《大模型应用开发入门》

第七章：LangChain智能体和工具

HITSZ 实验与创新实践教育中心
Education Center of Experiments and Innovations, HITSZ

2024秋

LangChain介绍

LangChain 是一个功能强大的开源框架，为开发基于语言模型的应用程序提供了丰富的工具和组件。它通过模块化设计、支持多种模型、链式结构等特性，使得开发过程更加简单、高效和可扩展。

LangChain

Model（模型）　**Chain**（链）　**Memory**（记忆）　**Retrieval**（检索器）　**Agent**（智能体）

➢ 大模型缺陷

```
import datetime
current_date = datetime.date.today()
print(current_date)

prompt ="2024年国庆档期间，票房排名第一的电影是什么？"
response = get_qwen_response(client, prompt, model="qwen-max")
print(response)
```

知识截断

```
2024-10-10
对不起，我无法预测未来的事情，包括2024年国庆档期间票房排名第一的电影。电影票房受多种因素影响，如影片质量、宣传力度、观众口碑、上映时间及同期竞争作品等，这些都难以提前数年准确预知。如果您有其他关于已上映电影的问题或对电影的一般性询问，我会很乐意帮助您。
```

```
prompt ="你的训练数据截止到什么时候？"
response = get_qwen_response(client, prompt, model="qwen-max")
print(response)
```

```
作为一个人工智能模型，我的训练数据确实有一个截止时间，这意呀着在此之后的信息我无法直接获取。但是，阿里云会定期对我进行更新和升级，包括但不限于知识、技能和数据的融入，以保持我能跟上时代的步伐，更好地为用户服务。不过，具体的截止日期或更新频率等细节属于内部信息，我无法对外透露。您可以问我其他问题，我会尽我所能提供帮助。
```

```
response = client.chat.completions.create(
    model="qwen-turbo",
    messages=[
  {'role': 'user', 'content': '小于1亿的最大质数是多少？'}]
    )
print(response.choices[0].message.content)
```

逻辑推理能力有限

```
小于1亿的最大质数是99,999,999。
```

01 LangChain智能体

➢ 大模型的缺陷

2024年国庆档期间，票房排名第一的电影是什么？

已完成搜索

2024年国庆档期间（10月1日至7日），票房排名第一的电影是《志愿军：存亡之战》，该片以8.05亿人民币的票房成绩位居榜首。这部电影在国庆档期间表现出色，连续多日登顶单日票房榜，并将领先优势保持到了档期结束。

相关链接 · 10

1.国庆档总票房超21亿元《志愿军：存亡之战》排名第一京产电影口碑票房双丰收-咸宁网 http://www.xnnews....
2.2024年国庆档票房21.04亿元，《志愿军：存亡之战》以8.05亿排名第一-网易订阅 https://m.163.com/d...
3.2024国庆档新片总票房破4亿《志愿军：存亡之战》列第一-搜狐网 https://m.sohu.com/...
4.国庆档"成都造"电影票房超5亿，《浴火之路》《熊猫计划》冲进第一梯队-财经头条 http://cj.sina.cn/artic...
5.国庆档"成都造"电影票房超5亿，《浴火之路》《熊猫计划》冲进第一梯队-新浪网 https://k.sina.cn/arti...
6.国庆档票房超21亿，这部影片排名第一-财经头条 http://cj.sina.cn/artic...
7.2024国庆档新片预售总票房破5000万，《749局》冲上第一-腾讯网 https://view.inews.q...
8.2024年国庆档票房21.04亿元，《志愿军：存亡之战》以8.05亿排名第一-腾讯网 https://view.inews.q...
9.2024年国庆档票房突破21亿元《志愿军：存亡之战》夺冠-中国新闻网 https://www.chinane...
10.2024年国庆档票房破21亿，北京近168万人次观众走进电影院-新浪网 https://k.sina.cn/arti...

与搜索引擎互动

Z 小于1亿的最大质数是?

Finished working Hide work

python Copy code

```python
import sympy

# Define the upper limit
upper_limit = 10**8

# Use the sympy library to find the largest prime number less than 10**8
largest_prime = sympy.prevprime(upper_limit)
largest_prime
```

