

1. Langchain-agent代理

1.1 什么是agent代理

在Langchain中，Agent代理是一种智能化的计算机制，它能够根据输入的指令或环境上下文，动态选择和调用特定的工具（如搜索引擎、数据库、API等）来完成任

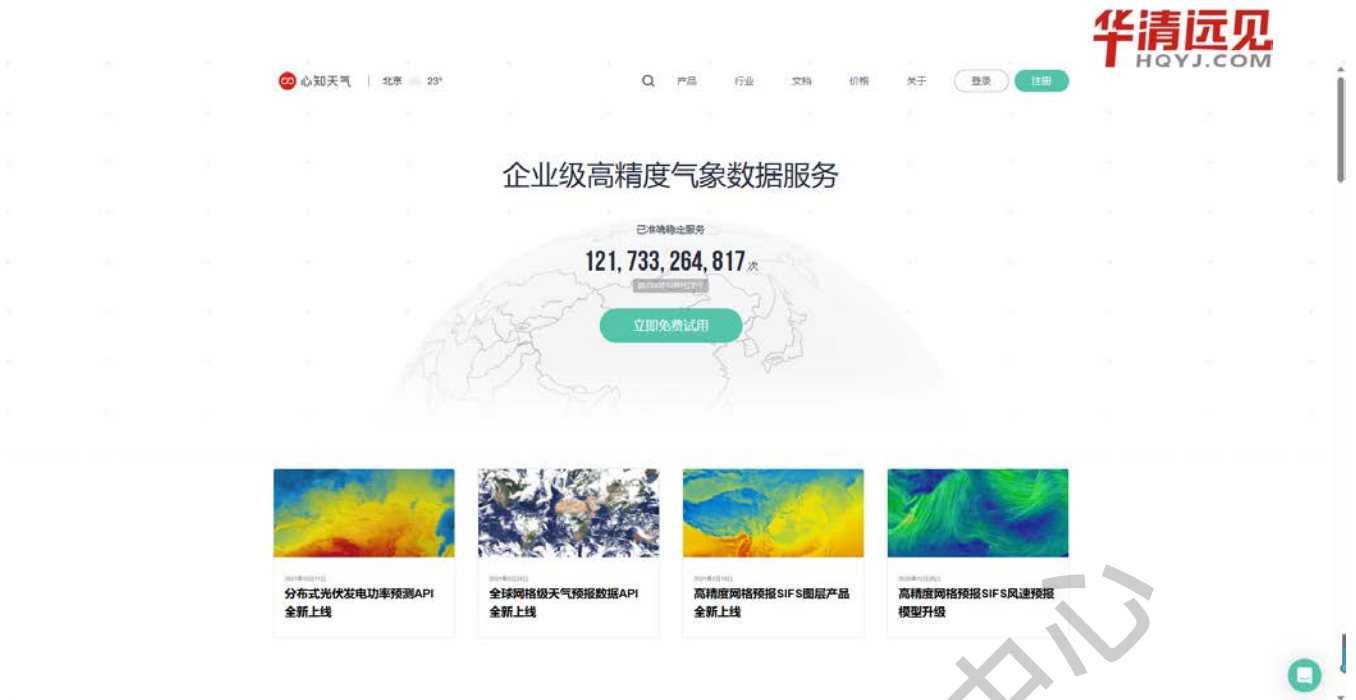
务。这种代理通过预先定义的逻辑流程或者学习到的策略，帮助开发者实现自动化、动态化和上下文敏感的计算与决策。

1.2 Langchain Agent代理例子

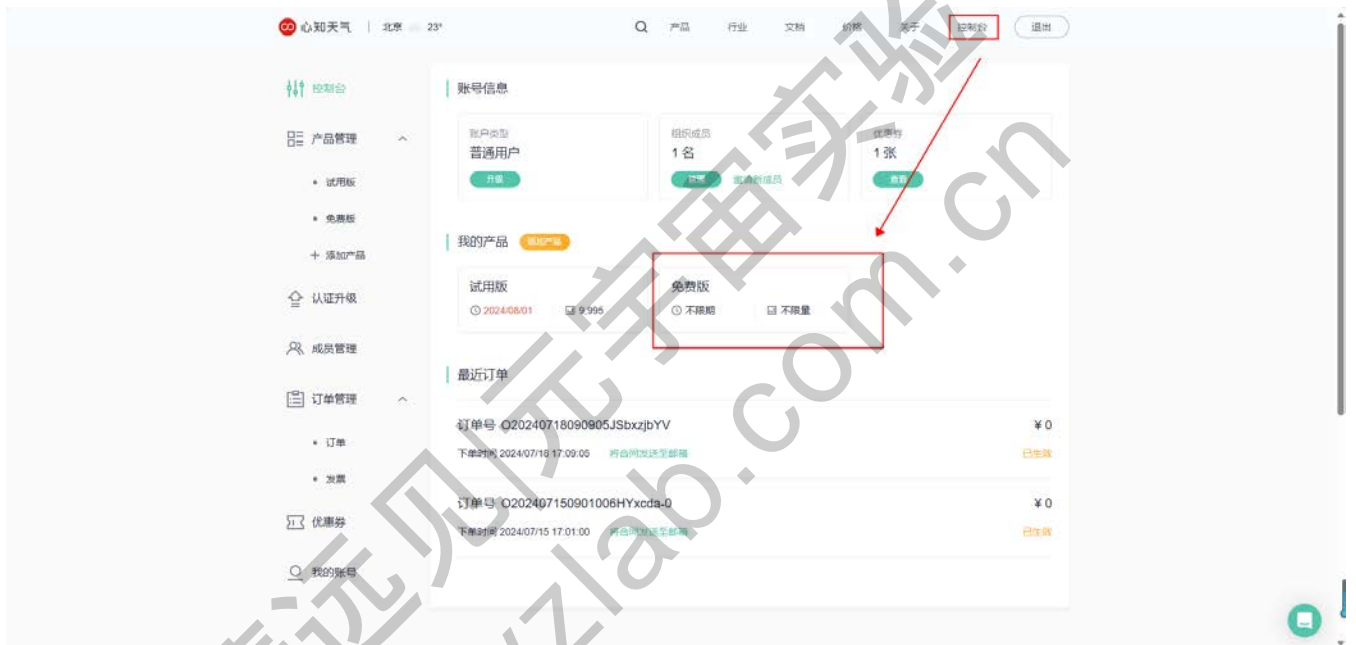
下面是一个使用Langchain Agent代理来查询天气信息的简单例子。在这个例子中，用户提出一个问题，Agent会根据任务内容调用天气API查询天气并给出最后结果。

