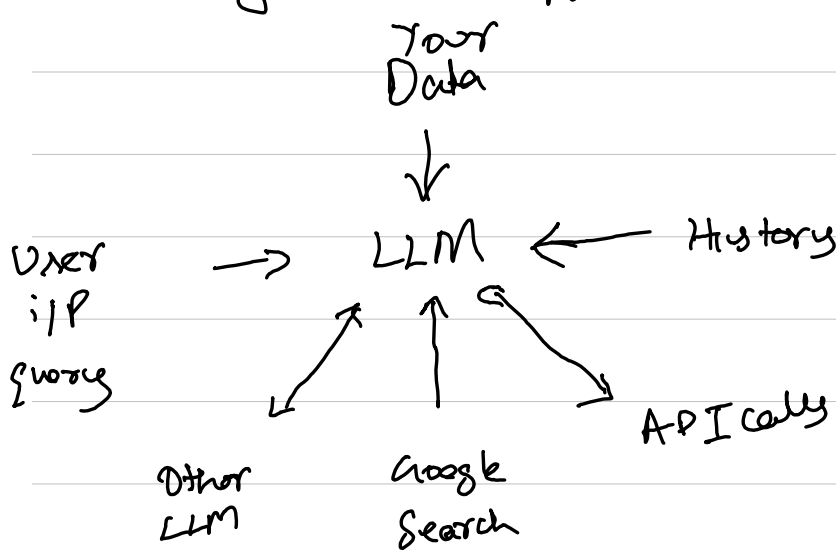
LANGCHAIN is a simplified framework that simplifies the process of building LLM powered Apps

Langchain has
-> tools
-> Abstraction

-> Building an LLM App isn't straightforward



-> To build an intelligent App like above, you will need LangChain.

-> Langchain has same interface across all LLM vendors.

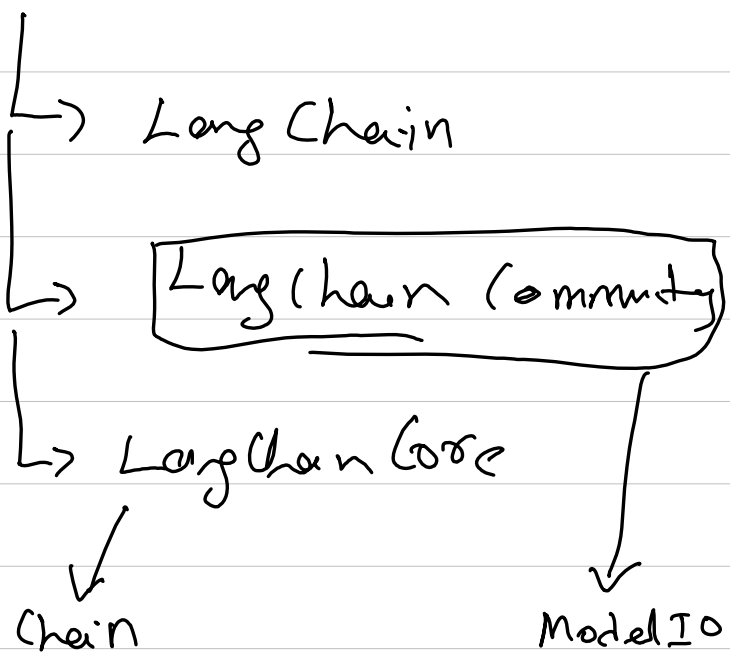
-> LangChain (LC) have prompt management, optimization, serialization

-> LC also has document loaders (.pdf, word, emails, etc)

-> LC supports tool calling, etc

-> LC creates abstraction "chains".

-> LangChain API



Expression Language
DocumentLoad
Vector Store
Agent tools

-> Langchain Hub
↓
Contains a lot of prompts contributed by community.

