LangChain Overview

Sean Ma

maye-msft@outlook.com

Start →

What is LangChain?

LangChain is a framework for developing applications powered by language models. It consists of a set of components

- 📝 **Prompts** - LangChain provides several classes and functions to make constructing and working with prompts easy.
- 🎨 **Indexes** - Indexes refer to ways to structure documents so that LLMs can best interact with them.
- 👩‍💻 **Memory** - Memory is the concept of storing and retrieving data in the process of a conversation.
- 👰 **Chains** - Chains is an incredibly generic concept which returns to a sequence of modular components (or other chains) combined in a particular way to accomplish a common use case.
- 🎥 **Agents** - Help application to access resources and tools depends on the user's input

Read more about LangChain Doc?

Introduction

We will use LangChain and Azure OpenAI.

In this session will use Prompts, Chains and Agents.

We will create a dataset about fruits.

And ask the questions:

- how many rows are there?
- which fruit with the lowest calories and what country produce it most?

First Question

List five most popular fruits.

```
from langchain.llms import AzureOpenAI
llm = AzureOpenAI(
    temperature=0,
    deployment_name="<model-deployment-id>",
    model_name="text-davinci-003",
)

QUESTION = "List five most popular fruits."
fruits_names = llm(QUESTION)
print(fruits_names)
```

```
The result is:
1. Apples
2. Bananas
3. Oranges
4. Strawberries
5. Watermelons
```

OutputParser

Format the result to a Python list by CommaSeparatedListOutputParser

```python
from langchain.output_parsers import CommaSeparatedListOutputParser
from langchain.prompts import PromptTemplate
from langchain.chains import LLMChain

output_parser = CommaSeparatedListOutputParser()
format_instructions = output_parser.get_format_instructions()
prompt = PromptTemplate(
    template="List five most popular {subject}.\n{format_instructions}",
    input_variables=['subject'],
    partial_variables={"format_instructions": format_instructions}
)

chain = LLMChain(llm=llm, prompt=prompt)
output = chain.run(subject='fruits')
fruits_names = output_parser.parse(output)
print(fruits_names)
```

```
The result is:
['Apples', 'Bananas', 'Oranges', 'Strawberries', 'Grapes']
```

Next Two Questions

- Agriculture question: What is the top fruit {name} producing country?
- Nutrition question: What is the calories value of fruit {name} per 100 grams?

The {name} is one fruit name in the result of the first question.

Prompt Template

Two prompt one for Agriculture question and one for Nutrition question.

```
agriculture_template = """You are a very smart agriculture professor. \
You are great at answering questions about agriculture. \
When you don't know the answer to a question you admit that you don't know.

Here is a question:
{input}"""


nutrition_template = """You are a very good nutritionist. \
You are great at answering nutrition questions. \
When you don't know the answer to a question you admit that you don't know.

Here is a question:
{input}"""
```

Prompt Router

We create a router and LangChain will router the question to the right prompt.

```
prompt_infos = [
    {
        "name": "agriculture",
        "description": "Good for answering questions about agriculture",
        "prompt_template": agriculture_template
    },
    {
        "name": "nutrition",
        "description": "Good for answering questions about nutrition",
        "prompt_template": nutrition_template
    },
]
destinations = [f"{p['name']}: {p['description']}" for p in prompt_infos]
destinations_str = "\n".join(destinations)
router_template = MULTI_PROMPT_ROUTER_TEMPLATE.format(destinations=destinations_str)
router_prompt = PromptTemplate(
    template=router_template,
    input_variables=["input"],
    output_parser=RouterOutputParser(),
)
```

Router Chain

Create a chain to router the questions to the right prompt with llm.

```
destination_chains = {}
for p_info in prompt_infos:
    prompt = PromptTemplate(template=p_info["prompt_template"], input_variables=["input"])
    destination_chains[p_info["name"]] = LLMChain(llm=llm, prompt=prompt)
multi_prompt_chain = MultiPromptChain(
    router_chain=LLMRouterChain.from_llm(llm, router_prompt),
    default_chain= ConversationChain(llm=llm, output_key="text"),
    destination_chains=destination_chains)
question = "what is the top fruit apple producing country?"
print("Q1: "question+"\n"+multi_prompt_chain.run(input=question))
question = "what is the calories value of apple per 100 grams?"
print("Q2: "question+"\n"+multi_prompt_chain.run(input=question))
```

```
The result is:
Q1: what is the top fruit apple producing country?
The top apple producing country is China.
According to the Food and Agriculture Organization of the United Nations,
China produced nearly 37 million metric tons of apples in 2019,
accounting for nearly half of the world's total apple production.
Q2: what is the calories value of apple per 100 grams?
The calorie value of an apple per 100 grams is 52 calories.
```

