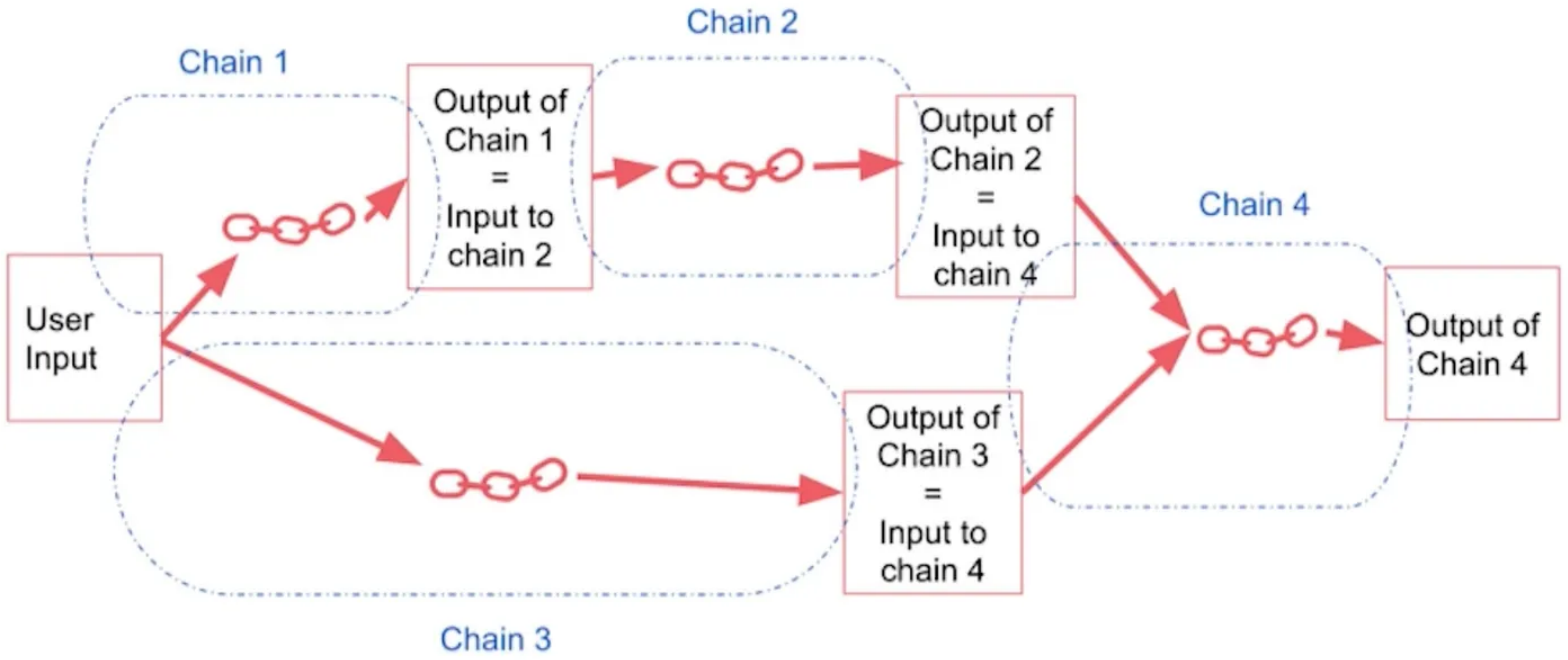
Sequential Chains

- Chains can be executed in a sequence.
- Two types:
  - Simple Sequential Chain and
  - SequentialChain

