Introduction to LangChain

22 May 2025 22:24

What is LangChain

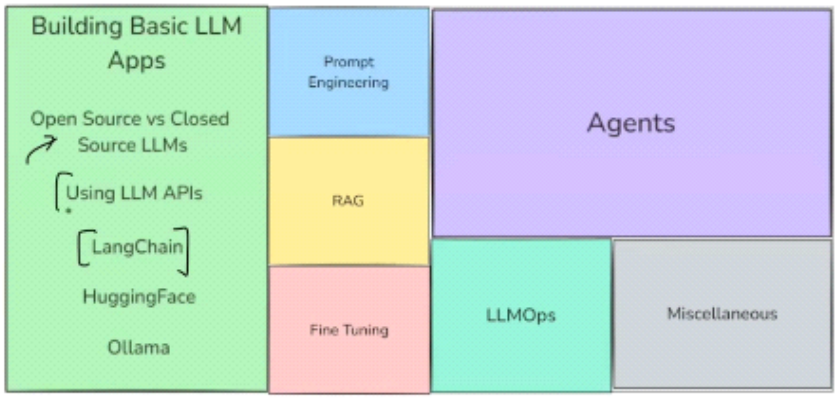
29 January 2025 08:35

LangChain is an open source framework that helps in building LLM based applications. It provides modular components and end-to-end tools that help developers build complex AI applications, such as chatbots, question-answering systems, retrieval-augmented generation (RAG), autonomous agents, and more.

1. Supports all the major LLMs
2. Simplifies developing LLM based applications
3. Integrations available for all major tools
4. Open source/Free/Actively developed
5. Supports all major GenAI use cases

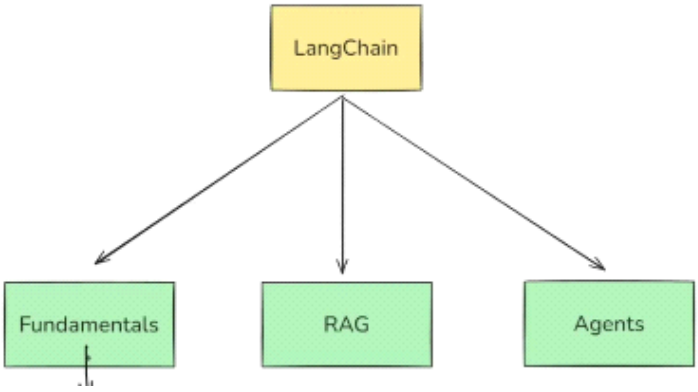
Why LangChain first

29 January 2025 08:35



Curriculum Structure

29 January 2025 08:36



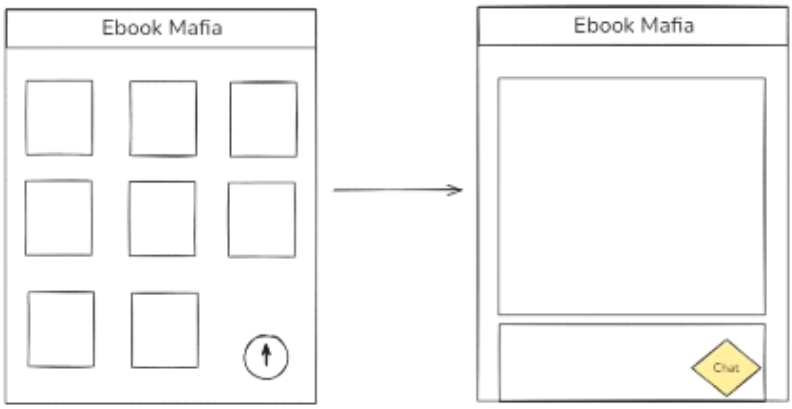
What is LangChain

07 January 2025 23:14

LangChain is an open-source framework for developing applications powered by large language models (LLMs).