首先我们需要申请一个api，这里以心知天气API为例。

进入官网 <https://www.seniverse.com/>。



注册并登录，点击控制台选择免费版



安装依赖

```
pip install requests
```

定义天气查看函数

```
# 心知天气API工具类
class WeatherTool:
    city: str = Field(description="City name, include city and county")

    def __init__(self, api_key):
        self.api_key = api_key

    def run(self, city):
```

```
city = city.split('\n')[0] # 清除多余的\n不然API会报错。
url = f"https://api.seniverse.com/v3/weather/now.json?key={api_key}&location={city}&language=zh-Hans&unit=c"
# 发送请求到心知天气API
print(url)
response = requests.get(url)
print(response.status_code)
# 如果请求成功，解析返回数据
if response.status_code == 200:
    data = response.json()
    # 获取天气信息
    weather = data['results'][0]['now']['text']
    temperature = data['results'][0]['now']['temperature']
    return f"{city}的天气是{weather}，温度为{temperature}°C"
else:
    return f"无法获取{city}的天气信息"
```

将天气查看工具封装成Langchain的Tool对象，name和description可以帮助agent找到工具。

```
tools = [
    Tool(
        name="weather check",
        func=weather_tool.run,
        description=""
    )
]
```

设置LLM，选择使用api大模型还是本地大模型。（较好的大模型具有更好的推理能力，agent表现更正常。）

白嫖API: <https://cloud.siliconflow.cn/>。

```
# 使用 OpenAI API 的 ChatOpenAI 模型
chat_model = ChatOpenAI(
    openai_api_key="EMPTY", # 替换为你的实际API密钥
    base_url="http://192.168.103.173:10259/v1",
    model="Qwen2___5-7B-Instruct"
)
```

设置agent的提示词，观察提示词我们可以发现agent的实现原理是基于REACT (Reason-acting) 的，即包括思考->观察->答案三步，并且这个过程可以重复直到得到理想的结果。

首先大模型拿到问题先思考应该如何解决是否调用工具调用什么工具，接着生成需要执行的动作Action，并分析动作的输入Action input是什么，然后分析得到的结果。

比如说这个例子中就是分析由调用API得到的天气信息，然后继续思考是否已经得到可以回答问题的答案，最后生成最终回答。

注意：确保模板中的关键字与代理期望的关键字一致。例如：

Thought: ...

Action: ...

Action Input: ...

Observation: ...

Thought: ...

Final Answer: ...

```
template = """请尽可能好地回答以下问题。如果需要，可以适当使用一些工具。
    你有以下工具可用：\n\n
    {tools}\n\n
    请使用以下格式：\n
    Question: 需要回答的问题\n
    Thought: 总是考虑应该做什么以及使用哪些工具。\n
    Action: 应采取的行动，应为 [{tool_names}] 中的一个\n
    Action Input: 行动的输入\n
    Observation: 行动的结果\n
    ... (这个 Thought/Action/Action Input/Observation
    过程可以重复零次或多次)\n
    Thought: 我现在知道最终答案了\n
    Final Answer: 对原问题的最终答案\n
    开始! \n\n
    Question: {input}\n\n
    Thought: {agent_scratchpad}\n"""
```

```
prompt = PromptTemplate.from_template(template)
```

创建一个agent，重点说一下stop_sequence，stop_sequence是指如何让模型输出停下来，为什么要停下来？

因为在大模型获得Action和Action input后需要进行正则化解析Action和Action input,然后调用对应的API, 此时就需要大模型停止生成。

这里的'\nObserv'是因为tokenizer分词的时候并没有分成完整的'\nObservation'因此无法被解析。

```
# 创建代理
agent = create_react_agent(chat_model, tools, prompt,
stop_sequence=["\nObserv"])
agent_executor = AgentExecutor.from_agent_and_tools(
    agent=agent,
    tools=tools,
    verbose=True, # 设置为True以启用详细输出
)
```

```
# 用户输入的查询
query = "北京天气怎么样?"

# 代理接收查询并调用工具
response = agent_executor.invoke({"input": query})

# 输出结果
print(response)
```

查看调试信息

```
> Entering new AgentExecutor chain...
我应该使用weather check工具来获取北京的天气情况。

Action: weather check
Action Input: 北京
北京的天气是晴，温度为13°C
Final Answer: 北京当前的天气是晴朗，气温为13°C。

> Finished chain.
{'input': '北京天气怎么样?', 'output': '北京当前的天气是晴朗，气温为13°C.'}
```

华清远见|元宇宙实验中心
yyzlab.com.cn