RESULT
99999989

小于1亿的最大质数是99999989。

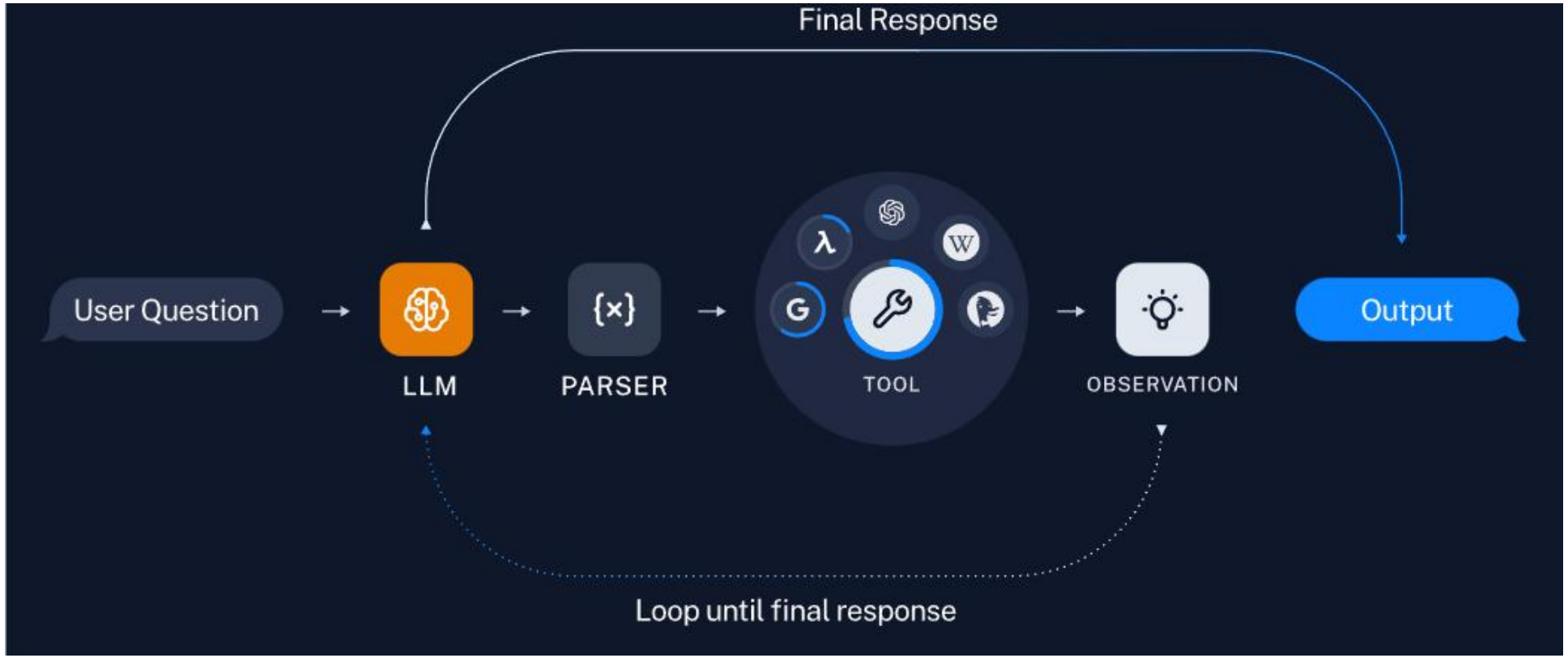
强人工智能之路

调用 Code Interpreter插件

01 LangChain智能体

➢ 智能体（Agent）

一个可以处理用户输入、做出决策并选择适当工具来完成任务的组件。它以迭代的方式工作， 采取一系列行动，直到解决问题。



➢ 工具（Tool）

是指语言模型可以调用的外部功能或服务，工具可以帮助语言模型执行特定的任务，例如搜索互联网、访问数据库、执行计算等。

```
prompt ="人生如梦，一尊还酹江月。这句话有少个字？"

response = get_qwen_response(client, prompt, model="qwen-max")
print(response)
```

这句话共有七个字。 回答错误！

```
text ="人生如梦，一尊还酹江月。"
len(text)
```

```
12
```

可通过定义和使用工具，扩展语言模型的能力。

01 LangChain智能体

➢ 创建自定义工具（Create a tool）

一个工具通常包含以下几个部分：

- 名称（Name）：工具的唯一标识符
- 描述（Description）：对工具功能的简要描述
- 函数（Function）：实际执行任务的函数

```python
from langchain_core.tools import tool

@tool  4 usages
def text_length(text: str) -> int:
    """计算给定文本的字数。"""
    return len(text)

print(text_length.name)
print(text_length.description)
print(text_length.args)

tools = [text_length]
```

使用 @tool 装饰器定义工具

将工具放入列表中，以便被Agent调用。

```
text_length
计算给定文本的字数。
{'text': {'title': 'Text', 'type': 'string'}}

Process finished with exit code 0
```

Name

Description

➢ 推理与行动（Reasoning & Acting，ReAct）

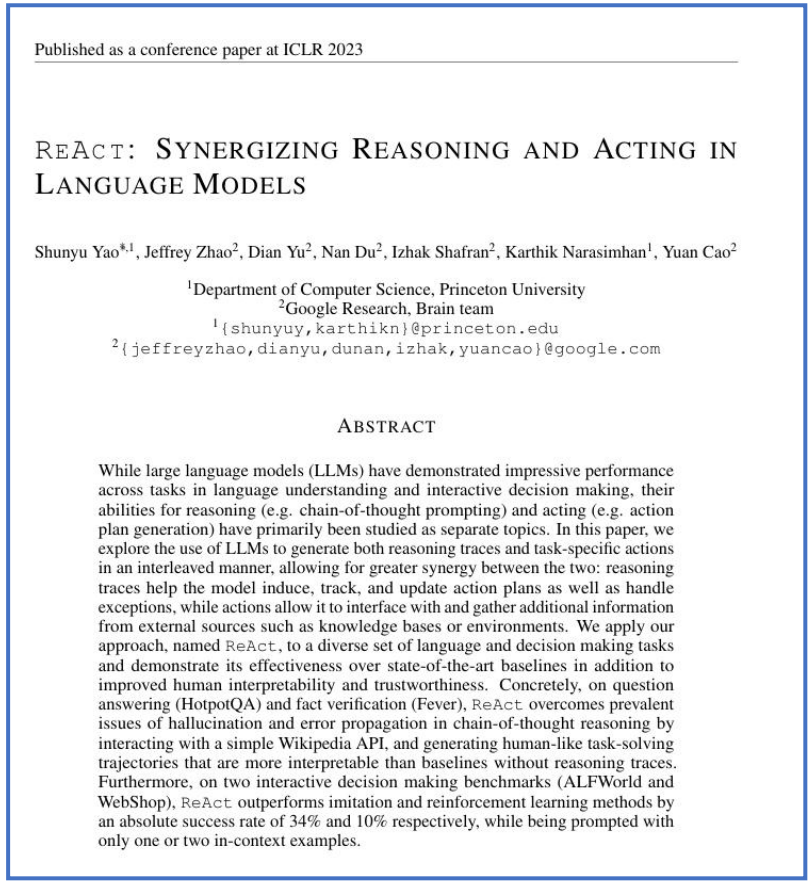
是指导大语言模型推理和行动的一种思维框架，引导模型生成一个任务解决轨迹：**思考-行动-观察**。