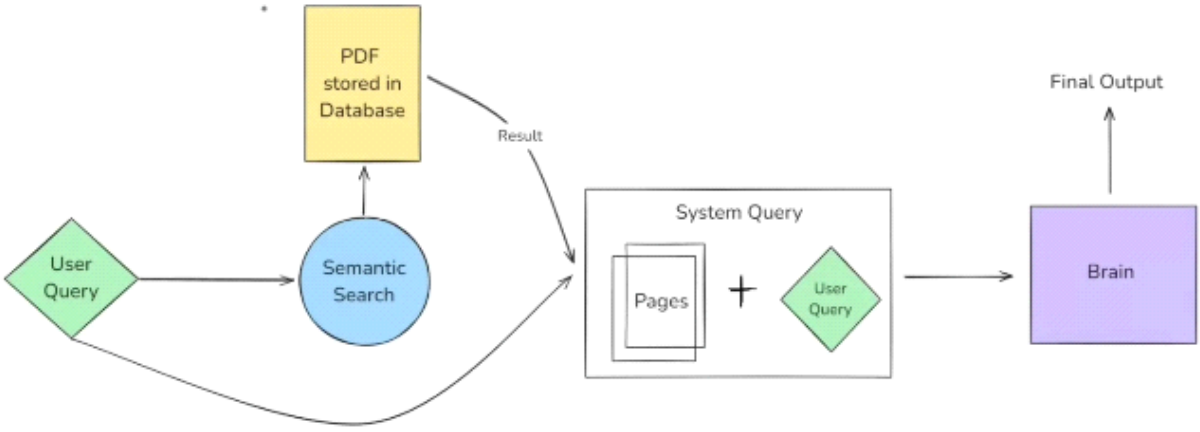
Why do we need LangChain

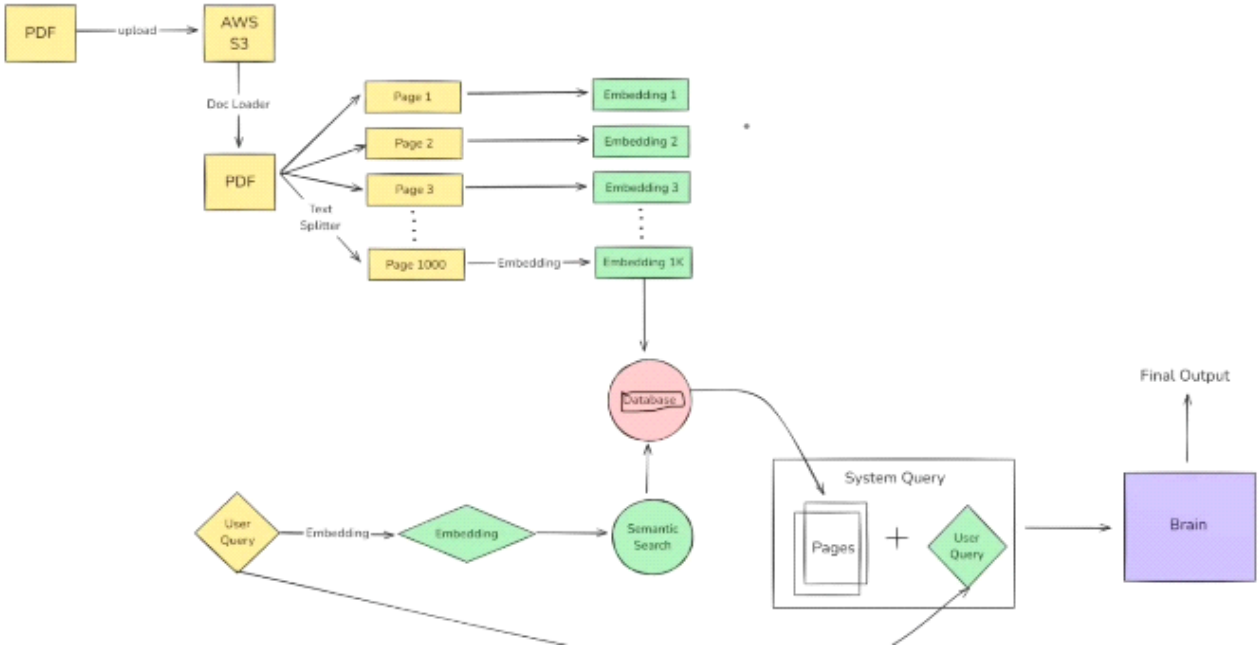
21 January 2025 23:34



Example Queries

1. Explain page number 5 as if I am a 5 year old
2. Generate a True False exercise on Linear Regression
3. Generate notes for Decision Trees





Benefit of Langchain

Concept of chains

Model Agnostic Development

Complete ecosystem

Memory and state handling

What can you build?

21 January 2025 23:34

Conversational Chatbots → scale → chatbot

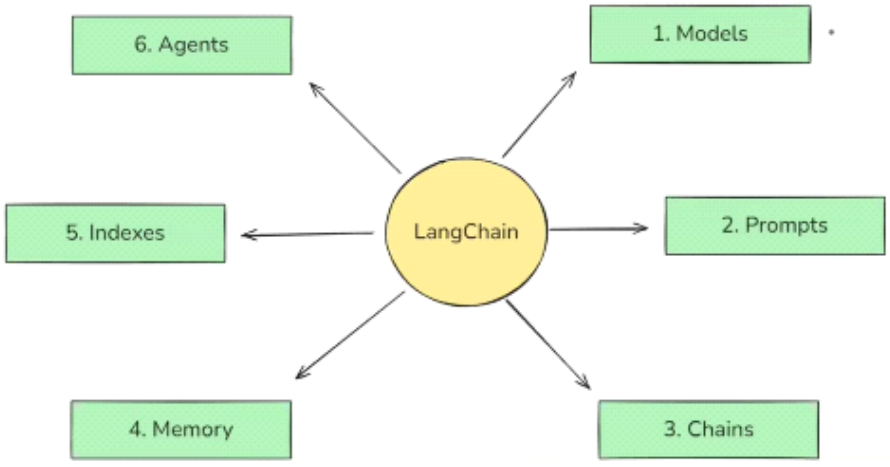
AI Knowledge Assistants

AI Agents

Workflow Automation

Summarization/Research Helpers

LangChain Components



openAI

langchain

Claude

```python
from langchain_openai import ChatOpenAI
from dotenv import load_dotenv

load_dotenv()

model = ChatOpenAI(model='gpt-4', temperature=0)

result = model.invoke("Now divide the result by 1.5")

print(result.content)
```

```python
from langchain_anthropic import ChatAnthropic
from dotenv import load_dotenv

load_dotenv()

model = ChatAnthropic(model='claude-3-opus-20240229')

result = model.invoke("Hi who are you")

print(result.content)
```

Prompts

1. Dynamic & Reusable Prompts

```
from langchain_core.prompts import PromptTemplate

prompt = PromptTemplate.from_template('Summarize {topic} in {emotion} tone')

print(prompt.format(topic='Cricket', length='fun'))
```

2. Role-Based Prompts

```
# Define the ChatPromptTemplate using from_template
chat_prompt = ChatPromptTemplate.from_template([
    ("system", "Hi you are a experienced {profession}"),
    ("user", "Tell me about {topic}"),
])

# Format the prompt with the variable
formatted_messages = chat_prompt.format_messages(profession="Doctor", topic="Viral Fever")
```

3. Few Shot Prompting

```
examples = [
    {"input": "I was charged twice for my subscription this month.", "output": "Billing Issue"},
    {"input": "The app crashes every time I try to log in.", "output": "Technical Problem"},
    {"input": "Can you explain how to upgrade my plan?", "output": "General Inquiry"},
    {"input": "I need a refund for a payment I didn't authorize.", "output": "Billing Issue"},
]
```

```
# Step 2: Create an example template
example_template = """
Ticket: {input}
Category: {output}
"""
```