Python 3.11.9

pipenv install these

- langchain
- langchain-openai
- langchain-community
- langchainhub
- black

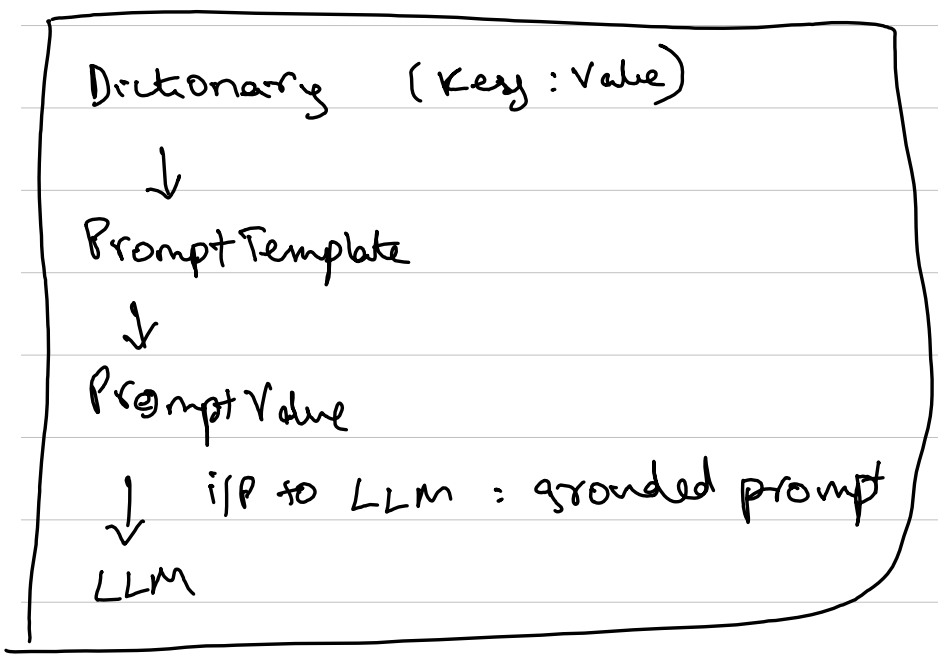
We used langchain core version of 0.3.61

Also install python-dotenv

PromptTemplate:

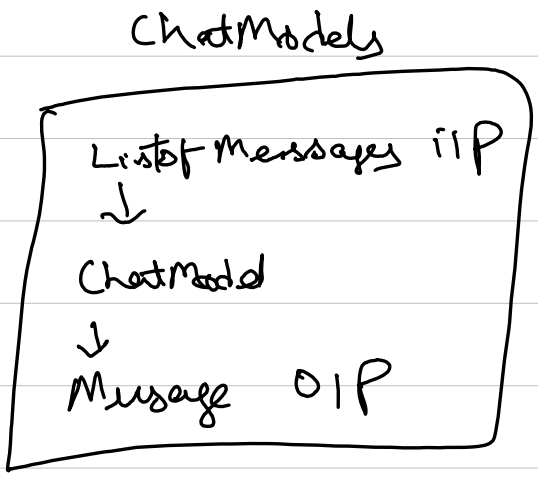
Prompt is input to LLM (instructions + data)

PromptTemplate helps to translate user inputs, parameters into instruction for LLM



ChatModels:

Provide access to LLM via ChatModels



→ They also support:
- → tool calling
- → Structured O/P
- → Multimodal i/p o/p.

→ By combining prompt templates, chatmodels, we can pass data to LLMs.

→ We can combine this with external APIs; pdf readers, file writers, etc in LangChain to properly make an Application

→ We can combine/Chain all these together to make an Application

→ Agent: In LLM world, an agent is a loop that repeatedly reads a conversation, decides what action or tool call to make, feeds the result back into the model until the goal is met

→ LangChain is a Python framework that gives us building blocks (prompt templates, chains, memories, callbacks, tool wrappers) to build LLM apps

→ REACT → It is a prompting pattern where the model (LLM) emits an interleaved thought → Action → Observation trace

letting it reason, call tools, see result in 1 pass.

Task:

→ We want to build a popular person information summarizer.

→ This is how you would do it using LangChain modules.

```
information = """ . . . . . .
        Some information about a
    popular person"""

summary_template = """
    given the information: {information}
    about a person, I want you create:
        1. A short summary
        2. two interesting facts
"""

summary_prompt_template = PromptTemplate(
    input_variables = ["information"],
    template = summary_template)
llm = ChatOpenAI(temperature=0,
                 model_name = 'gpt-3.5-turbo')
chain = summary_prompt_template | llm
```

pipe symbol

```
res = chain.invoke(input={'information':
                          information})
print(res)
```

query

Using Open Source Models with Langchain

- Langchain can be configured to be used with Ollama, LLAMA 3, Mixtral, etc (local models)
- However the latency will suffer as the models take forever to run.

