Introduction to LangChain

Last Updated : 25 Aug, 2025

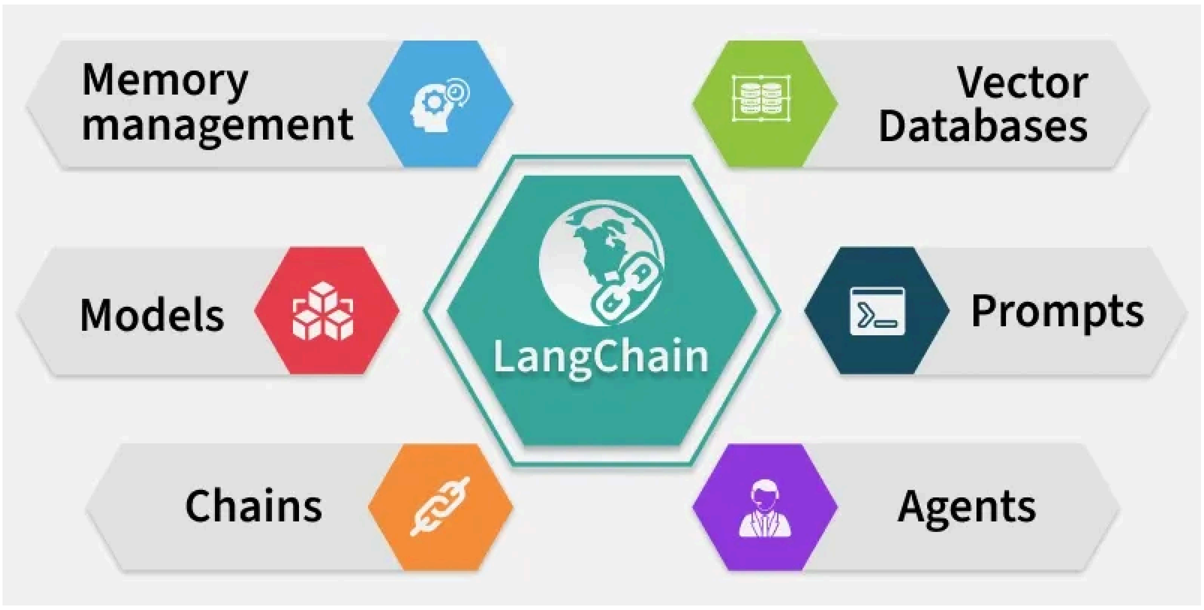
LangChain is an open-source framework designed to simplify the creation of applications using large language models (LLMs). It provides a standard interface for integrating with other tools and end-to-end chains for common applications. It helps AI developers connect LLMs such as GPT-4 with external data and computation. This framework comes for both Python and JavaScript.

Key benefits include:

- **Modular Workflow**: Simplifies chaining LLMs together for reusable and efficient workflows.
- **Prompt Management**: Offers tools for effective prompt engineering and memory handling.
- **Ease of Integration**: Streamlines the process of building LLM-powered applications.

Key Components of LangChain

Lets see various components of Langchain:



1. Chains: Chains define sequences of actions, where each step can involve querying an LLM, manipulating data or interacting with external tools. There are two types:

- **Simple Chains**: A single LLM invocation.
- **Multi-step Chains**: Multiple LLMs or actions combined, where each step can take the output from the previous step.

2. Prompt Management: LangChain facilitates managing and customizing prompts passed to the LLM. Developers can use **PromptTemplates** to define how inputs and outputs are formatted before being passed to the model. It also simplifies tasks like handling dynamic variables and prompt engineering, making it easier to control the LLM's behavior.

3. Agents: Agents are autonomous systems within LangChain that take actions based on input data. They can call external APIs or query databases dynamically, making decisions based on the situation. These agents leverage LLMs for decision-making, allowing them to respond intelligently to changing input.

4. Vector Database: LangChain integrates with a vector database which is used to store and search high-dimensional vector representations of data. This is important for performing similarity searches, where the LLM converts a query into a vector and compares it against the vectors in the database to retrieve relevant information.

Vector database plays a key role in tasks like document retrieval, knowledge base integration or context-based search providing the model with dynamic, real-time data to enhance responses.