Extract Value from Result

Create a chain to extract the value from the result.

```
extract_template = """
You are a very smart assistant. \
You need to extract information from a text base on instructions. \
Here is the instructions: \
{extract_instructions} \
Here is  the text: \
{text} \
{format_instructions}"""

extract_prompt = PromptTemplate(
    template=extract_template,
    input_variables=['extract_instructions', 'text'],
    partial_variables={"format_instructions": format_instructions}
)
extract_chain = LLMChain(llm=llm, prompt=extract_prompt, output_key="extracted_value")
output = extract_chain.run({"extract_instructions":"Get the calories number only.",
                   "text":"The calorie value of an apple per 100 grams is 52 calories."})
```

```
The result is:
52
```

SequentialChain

Join the multi prompt chain and extract chain into a SequentialChain.

```
from langchain.chains import SequentialChain
overall_chain = SequentialChain(
    chains=[multi_prompt_chain, extract_chain],
    input_variables=["input", "extract_instructions"],
    output_variables=["extracted_value"],
    verbose=True)
output = overall_chain.run(input="What is the top fruit apple producing country?",
                           extract_instructions="Get the country name only.")
print(output)
output = overall_chain.run(input="what is the calories value of apple per 100 grams?",
                           extract_instructions="Get the calories number only.")
print(output)
```

```
The result is:
China
52
```

Agent

Add a fruit description with DuckDuckGo Search

```
from langchain.tools import DuckDuckGoSearchRun
from langchain.agents import Tool, AgentType, initialize_agent
search = DuckDuckGoSearchRun()
tools = [
    Tool(
        name = "Current Search",
        func=search.run,
        description="query with search engine",
    ),
]
agent_chain = initialize_agent(tools, llm, agent=AgentType.CHAT_ZERO_SHOT_REACT_DESCRIPTION)
result = agent_chain.run(input="give me a short description of fruit apple")
print(result)
```

```
the result is:
Apples are a type of fruit that are grown in orchards and have been bred to contain more sugar.
```

Put All Together

```
import pandas as pd
fruits_countries = [], fruits_kalories = [], fruits_desc = []
for name in fruits_names:
    output = overall_chain.run(input=f"What is the top fruit {name} producing country?",
                               extract_instructions="get the country name only.")
    fruits_countries.append(output_parser.parse(output)[0])
    output = overall_chain.run(input=f"what is the calories value of {name} per 100 grams",
                               extract_instructions="get the calories number only.")
    fruits_kalories.append(output_parser.parse(output)[0])
    output = agent_chain.run(input="give me a short description of fruit {name}")
    fruits_desc.append(output)
df = pd.DataFrame({"country": fruits_countries, "calories": fruits_kalories, "desc": fruits_desc},
                  index = fruits_names)
print(df)
```

```
the result is:
           country calories                                                desc
Apple        China       52  Apples are a deciduous tree, generally standin...
Banana       India       89  Bananas are an elongated, edible fruit - botan...
Orange      Brazil       47  Orange is a citrus fruit that originated from ...
Strawberry   China       33  Strawberries are a type of accessory fruit, me...
Watermelon   China       30  Watermelon is a sweet, refreshing summertime f...
```

Query the data with Agent

```
from langchain.agents import create_pandas_dataframe_agent
csv_agent = create_pandas_dataframe_agent(llm, df)
print(csv_agent.run("how many rows are there?"))
print(csv_agent.run("which fruit with the lowest calories and what country produce it most?"))
```

```
the result is:
There are 5 rows in the dataframe.
Watermelon from China has the lowest calories at 30.
```

Learn More

LangChain Documentations / GitHub Repo