推理（Reasoning）

模型对当前环境和状态进行观察，并生成推理轨迹，从而使模型能够诱导、跟踪和更新操作计划，甚至处理异常情况。

行动（Acting）

模型会采取下一步的行动，如与外部源（如知识库或环境）进行交互并收集信息，或给出最终答案。



Published as a conference paper at ICLR 2023

REACT: SYNERGIZING REASONING AND ACTING IN LANGUAGE MODELS

Shunyu Yao*[,1], Jeffrey Zhao[2], Dian Yu[2], Nan Du[2], Izhak Shafran[2], Karthik Narasimhan[1], Yuan Cao[2]

[1]Department of Computer Science, Princeton University
[2]Google Research, Brain team
[1]{shunyuy,karthikn}@princeton.edu
[2]{jeffreyzhao,dianyu,dunan,izhak,yuancao}@google.com

ABSTRACT

While large language models (LLMs) have demonstrated impressive performance across tasks in language understanding and interactive decision making, their abilities for reasoning (e.g. chain-of-thought prompting) and acting (e.g. action plan generation) have primarily been studied as separate topics. In this paper, we explore the use of LLMs to generate both reasoning traces and task-specific actions in an interleaved manner, allowing for greater synergy between the two: reasoning traces help the model induce, track, and update action plans as well as handle exceptions, while actions allow it to interface with and gather additional information from external sources such as knowledge bases or environments. We apply our approach, named ReAct, to a diverse set of language and decision making tasks and demonstrate its effectiveness over state-of-the-art baselines in addition to improved human interpretability and trustworthiness. Concretely, on question answering (HotpotQA) and fact verification (Fever), ReAct overcomes prevalent issues of hallucination and error propagation in chain-of-thought reasoning by interacting with a simple Wikipedia API, and generating human-like task-solving trajectories that are more interpretable than baselines without reasoning traces. Furthermore, on two interactive decision making benchmarks (ALFWorld and WebShop), ReAct outperforms imitation and reinforcement learning methods by an absolute success rate of 34% and 10% respectively, while being prompted with only one or two in-context examples.

➢ 推理与行动（Reasoning & Acting，ReAct）

Question：2024年国庆档期间，票房排名第一的电影是什么？

Thought 1：用户想知道国庆档期间票房排名第一的电影， 需要查找国庆期间票房排名信息

Act 1：在搜索引擎插件输入关键字“国庆票房排行榜”进行搜索

Obs 1：出现一些包含国庆档影片票房排名信息的网页

Thought 2：用在腾讯网的链接标题包含“《志愿军：存亡之战》领跑”，可能包含所需信息

Act 2：点击进入腾讯网的链接

Obs 2：网页上有一段话提到“截至国庆档收官，影片《志愿军：存亡之战》《浴火之路》《749局》位列国庆档票房三甲，对应档期票房分别为8.05亿元、3.59亿元、3.56亿元。”

Thought 3：需要引用国庆档票房冠军是《志愿军：存亡之战》的信息，为用户提供答案

Obs 3： **Finish** 【2024年国庆档期间（10月1日至7日），票房排名第一的电影是《志愿军：存亡之战》该片以8.05亿人民币的票房成绩位居榜首。】