Simple Sequential Chain



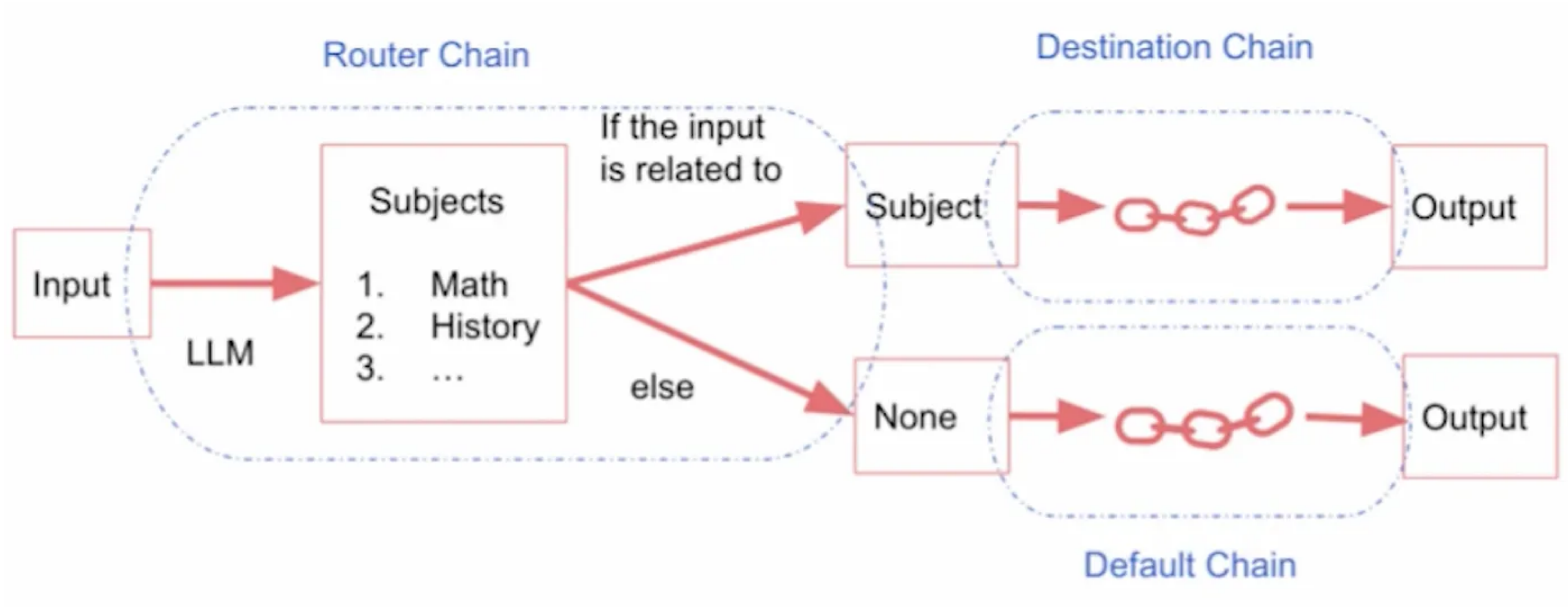
Sequential Chain


Chain 1
Chain 2
Chain 3
Chain 4
User Input
Output of Chain 1 = Input to chain 2
Output of Chain 2 = Input to chain 4
Output of Chain 3 = Input to chain 4
Output of Chain 4

Router Chain

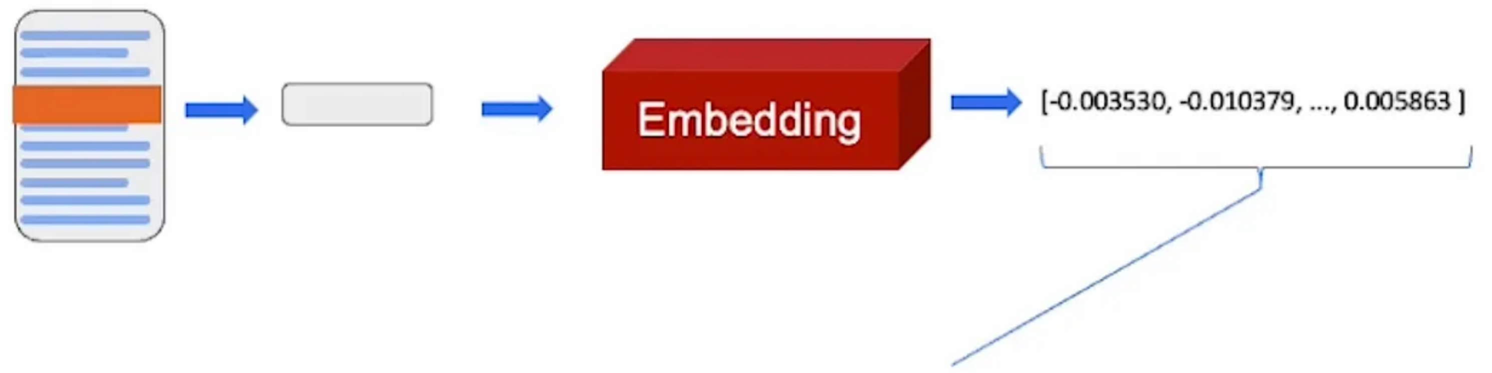
- RouterChain dynamically selects the next chain to execute.
- It routes the flow of execution based on certain conditions.