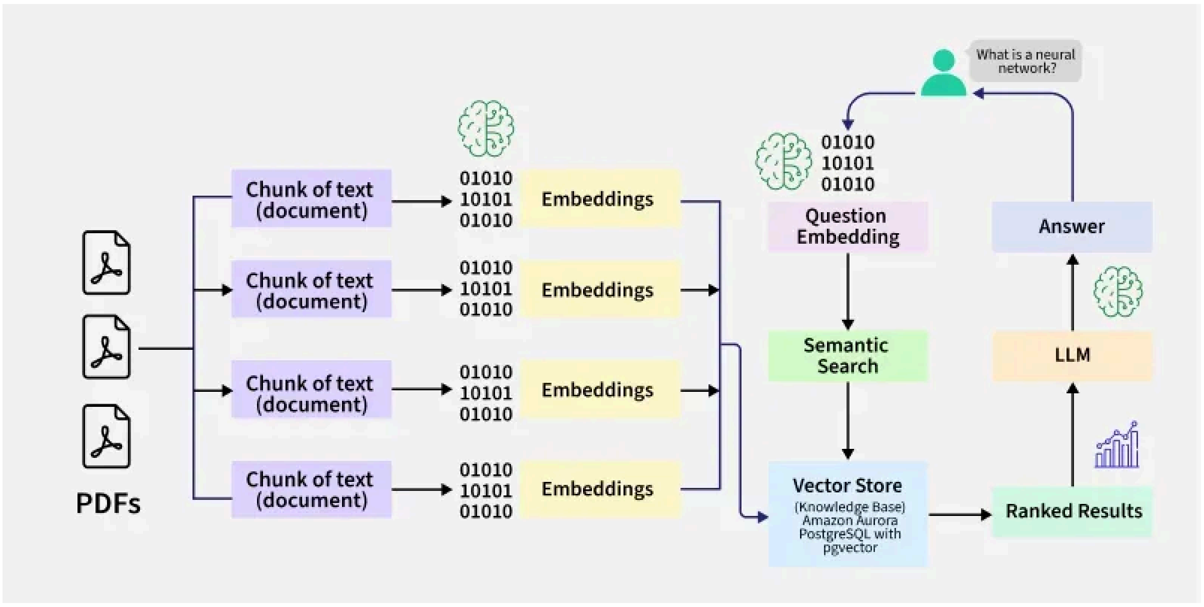
5. Models: LangChain is model-agnostic meaning it can integrate with different LLMs such as OpenAI's GPT, Hugging Face models, DeepSeek R1 and more. This flexibility allows developers to choose the best model for their use case while benefiting from LangChain’s architecture.

6. Memory Management: LangChain supports memory management allowing the LLM to "remember" context from previous interactions.

This is especially useful for creating conversational agents that need context across multiple inputs. The memory allows the model to handle sequential conversations, keeping track of prior exchanges to ensure the system responds appropriately.

How LangChain Works?

LangChain follows a structured pipeline that integrates user queries, data retrieval and response generation into seamless workflow.



LangChain Pipeline

1. User Query

The process begins when a user submits a query or request.

> *For example, a user might ask, "What's the weather like today?" This query serves as the input to the LangChain pipeline.*

2. Vector Representation and Similarity Search

- Once the query is received, LangChain converts it into a **vector representation** using embeddings. This vector captures the semantic meaning of the query.
- The vector is then used to perform a similarity search in a vector database. The goal is to find the most relevant information or context stored in the database that matches the user's query.

3. Fetching Relevant Information

Based on the similarity search, LangChain retrieves the most relevant data or context from the database. This step ensures that the language model has access to accurate and contextually appropriate information to generate a meaningful response.

4. Generating a Response

The retrieved information is passed to the language model (e.g., OpenAI's GPT, Anthropic's Claude or others). The LLM processes the input and generates a response or takes an action based on the provided data.

> *For example, if the query is about the weather, the LLM might generate a response like, "Today's weather is sunny with a high of 75°F."*

The formatted response is returned to the user as the final output. The user receives a clear, accurate and contextually relevant answer to their query.

Step-by-Step Implementation

Let's implement a model using LangChain and OpenAI API:

Step 1: Install the dependencies

We will install all the required dependencies for our model.

- **langchain:** the core LangChain framework (chains, prompts, tools, memory, etc.).
- **langchain-openai:** OpenAI model wrapper for LangChain (GPT-3.5, GPT-4, etc.).
- **python-dotenv:** to securely manage our API keys inside a .env file.

```
!pip install langchain langchain-openai python-dotenv
```

Step 2: Import Libraries

We will import all the required libraries.

- **os:** interact with environment variables.
- **load_dotenv:** loads .env file values into our environment.
- **OpenAI:** lets us call OpenAI's GPT models in LangChain.
- **PromptTemplate:** define structured prompts with placeholders.
- **StrOutputParser:** ensures model response is returned as clean string text.

```
import os
from dotenv import load_dotenv
from langchain_openai import OpenAI
from langchain.prompts import PromptTemplate
from langchain_core.output_parsers import StrOutputParser
```

Step 3: Load API Key

We need to load the OpenAI API Key, but first we create a .env file to store our API key.

```
OPENAI_API_KEY = your_openai_api_key_here
```

Now we use the os.getenv() function to securely fetch the API key.

```
load_dotenv()
api_key = os.getenv("OPENAI_API_KEY")
```

Step 4: Initialize the OpenAI LLM

We initialize the LLM model:

- **temperature=0.7:** controls creativity (0 = deterministic, 1 = very creative).
- **openai_api_key=api_key:** authenticates with OpenAI.

```
llm = OpenAI(
    temperature=0.7,
    openai_api_key=api_key
)
```

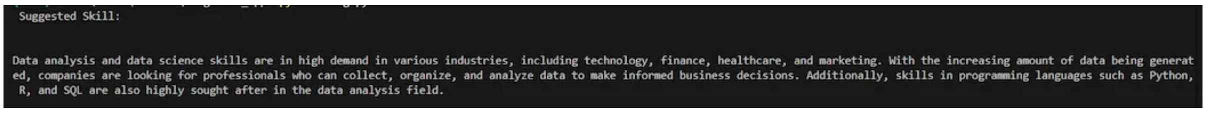
Step 5: Run a Simple Prompt

We will check by running a simple prompt.