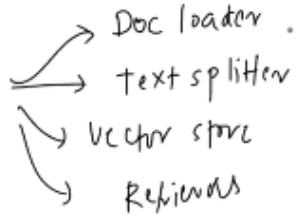
```
# Step 3: Build the few-shot prompt template
few_shot_prompt = FewShotPromptTemplate(
    examples=examples,
    example_prompt=PromptTemplate(input_variables=["input", "output"], template=example_template),
    prefix="Classify the following customer support tickets into one of the categories: 'Billing
    Issue', 'Technical Problem', or 'General Inquiry'.\n\n",
    suffix="\nTicket: {user_input}\nCategory:",
    input_variables=["user_input"],
)
```

```
Classify the following customer support tickets into one of the categories: 'Billing Issue', 'Technical Problem', or 'General
Inquiry'.

Ticket: I was charged twice for my subscription this month.
Category: Billing Issue

Ticket: The app crashes every time I try to log in.
Category: Technical Problem

Ticket: Can you explain how to upgrade my plan?
Category: General Inquiry

Ticket: I need a refund for a payment I didn't authorize.
Category: Billing Issue

Ticket: I am unable to connect to the internet using your service.
Category:
```

Indexes

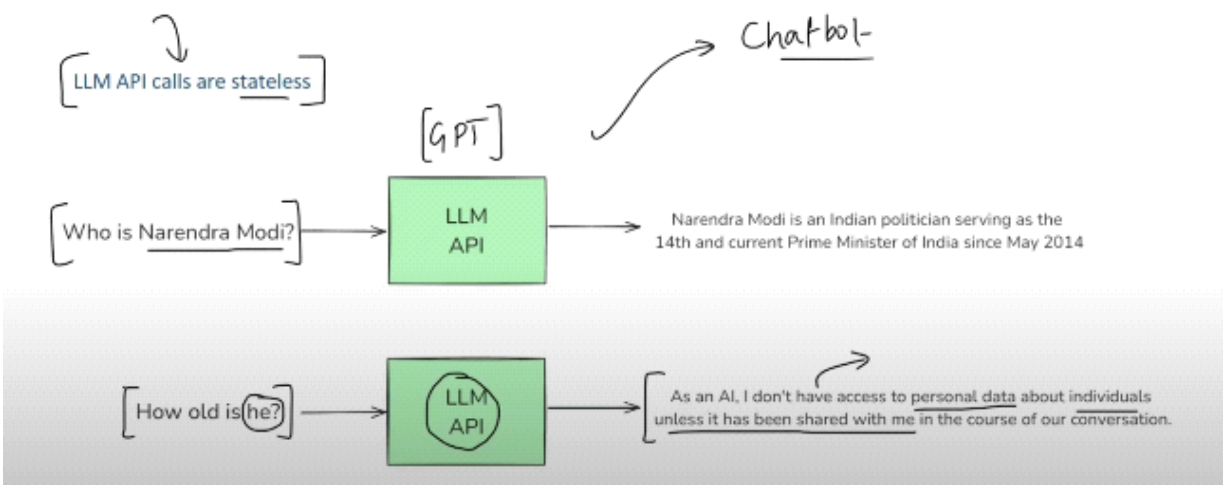
23 January 2025 10:31

Indexes connect your application to external knowledge—such as PDFs, websites or databases



Memory

23 January 2025 10:31



- ConversationBufferMemory: Stores a transcript of recent messages. Great for short chats but can grow large quickly.
- ConversationBufferWindowMemory: Only keeps the last N interactions to avoid excessive token usage.
- Summarizer-Based Memory: Periodically summarizes older chat segments to keep a condensed memory footprint.
- Custom Memory: For advanced use cases, you can store specialized state (e.g., the user's preferences or key facts about them) in a custom memory class.