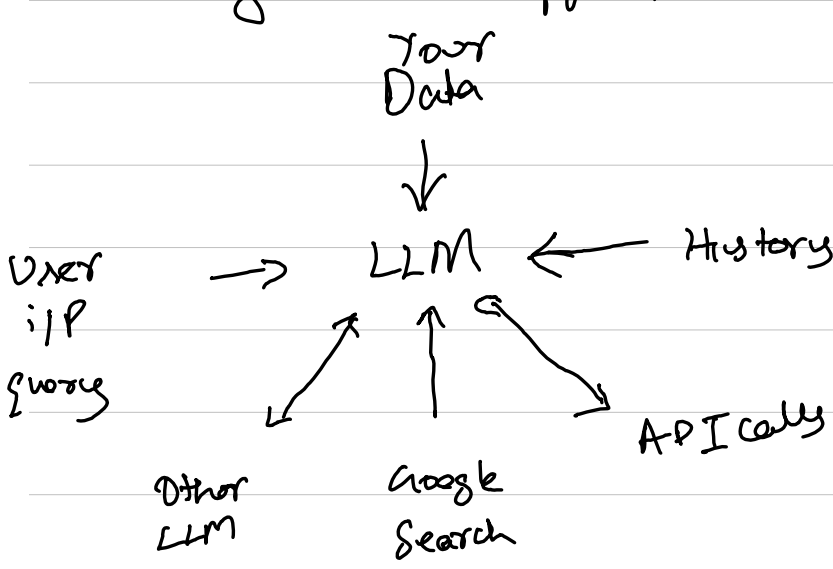
LANGCHAIN is a simplified framework that simplifies the process of LLM powered Apps

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.

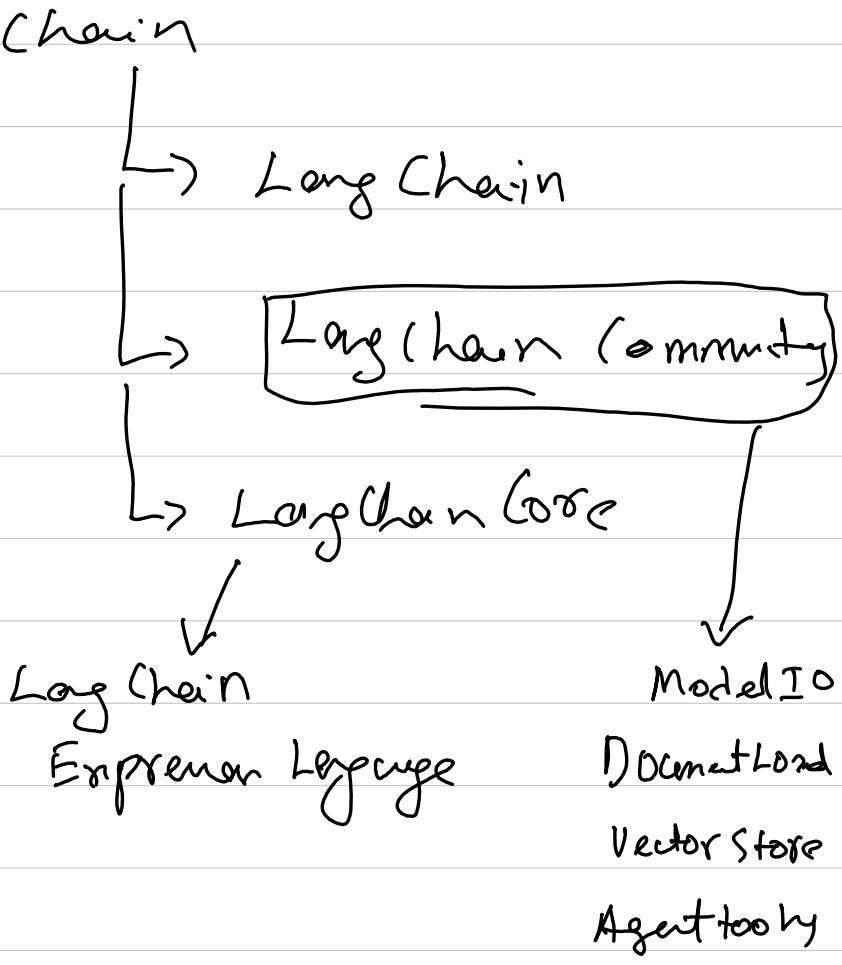
-> LangChain (LC) have prompt management, optimization, serialization

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

-> LC supports tool calling, etc

-> LC creates abstraction "chains".

-> LangChain



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

Python 3.11.9

Pipenv install three

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

PromptTemplate:

Prompt is input to LLM (instructions + data)

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

Dictionary (key : value)

↓

PromptTemplate

↓

PromptValue

↓ i/p to LLM : grounded prompt

LLM

Chat Models:

Provide access to LLM via ChatModels

List of messages i/p

↓

ChatModel

↓

Message o/p

→ 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

Deadline

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})
```

query

```
print(res)
```