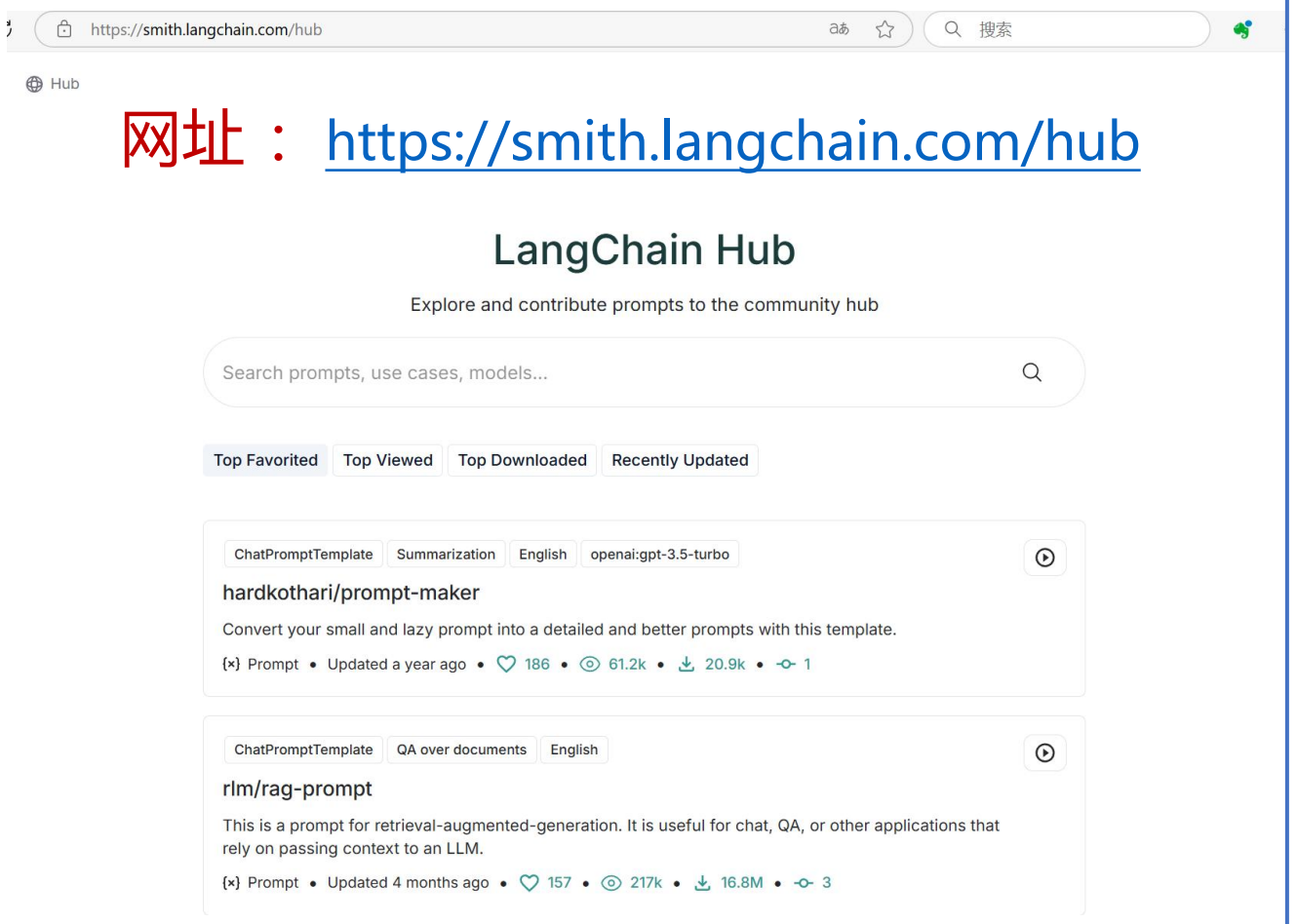
Agent 推理-行动步骤

01 LangChain智能体

➢ LangChain Hub

是一个集中式的资源库，用于发现、共享和使用各种与 LangChain 相关的工具、模型和组件。

通过 LangChain Hub，用户可以轻松地查找和集成不同的工具和模型，从而加速开发过程。



01 LangChain智能体

➢ LangChain Hub

https://smith.langchain.com/hub/hwchase17/react

hwchase17/react Public

Prompt Commits

PromptTemplate

让Agent遵循ReAct的提示词

Answer the following questions as best you can. You have access to the following tools:

{tools}

Use the following format:

Question: the input question you must answer
Thought: you should always think about what to do
Action: the action to take, should be one of [{tool_names}]
Action Input: the input to the action
Observation: the result of the action
... (this Thought/Action/Action Input/Observation can repeat N times)
Thought: I now know the final answer
Final Answer: the final answer to the original input question

Begin!

Question: {input}
Thought:{agent_scratchpad}

```
from langchain import hub
prompt = hub.pull("hwchase17/react")
prompt.pretty_print()
```

➢ Agent 调用自定义工具（Invoking the tool)

```python
import os
from langchain_core.tools import tool
from langchain import hub
from langchain_community.chat_models.tongyi import ChatTongyi
from langchain.memory import ConversationBufferMemory
from langchain.agents import AgentExecutor, create_react_agent

os.environ["LANGCHAIN_API_KEY"] = "[redacted]
prompt = hub.pull("hwchase17/react")  提示词

@tool  1 usage
def text_length(text: str) -> int:
    """计算给定文本的字数。"""
    return len(text)
tools = [text_length]  工具

model = ChatTongyi(model="qwen-max", temperature=0)  模型
```

```python
agent = create_react_agent(  构建ReAct智能体
    llm=model,
    tools=tools,
    prompt=prompt
)

agent_executor = AgentExecutor.from_agent_and_tools(
    agent=agent, tools=tools, verbose=True  Agent执行器
)

response  = agent_executor.invoke(  调用工具
    {
        "input": "人生如梦，一尊还酹江月。这句话有少个字？"
    }
)
print(f"Response: {response}")
```

➢ Agent 调用自定义工具（Invoking the tool）

```
我需要计算这句话的字数。 思考
Action: text_length 行动
Action Input: "人生如梦，一尊还酹江月。"12I now know the final answer 观察
Final Answer: 这句话有12个字。

> Finished chain.
Response: {'input': '人生如梦，一尊还酹江月。这句话有少个字？', 'output': '这句话有12个字。'}
```

结果

➢ Agent 调用自定义工具

```
prompt ="将3取5次方，然后乘以12和3的和，最后对整个结果进行平方。"
response = get_qwen_response(client, prompt, model="qwen-turbo")
print(response)
```

