大模型应用系列——智能体（Agent）

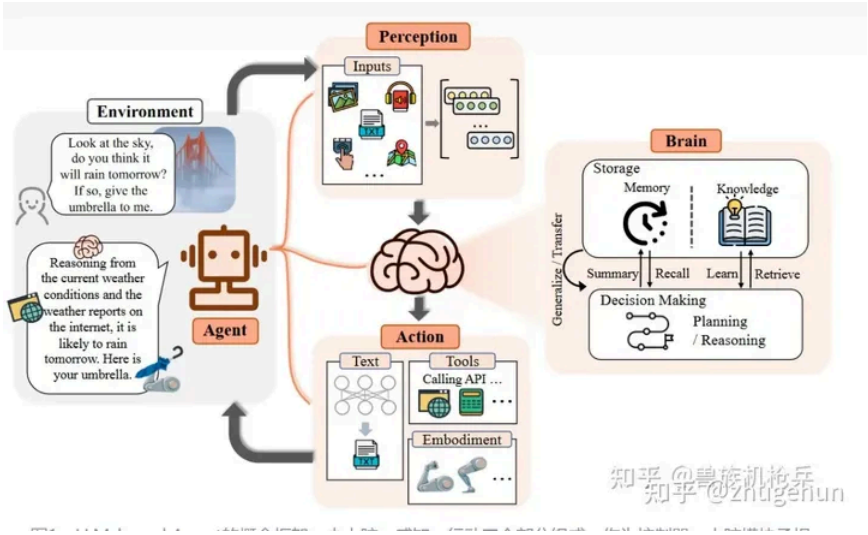
随着大模型思维链（CoT -Chain of Thought）的产生，基于大模型的智能体如雨后春笋般冒出来。基于大模型的智能体相比于以前智能体，最大变化是利用大模型可以实现智能体行为的 路径规划 。行为路径规划过程是利用的大模型思维链能力对用户不清晰或者复杂的指令进行步骤拆解，拆解成清晰的工具执行或者答案，并且可以根据工具执行结果或者环境的变化，重新使用大模型再进行新的路径规划，直到找到合适的路径。

现实情况中受限于大模型思维链的能力，能真正利用大模型思维链实现自动化分步骤完成复杂指令的情况还不多，更多还得借助于人工预先定义的规划路径。

随着LangChain 等开源组件不断成熟，使用LangChain + LLM来实现智能体逐渐变得简单，下文主要是通过实现LangChain中的Agent来介绍几种比较常见的智能体。除了LangChain可实现智能体之外，还有其他框架也支持智能体的实现如 LammaIndex、AutoGen、SuperAGI等。

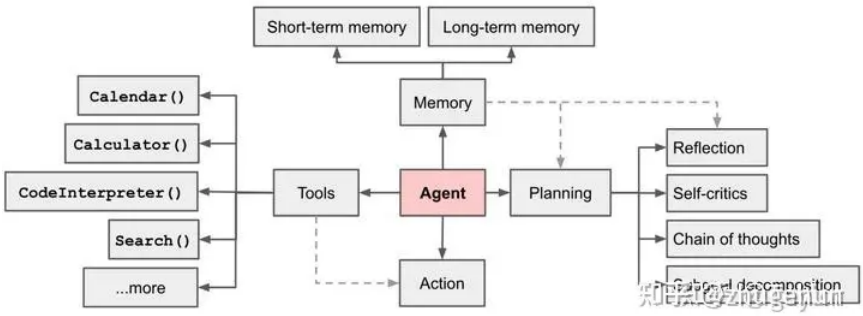
1、LLM-based Agent概念框架

LLM-based Agent概念框架由大脑、感知、行动三个部分组成。作为 控制器 ，大脑模块承担记忆、思考和决策等基本任务；感知模块负责感知和处理来自外部环境的多模态信息；行动模块负责使用工具执行任务并影响周围环境。



基于大模型智能体

Agent 是让 LLM 具备目标实现的能力，并通过自我激励循环来实现这个目标。要让LLM替代人去做事，我们可以基于PDCA模型进行 规划、执行、评估和反思。
规划能力（Plan）-> 分解任务：Agent大脑把大的任务拆解为更小的，可管理的子任务，这对有效的、可控的处理好大的复杂的任务效果很好。
执行能力（Done）-> 使用工具：Agent能学习到在模型内部知识不够时去调用外部API，比如：获取实时的信息、执行代码的能力、访问专有的信息知识库等等。
评估能力（Check ）-> 确认执行结果：Agent要能在任务正常执行后判断产出物是否符合目标，在发生异常时要能对异常进行分类（危害等级），对异常进行定位（哪个子任务产生的错误），对异常进行原因分析（什么导致的异常）。这个能力是通用大模型不具备的，需要针对不同场景训练独有的小模型。
反思能力（Action）-> 基于评估结果重新规划：Agent要能在产出物符合目标时及时结束任务，是整个流程最核心的部分；同时，进行归因分析总结导致成果的主要因素，另外，Agent要能在发生异常或产出物不符合目标时给出应对措施，并重新进行规划开启再循环过程。