- **.invoke():** sends prompt to LLM and returns text output.

```
prompt = "Suggest me a skill that is in demand?"
response = llm.invoke(prompt)
print(" Suggested Skill:\n", response)
```

Output:



```
 Suggested Skill:


Data analysis and data science skills are in high demand in various industries, including technology, finance, healthcare, and marketing. With the increasing amount of data being generat
ed, companies are looking for professionals who can collect, organize, and analyze data to make informed business decisions. Additionally, skills in programming languages such as Python,
 R, and SQL are also highly sought after in the data analysis field.
```

Output

Step 6: Create a Prompt Template

We create a dynamic prompt where {year} can be replaced with input values.

```
template = "Give me 3 career skills that are in high demand in {year}.
prompt_template = PromptTemplate.from_template(template)
```

Step 7: Build a Chain with LCEL

LCEL (LangChain Expression Language): It’s a new way to compose LLM workflows using a simple, chainable syntax with the | (pipe) operator.

1. prompt_template

- Fills placeholders (like {year}) with actual inputs.
- Example: "Give me 3 career skills in 2025."

2. llm

- Sends the formatted prompt to the OpenAI model.

- Example input: "Give me 3 career skills in 2025."
- Example output: "1. Data Analytics\n2. AI/ML\n3. Cybersecurity"

3. StrOutputParser()

- Cleans up and ensures the LLM's response is returned as a string.

```
chain = prompt_template | llm | StrOutputParser()
```

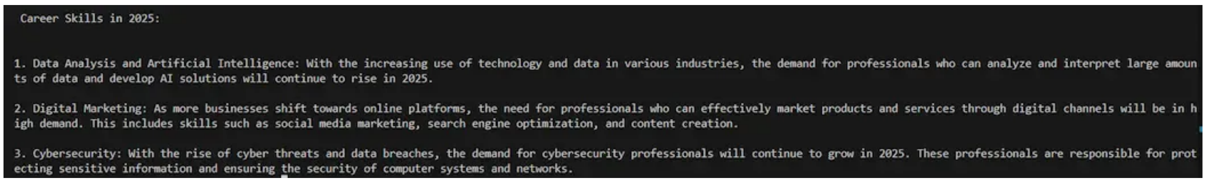
Step 8: Run the Chain

We run the chain to fetch results.

- **.invoke({"year": "2025"})** replaces {year} with 2025 in the prompt.
- **Final formatted prompt:** "Give me 3 career skills that are in high demand in 2025."

```
response = chain.invoke({"year": "2025"})
print("\n Career Skills in 2025:\n", response)
```

Output:



```
 Career Skills in 2025:

1. Data Analysis and Artificial Intelligence: With the increasing use of technology and data in various industries, the demand for professionals who can analyze and interpret large amoun
ts of data and develop AI solutions will continue to rise in 2025.

2. Digital Marketing: As more businesses shift towards online platforms, the need for professionals who can effectively market products and services through digital channels will be in h
igh demand. This includes skills such as social media marketing, search engine optimization, and content creation.

3. Cybersecurity: With the rise of cyber threats and data breaches, the demand for cybersecurity professionals will continue to grow in 2025. These professionals are responsible for prot
ecting sensitive information and ensuring the security of computer systems and networks.
```

Output

Applications of LangChain

Let's see the applications of LangChain,

- **Chatbots and Virtual Assistants:** They can be designed to remember past interactions, connect with external APIs and deliver more natural, context-aware conversations.
- **Document Question Answering:** Users can query PDFs, research papers, contracts or enterprise documentation and get precise answers instead of manually searching.
- **Knowledge Management Systems:** They help organize and retrieve company knowledge by linking LLMs with structured and

unstructured data, enabling intelligent search, summarization and recommendations.
- **Workflow Automation:** Complex multi-step processes like customer support ticket resolution, report generation or CRM updates can be automated seamlessly.
- **Data Analysis and Business Intelligence (BI):** Natural language queries can be translated into SQL, turning raw data into insights, charts or business reports with minimal effort.

The LangChain framework is a great interface to develop interesting AI-powered applications and from personal assistants to prompt management as well as automating tasks. So, keep learning and keep developing powerful applications.

Introduction to Langchain

Comment More info