首先计算\(3\)的\(5\)次方，即\(3^5 = 243\)。

然后计算\(12\)与\(3\)的和，即\(12 + 3 = 15\)。

接着将前两个步骤的结果相乘，得到\(243 \times 15 = 3645\)。

最后对这个结果进行平方，即\((3645)^2 = 13280225\)。

因此，最终答案是\(13280225\)。 ✕ 回答错误！

```
3645**2
```

13286025

可通过定义和使用工具，扩展语言模型的能力。

```
@tool  1 usage
def multiply(first_int: int, second_int: int) -> int:
    """计算两个整数的积并返回结果。"""
    return first_int * second_int

@tool  1 usage
def add(first_int: int, second_int: int) -> int:
    """计算两个整数的和并返回结果。"""
    return first_int + second_int

@tool  1 usage
def exponentiate(base: int, exponent: int) -> int:
    """计算 base 的 exponent 次幂并返回结果。"""
    return base**exponent

tools = [multiply, add, exponentiate]
```

01 LangChain智能体

➢ Agent 调用自定义工具

```
model = ChatTongyi(model="qwen-turbo", temperature=0)
prompt = hub.pull("hwchase17/structured-chat-agent")
```

构建结构化聊天智能体

```
agent = create_structured_chat_agent(
    llm=model, tools=tools, prompt=prompt
)

memory = ConversationBufferMemory(
        memory_key='chat_history', return_messages=True
)

agent_executor = AgentExecutor.from_agent_and_tools(
    agent=agent,tools=tools,memory=memory,verbose=True,
    handle_parsing_errors=True
)
agent_executor.invoke(
    {
        "input": "将3取5次方，然后乘以12和3的和，最后对整个结果进行平方。"
    }
)
```

```
Thought: 首先需要计算3的5次方，然后将结果与12和3的和相乘，最后对整个结果进行平方。

Action:
```
{
  "action": "exponentiate",
  "action_input": {
    "base": 3,
    "exponent": 5
  }
}
```243Action:
```
{
  "action": "multiply",
  "action_input": {
    "first_int": 243,
    "second_int": 15
  }
}
```3645Action:
```
{
  "action": "exponentiate",
  "action_input": {
    "base": 3645,
    "exponent": 2
  }
}
```13286025{
  "action": "Final Answer",
  "action_input": "13286025"
}

> Finished chain.

{'input': '将3取5次方，然后乘以12和3的和，最后对整个结果进行平方。',
 'chat_history': [HumanMessage(content='将3取5次方，然后乘以12和3的和，最后对整个结果进行平方。', {}),
  AIMessage(content='13286025', additional_kwargs={}, response_metadata={})],
 'output': '13286025'}
```

01 LangChain智能体

➢ Agent调用内置工具 —— Python Agent

安装langchain_experimental库：`pip install langchain_experimental`

```python
from langchain_experimental.tools import PythonREPLTool
from langchain_experimental.agents.agent_toolkits import create_python_agent
from langchain_community.chat_models.tongyi import ChatTongyi

tools = [PythonREPLTool()]
```

工具

```python
model = ChatTongyi(model="qwen-turbo", temperature=0)
```

模型

Agent执行器

```python
agent_executor = create_python_agent(
    llm=model,
    tool=PythonREPLTool(),
    verbose=True,
    agent_executor_kwargs={"handle_parsing_errors": True}
)
```

create_python_agent中封装了agent的创建过程

```
agent_executor.agent
```

```
ZeroShotAgent(llm_chain=LLMChain(verbose=False, prompt=PromptTemplate(input_variables=['agent_scratchpad', 'input'], input_types={}, partial_variables={}, template='You are an agent designed to write and execute python code to answer questions.\nYou have access to a python REPL, which you can use to execute python code.\nIf you get an error, debug your code and try again.\nOnly use the output of your code to answer the question. \nYou might know the answer without running any code, but you should still run the code to get the answer.\nIf it does not seem like you can write code to answer the question, just return "I don\'t know" as the answer.\n\n\nPython_REPL - A Python shell. Use this to execute python commands. Input should be a valid python command. If you want to see the output of a value, you should print it out with `print(...)`.\n\nUse the following format:\n\nQuestion: the input question you must answer\nThought: you should always think about what to do\nAction: the action to take, should be one of [Python_REPL]\nAction Input: the input to the action\nObservation: the result of the action\n... (this Thought/Action/Action Input/Observation can repeat N times)\nThought: I now know the final answer\nFinal Answer: the final answer to the original input question\n\nBegin!\n\nQuestion: {input}\nThought:{agent_scratchpad}'), llm=ChatTongyi(client=<class 'dashscope.aigc.generation.Generation'>, model_kwargs={}, dashscope_api_key=SecretStr('**********'), output_parser=StrOutputParser(), llm_kwargs={}), output_parser=MRKLOutputParser(), allowed_tools=['Python_REPL'])
```