智能体四个部分：大模型+工具+记忆+规划

2、LangChain中的Agent实现

2.1 使用LangChain实现零样本智能体

零样本智能体是指在不给大模型任何样本提示信息的情况，由大模型去实现工具调用来得到问题的答案，下面的示例就“姚明的妻子是谁？她现在的年龄是多少？她年龄的0.76次方是多少？”对零样本智能体进行演示。

- **安装Python包**

```
pip install langchain
pip install openai
pip install google-search-results
```

```
pip install -U duckduckgo-search
```

AI写代码

- **使用OpenAI大模型接口、Google搜索接口、数学接口实现零样本智能体**

```
import os
os.environ['OPENAI_API_KEY'] = "sk-rtyPFY3rItc5RB5prqsHT3BlbkFJkRho4VJhrckgCO****"
os.environ['SERPAPI_API_KEY'] = "e14c6e3d667fa9d988bf8966034e265a227e33985aba958a27c9e62852*****"
from langchain.agents import load_tools
from langchain.agents import initialize_agent
from langchain.agents import AgentType
from langchain.llms import OpenAI
# 加载将要用来控制 Agents 的语言模型
llm = OpenAI(temperature=0)
# 加载一些要使用的工具
tools = load_tools(["serpapi", "llm-math"], llm=llm)
# 初始化 Agents
agent = initialize_agent(tools, llm, agent=AgentType.ZERO_SHOT_REACT_DESCRIPTION, verbose=True)
# 测试一下智能体
agent.run("姚明的妻子是谁？她现在的年龄是多少？她年龄的0.76次方是多少？")
```

AI写代码

- **返回零样本智能体执行过程：**

```
Entering new AgentExecutor chain... I need to find out who Yao Ming's wife is and her age
Action: Search
Action Input: "Yao Ming's wife"
Observation: Ye Li
Thought: I need to find out her age
Action: Search
Action Input: "Ye Li age"
Observation: 42 years
Thought: I need to calculate her age to the 0.76 power
Action: Calculator
Action Input: 42^0.76
Observation: Answer: 17.126533197047873
Thought: I now know the final answer
Final Answer: Ye Li is Yao Ming's wife and her age to the 0.76 power is 17.126533197047873.
> Finished chain.
Ye Li is Yao Ming's wife and her age to the 0.76 power is 17.126533197047873.
```

AI写代码

- **自定义日期工具，通过零样本智能体调用自定义日期工具示例**

```
from langchain.agents import tool
from datetime import date
llm = OpenAI(temperature=0)
@tool
def time( text: str ) -> str:
 """
  Return the date of Today
  """
 return str(date.today())
agent = initialize_agent(tools+[time], llm, agent=AgentType.ZERO_SHOT_REACT_DESCRIPTION,
verbose=True)
agent.run("今天是哪天")
```

AI写代码

2.2、LCEL(LangChain Expression Language)介绍

最新的LangChain中LangChain 表达式语言（LCEL）已经成为了主流。LCEL是一种轻松地将链组合在一起的声明性方式。 LCEL 从第一天起就被设计为支持将原型投入生产，无需更改代码，从最简单的“提示 + LLM”链到最复杂的链。

- **LCEL基本样例: prompt + model + output parser**

```
from langchain_core.output_parsers import StrOutputParser
from langchain_core.prompts import ChatPromptTemplate
from langchain_openai import ChatOpenAI

prompt = ChatPromptTemplate.from_template("tell me a short joke about {topic}")
model = ChatOpenAI()
```

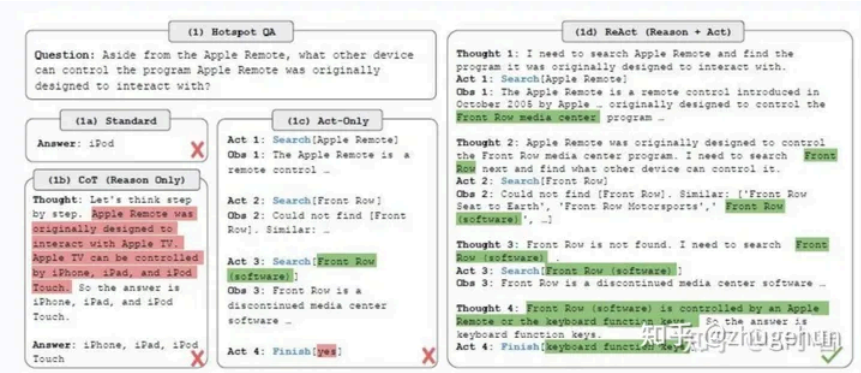
```
output_parser = StrOutputParser()

chain = prompt | model | output_parser
chain.invoke({"topic": "ice cream"})
```