Router Chain



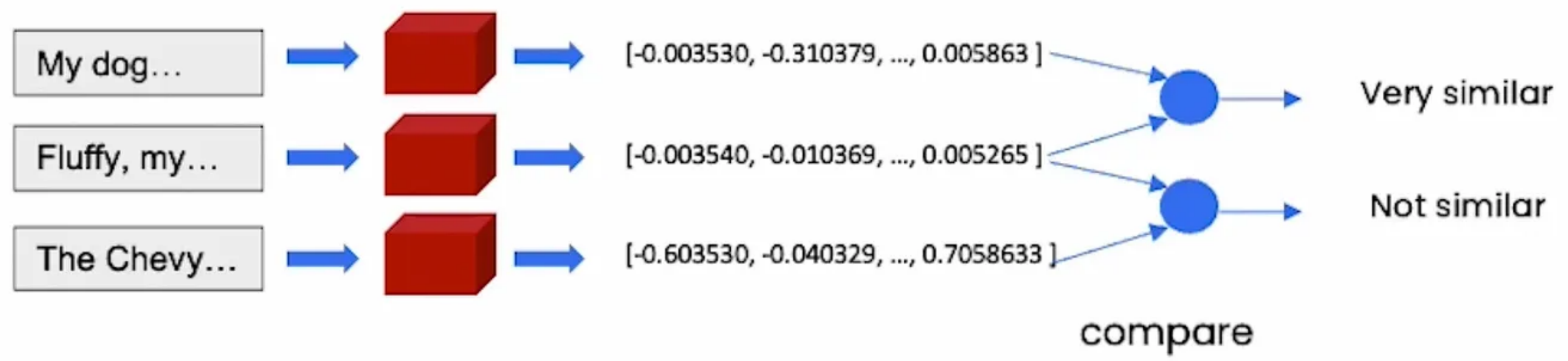
Embeddings

Embeddings

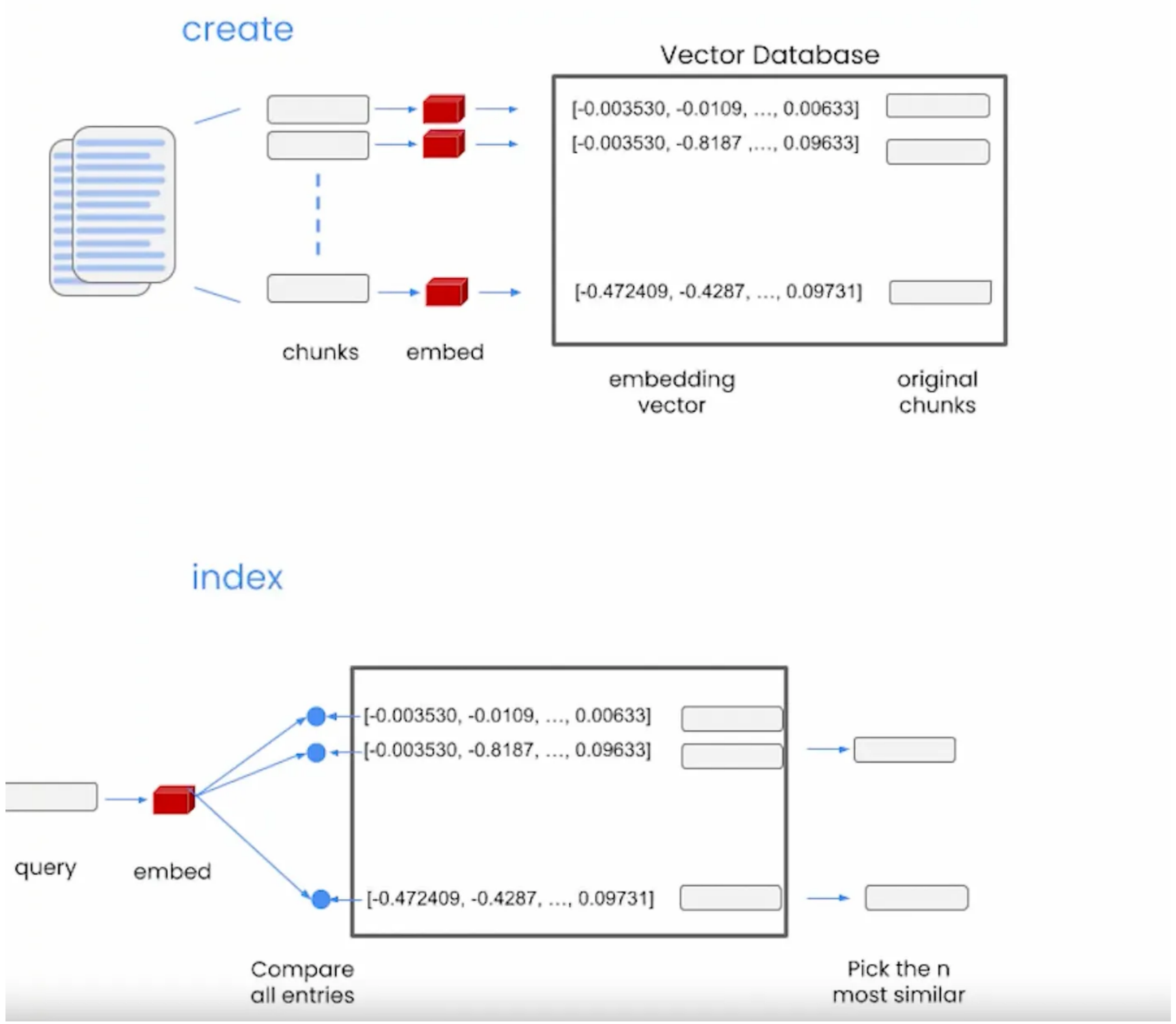


- Embedding vector captures content/meaning
- Text with similar content will have similar vectors

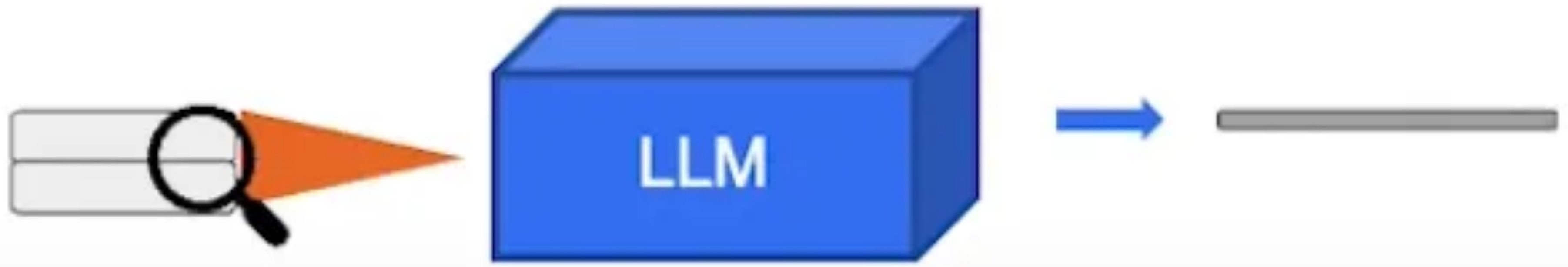
1) My dog Rover likes to chase squirrels.
2) Fluffy, my cat, refuses to eat from a can.
3) The Chevy Bolt accelerates to 60 mph in 6.7 seconds.



Vector Database



Process with llm



The returned values can now fit in the LLM context

Agents

Agents

The purpose of the agent is to treat the LLM as a reasoning engine. You can provide it with text or other sources of information, and this LLM can use background knowledge learned from the internet to help answer questions and infer content.

This section mainly covers how to create and use agents, and how to equip them with different types of tools within the inner LangChain, allowing the agents to interact with any data storage, API, and functions.

Initialize Agents

Predefined Tools

- **llm-math**: Essentially a Chain, it uses a language model and calculator to solve mathematical problems.
- **wikipedia**: Wikipedia tool, connected to the Wikipedia API, allowing run search queries from Wikipedia and returning results.

Example: Initializing an agent and asking it to calculate 25% of 300.

Python Agent

Applications like Copilot and code interpreter plugins write code using a language model and execute the generated code. Based on LangChain, we can also implement such applications.

PythonREPLTool

A Python execution environment that, based on input prompts, uses Python for reasoning and processes the results accordingly, such as printing them out.

Example: Sorting a list of customers by last name and then by first name.

Define Your Own Tool

Advantages

An advantage of agents is that you can link them to your own sources of information, your own APIs, and your own data, among other things.