→ pipenv install langchain-ollama

→ In your code, instead of `ChatOpenAI` object instantiation use `ChatOllama`.

eg: `llm = ChatOllama(model='llama3')`

Most of the heavy lifting comes from this line

```
chain = summary_prompt_template | llm
        | StrOutputParser
```

here we are using operator overloading to chain prompt → LLM → format O/P → StrOutputParser

→ The ice breaker application
- → gathers information on a given individual (twitter, Linkedin)
- → Summarizes it
- → display to user

→ Building this App using:

1) Chains
2) Agents
3) Custom Agents
4) Tools, toolkits
5) Output Parsers

Scrapin io signup

Service API

to

scrape LinkedIn profile and data

→ Get data of linkedin profiles as JSON.

→ Pass to LLM if necessary, make decisions / generate o/p based on that profile data.

☆

LLM isn't connected to the Web. They can be considered as a repo of static information

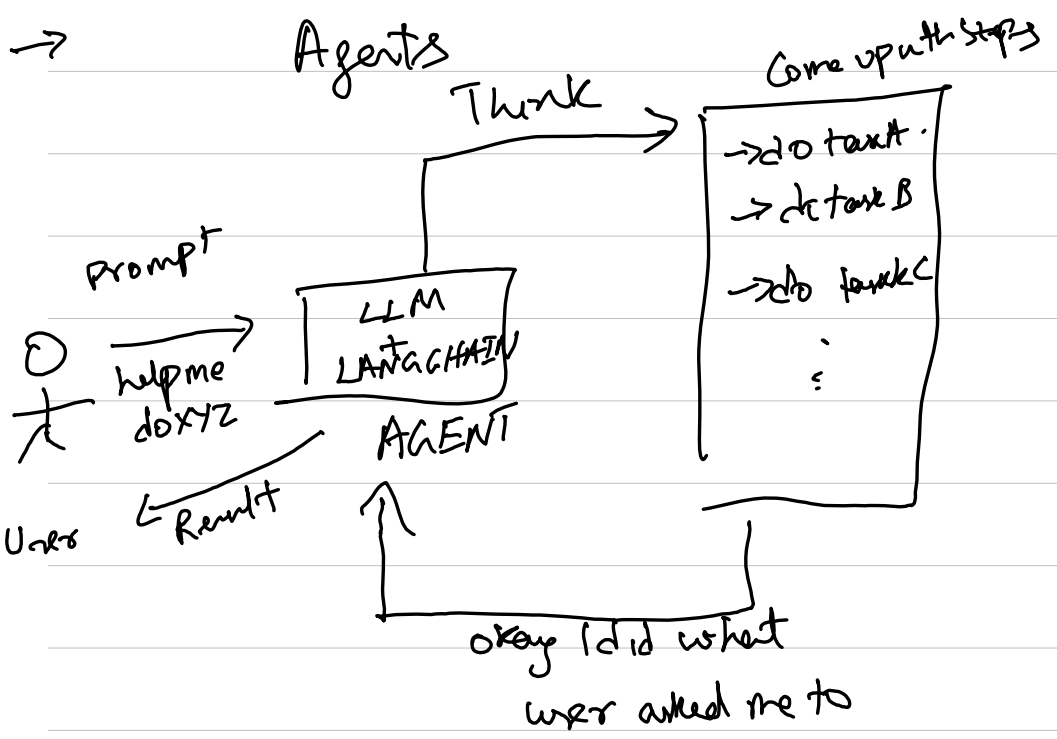
→ Agent / bot can connect to both internet through tools.

→ tools are scripts or 3rd party services

→ LLM with a prompt can breakdown big problem into smaller problems achieve those tasks (using tool calls).

→ These actions can be then chained to one another.

→



→ Chain of Thought is a prompt engg technique that helps LLM to think and answer questions

→ Agents use LLM + tools to solve what user is asking the agent to solve

→ Core idea of Agents is to use language model to choose a sequence of tasks to take.

→ Agents use LLM as a reasoning engine to determine which action to take.

→ Build a search agent.
  ↳ Get name from text box
  ↳ search name (REACT agent)
    ↳ get linkedin profile based on URL.

We build a search tool
  ↳ linkedin_lookup_tool.py

→ Import
  - prompt template
  - tool
  - create react agent
  - Agent executor

from langchain hubs

hub
  ↳ hub contains premade prompts.