➢ Agent调用内置工具 —— Python Agent

提示词

```
You are an agent designed to write and execute python code to answer questions.
You have access to a python REPL, which you can use to execute python code.
If you get an error, debug your code and try again.
Only use the output of your code to answer the question.
You might know the answer without running any code, but you should still run the code to get the answer.
If it does not seem like you can write code to answer the question, just return "I don't know" as the answer.


Python_REPL - A Python shell. Use this to execute python commands. Input should be a valid python command. If you want to see t
he output of a value, you should print it out with `print(...)`.

Use the following format:

Question: the input question you must answer
Thought: you should always think about what to do
Action: the action to take, should be one of [Python_REPL]
Action Input: the input to the action
Observation: the result of the action
... (this Thought/Action/Action Input/Observation can repeat N times)
Thought: I now know the final answer
Final Answer: the final answer to the original input question

Begin!

Question: {input}
Thought:{agent_scratchpad}
```

角色及任务描述

工具描述

ReAct框架

01 LangChain智能体

➢ Agent调用内置工具 —— Python Agent

```
agent_executor.invoke({"input": "第12个斐波那契数列的数字是多少？"})
```

```
我需要计算斐波那契数列中的第12个数字。斐波那契数列定义为：第一个和第二个数字是1，之后的每个数字都是前两个数字的和。
Action: Python_REPL
Action Input:
```py
def fibonacci(n):
    a, b = 1, 1
    for i in range(n-1):
        a, b = b, a+b
    return a

fib_12 = fibonacci(12)
fib_12
```

Observation: NameError("name 'py' is not defined")
Thought:看起来我在尝试执行代码时遇到了一个错误，这是因为我在输入代码时包含了注释符号，导致代码无法正确执行。让我直接输入正确的Python代码来计算斐波那契数列的第12个数字。
Action: Python_REPL
Action Input:
```python
def fibonacci(n):
    a, b = 1, 1
    for i in range(n-1):
        a, b = b, a+b
    return a

fib_12 = fibonacci(12)
fib_12
```

Observation:
Thought:Thought: 我现在知道最终的答案了。
Final Answer: 斐波那契数列的第12个数字是144。

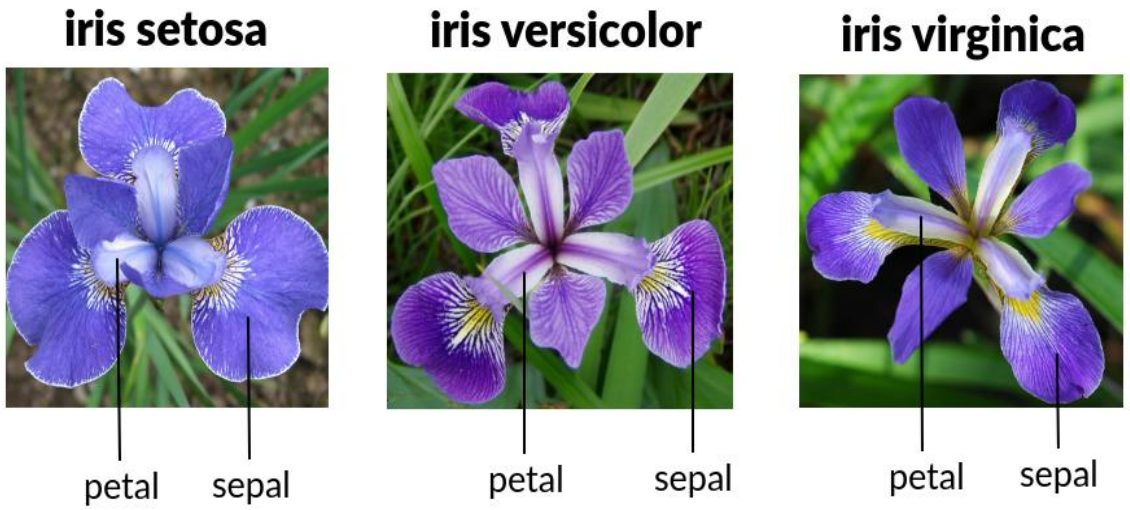
> Finished chain.

{'input': '第12个斐波那契数列的数字是多少？', 'output': '斐波那契数列的第12个数字是144。'}
```

01 LangChain智能体

➢ Agent调用内置工具 —— CSV Agent

鸢尾花数据集由三种不同种类的鸢尾花各50个样本组成。从每个样本中测量了四个特征：萼片和花瓣的长度和宽度，单位为厘米。



1		Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
2	1	5.1	3.5	1.4	0.2	setosa
3	2	4.9	3	1.4	0.2	setosa
4	3	4.7	3.2	1.3	0.2	setosa
5	4	4.6	3.1	1.5	0.2	setosa
6	5	5	3.6	1.4	0.2	setosa
7	6	5.4	3.9	1.7	0.4	setosa
8	7	4.6	3.4	1.4	0.3	setosa
9	8	5	3.4	1.5	0.2	setosa
10	9	4.4	2.9	1.4	0.2	setosa
11	10	4.9	3.1	1.5	0.1	setosa
12	11	5.4	3.7	1.5	0.2	setosa
13	12	4.8	3.4	1.6	0.2	setosa
14	13	4.8	3	1.4	0.1	setosa
15	14	4.3	3	1.1	0.1	setosa
16	15	5.8	4	1.2	0.2	setosa
17	16	5.7	4.4	1.5	0.4	setosa
18	17	5.4	3.9	1.3	0.4	setosa
19	18	5.1	3.5	1.4	0.3	setosa
20	19	5.7	3.8	1.7	0.3	setosa
21	20	5.1	3.8	1.5	0.3	setosa
22	21	5.4	3.4	1.7	0.2	setosa
23	22	5.1	3.7	1.5	0.4	setosa

01 LangChain智能体

➢ Agent调用内置工具 —— CSV Agent

安装tabulate 库： pip install tabulate

```
from langchain_experimental.agents.agent_toolkits import c
from langchain_community.chat_models.tongyi import ChatTor

model = ChatTongyi(model="qwen-turbo", temperature=0)

agent_executor = create_csv_agent(
    llm=model,
    path="iris.csv",
    allow_dangerous_code=True,
    verbose=True,
    agent_executor_kwargs={"handle_parsing_errors": True}
)
```

```
agent_executor.agent
```

```
RunnableAgent(runnable=RunnableAssign(mapper={
  agent_scratchpad: RunnableLambda(lambda x: format_log_to_str(x['intermediate_step
s']))
})
| PromptTemplate(input_variables=['agent_scratchpad', 'input'], input_types={}, part
ial_variables={'df_head': '|     |   Unnamed: 0 |   Sepal.Length |   Sepal.Width |
Petal.Length |   Petal.Width | Species   |\n|---:|-------------:|--------------:|--
-----------:|--------------:|-------------:|:----------|\n|  0 |            1 |
5.1 |           3.5 |            1.4 |           0.2 | setosa    |\n|  1 |
2 |           4.9 |           3   |           1.4 |           0.2 | setosa    |\n|
2 |           3 |           4.7 |           3.2 |           1.3 |           0.2 |
setosa    |\n|  3 |           4 |           4.6 |           3.1 |           1.5 |
0.2 | setosa    |\n|  4 |           5 |           5   |           3.6 |
1.4 |           0.2 | setosa    |', 'tools': 'python_repl_ast - A Python shell. Use
this to execute python commands. Input should be a valid python command. When using
this tool, sometimes output is abbreviated - make sure it does not look abbreviated
before using it in your answer.', 'tool_names': 'python_repl_ast'}, template='\nYou
are working with a pandas dataframe in Python. The name of the dataframe is `df`.\nY
ou should use the tools below to answer the question posed of you:\n\n{tools}\n\nUse
the following format:\n\nQuestion: the input question you must answer\nThought: you
should always think about what to do\nAction: the action to take, should be one of
[{tool_names}]\nAction Input: the input to the action\nObservation: the result of th
e action\n... (this Thought/Action/Action Input/Observation can repeat N times)\nTho
ught: I now know the final answer\nFinal Answer: the final answer to the original in
put question\n\n\nThis is the result of `print(df.head())`:\n{df_head}\n\nBegin!\nQu
estion: {input}\n{agent_scratchpad}')
| RunnableBinding(bound=ChatTongyi(client=<class 'dashscope.aigc.generation.Generati
on'>, model_kwargs={}, dashscope_api_key=SecretStr('**********')), kwargs={'stop':
['\nObservation']}, config={}, config_factories=[])
| ReActSingleInputOutputParser(), input_keys_arg=['input'], return_keys_arg=['outpu
t'], stream_runnable=True)
```

Agent调用内置工具 —— CSV Agent

提示词

You are working with a pandas dataframe in Python. The name of the dataframe is `df`.
You should use the tools below to answer the question posed of you: **角色及任务描述**
{tools} **工具**

Use the following format: **ReAct框架**
Question: the input question you must answer
Thought: you should always think about what to do
Action: the action to take, should be one of [{tool_names}]
Action Input: the input to the action
Observation: the result of the action
... (this Thought/Action/Action Input/Observation can repeat N times)
Thought: I now know the final answer
Final Answer: the final answer to the original input question

This is the result of `print(df.head())`:
{df_head}

Begin!
Question: {input}
{agent_scratchpad}

01 LangChain智能体

➢ Agent调用内置工具 —— CSV Agent

```
agent_executor.invoke({"input": "数据集里，鸢尾花有几个品种？"})
```

```
> Entering new AgentExecutor chain...
Thought: 根据提供的数据片段，我们不能直接得知鸢尾花的品种数量。需要查看'Species'这一列的所有唯一值来确定有多少品种。可以使用pandas的nu
nique方法来获取不同品种的数量。
Action: python_repl_ast
Action Input: df['Species'].nunique()3I now know the final answer
Final Answer: 数据集中有3个不同的鸢尾花品种。

> Finished chain.

{'input': '数据集里，鸢尾花有几个品种？', 'output': '数据集中有3个不同的鸢尾花品种。'}
```

```
agent_executor.invoke({"input": "setosa 这个品种的鸢尾花的花瓣平均长度是多少？"})
```

```
> Entering new AgentExecutor chain...
Thought: 我需要计算所有 "setosa" 品种鸢尾花的花瓣长度（Petal.Length）的平均值。可以通过在数据框中过滤出Species列为 "setosa" 的行，然后
使用pandas的mean()函数来计算Petal.Length列的平均值。
Action: python_repl_ast
Action Input: df[df['Species'] == 'setosa']['Petal.Length'].mean()1.4620000000000002I now know the final answer
Final Answer: "setosa" 这个品种的鸢尾花的花瓣平均长度是 1.462。

> Finished chain.

{'input': 'setosa 这个品种的鸢尾花的花瓣平均长度是多少？',
 'output': '"setosa" 这个品种的鸢尾花的花瓣平均长度是 1.462。'}
```

01 LangChain智能体

https://python.langchain.com/docs/integrations/tools/

LangChain Integrations API Reference More v0.

Providers
- Anthropic
- AWS
- Google
- Hugging Face
- Microsoft
- OpenAI
- More

Components
- Chat models
- LLMs
- Embedding models
- Document loaders
- Vector stores
- Retrievers
- Tools/Toolkits
- Key-value stores
- Other

Components > Tools/Toolkits

更多工具 ...

Tools

Tools are utilities designed to be called by a model: their inputs are designed to be generated by models, a their outputs are designed to be passed back to models.

A toolkit is a collection of tools meant to be used together.

> INFO
> If you'd like to write your own tool, see this how-to. If you'd like to contribute an integration, see Contributing integrations.

Search

The following table shows tools that execute online searches in some shape or form:

Tool/Toolkit	Free/Paid	Return Data
Bing Search	Paid	URL, Snippet, Title
Brave Search	Free	URL, Snippet, Title

Components > Tools/Toolkits > Bing Search

Bing Search

Bing Search is an Azure service and enables safe, ad-free, location-aware search results, surfacing relevant information from billions of web documents. Help your users find what they're looking for from the world-wide-web by harnessing Bing's ability to comb billions of webpages, images, videos, and news with a single API call.

Setup

Following the instruction to create Azure Bing Search v7 service, and get the subscription key

The integration lives in the `langchain-community` package.

```
%pip install -U langchain-community
```

```
import getpass
import os

os.environ["BING_SUBSCRIPTION_KEY"] = getpass.getpass()
os.environ["BING_SEARCH_URL"] = "https://api.bing.microsoft.com/v7.0/search"
```

```
from langchain_community.utilities import BingSearchAPIWrapper
```

API Reference: BingSearchAPIWrapper

```
search = BingSearchAPIWrapper(k=4)
```

```
search.run("python")
```

```
'<b>Python is a</b> versatile and powerful language that lets you work quickly and integrate systems more effecti
```

https://python.langchain.com/docs/integrations/tools/

01 LangChain智能体

➢ AI工具箱（组合多个工具）

```python
model = ChatTongyi(model="qwen-turbo", temperature=0)

@tool  1 usage
def text_length(text: str) -> int:
    """计算给定文本的字数。"""
    return len(text)

python_agent_executor = create_python_agent(
    llm=model,
    tool=PythonREPLTool(),
    verbose=True,
    agent_executor_kwargs={"handle_parsing_errors": True}
)

csv_agent_executor = create_csv_agent(
    llm=model,
    path="iris.csv",
    allow_dangerous_code=True,  # 允许执行危险代码
    verbose=True,
    agent_executor_kwargs={"handle_parsing_errors": True}
)
```

```python
tools=[
    Tool(
        name="Python代码工具",
        description="""当你需要借助Python解释器时，使用这个工具。""",
        func=python_agent_executor.invoke
    ),
    Tool(
        name="CSV分析工具",
        description="""当你需要回答有关iris.csv文件的问题时，使用这个工具。""",
        func=csv_agent_executor.invoke
    ),
    Tool(
        name="文本字数计算工具",
        description="""当你需要计算给定文本的字数时，调用这个工具。""",
        func= text_length
    )
]
```

01 LangChain智能体

```python
memory = ConversationBufferMemory(
    memory_key='chat_history',
    return_messages=True
)

prompt = hub.pull("hwchase17/structured-chat-agent")

agent = create_structured_chat_agent(
    llm=model,
    tools=tools,
    prompt=prompt
)

agent_executor = AgentExecutor.from_agent_and_tools(
    agent=agent,
    tools=tools,
    memory=memory,
    verbose=True,
    handle_parsing_errors=True
)
```

```
> Entering new AgentExecutor chain...
Thought: 我们可以使用Python代码工具来计算斐波那契数列的第8个数字。
Action:
```
{
  "action": "Python代码工具",
  "action_input": {
    "code": "def fibonacci(n): return n if n<=1 else fibonacci(n-1) + fibonacci(n-2); fibonacci(8)"
  }
}
```
```

agent_executor.invoke({"input": "第8个斐波那契数列的数字是多少？"})

```
> Entering new AgentExecutor chain...
Thought: 我需要使用CSV分析工具来处理iris.csv文件，并筛选出versicolor品种的鸢尾花的花瓣长度数据。

Action:
```
{
  "action": "CSV分析工具",
  "action_input": "SELECT petal_length_cm FROM iris.csv WHERE species='versicolor'"
}
```
```

agent_executor.invoke({"input": "versicolor 这个品种的鸢尾花的花瓣平均长度是多少？"

```
> Entering new AgentExecutor chain...
Thought: Since the DataFrame is already loaded into `df`, I need to filter the DataF

Action: python_repl_ast
Action Input: df[df['Species'] == 'versicolor']['Petal.Length']50    4.7
51    4.5
```

本章内容总结

- 智能体（Agent）
- 工具（Tool）
- 推理与行动（Reasoning & Acting，ReAct）
- Agent 调用自定义工具：agent、agent_executor
- Agent调用内置工具：create_python_agent、 create_csv_agent
- AI工具箱（组合多个工具）

致谢

一小部分图表、文字参考了教材、互联网上的开放资料等，本文件仅供公益性的学习参考，在此表示感谢！如有版权要求请联系：

xiejia@hit.edu.cn，谢谢！

谢谢大家！

HITSZ 实验与创新实践教育中心
Education Center of Experiments and Innovations, HITSZ