AI写代码

2.3、对话型 Agent

对话型Agent是指通过对话的形式来实现智能体，支持对用户对话历史记录的保留。在介绍对话型Agent之前，还需要先介绍下ReAct ：结合推理和执行来克服大模型的胡言乱语情况。

2.3.1、React



ReAct 推理+执行举例

以下面这个问题为例：

除了Apple遥控器，还有什么其他设备可以控制相关软件？相关软件指的是，Apple遥控器最早可以控制的软件。

说明：

1.Apple遥控器最早只能控制Front Row软件。

2.Front Row软件可以被两种设备控制，Apple遥控器和键盘的功能键。

3.所以，正确答案是键盘的功能键。

传统方式1，直接问LLM。得到结果：

iPod。

答案错误，显然是在胡言乱语。因为LLM的内容生成，是根据之前所有的字预测后续内容，所以，我推测，出现这个结果很可能因为，LLM其实并不知道相关答案，只是Apple遥控器和iPod的语义很接近，所以才得出iPod这个结论。

传统方法2，思维链。得到结果：

想法：让我们一步一步思考这个问题。Apple遥控器可以控制Apple电视。Apple电视可以被iPhone、iPad和iPod Touch控制。
答案：iPhone、iPad和iPod Touch控制。

答案错误，思维链范式，其实也是在胡诌，因为它不知道答案。

传统方法3，动作范式。得到结果：

```
动作1：搜索【Apple 遥控器】
结果1：Apple遥控器是一种遥控器……最早可以控制“Front Row”……

动作2：搜索【Front Row】
结果2：无结果

动作3：搜索【Front Row软件】
结果3：FrontRow是一种过时软件……

动作4：完成【是的】
```

AI写代码

答案错误，动作范式，即隐式推理，也没有得到正确答案。不难理解，隐式推理对LLM的推理能力要求很高，因为他会隐去思考过程。我觉得当LLM推理能力达到一定程度时，应该也能得出正确答案。

ReAct方法，即推理+动作。得到结果：

```
想法1：我需要搜索Apple遥控器，并找到Apple遥控器最早可以控制的软件。
动作1：搜索【Apple 遥控器】
结果1：Apple遥控器是一种遥控器……最早可以控制“Front Row”……

```

```
想法2：Apple遥控器最早可以控制Front Row。我需要搜索Front Row下一步，找出还有什么其他设备可以控制他
动作2：搜索【Front Row】
结果2：无结果，可以试试“Front Row Seat to Earth”和“Front Row软件”

想法3：Front Row没有找到，我可以搜索“Front Row软件”
动作3：搜索【Front Row软件】
结果3：FrontRow是一种过时软件……可以被Apple遥控器和键盘功能键控制。

想法4：现在我知道答案了。
动作4：完成【键盘功能键】
```

AI写代码

答案正确。通过显式推理，加上动作，LLM Agent自主找到了答案。整个过程感觉就像是个不是特别聪明的孩子，要把想法写出来，然后去综合所有想法和观察，接着再做出相应的动作。但显然这个方法很有效，它最终找到了答案。

2.3.2、hwchase17/react-chat 提示词

React-chat提示词是别人总结好的ReAct的提示词，具体内容如下所示。

```
TOOLS:
------

Assistant has access to the following tools:

{tools}

To use a tool, please use the following format:

```
Thought: Do I need to use a tool? Yes
Action: the action to take, should be one of [{tool_names}]
Action Input: the input to the action
Observation: the result of the action
```

When you have a response to say to the Human, or if you do not need to use a tool, you MUST use the format:

```
Thought: Do I need to use a tool? No
Final Answer: [your response here]
```

Begin!

Previous conversation history:
{chat_history}

New input: {input}
{agent_scratchpad}
```

2.3.3、使用LangChain LCEL实现对话型Agent

```python
from langchain import hub
from langchain.agents.format_scratchpad import format_log_to_str
from langchain.agents.output_parsers import ReActSingleInputOutputParser
from langchain.tools.render import render_text_description
from langchain.memory import ConversationBufferMemory
#直接拉取别人写好的提示词
prompt = hub.pull("hwchase17/react-chat")
#提示词设置
prompt = prompt.partial(
    tools=render_text_description(tools),
    tool_names=", ".join([t.name for t in tools]),
)
llm_with_stop = llm.bind(stop=["\nObservation"])
#智能体设置
agent = (
    {
 "input": lambda x: x["input"],
 "agent_scratchpad": lambda x: format_log_to_str(x["intermediate_steps"]),
 "chat_history": lambda x: x["chat_history"],
    }
    | prompt
    | llm_with_stop
    | ReActSingleInputOutputParser()
)
from langchain.agents import AgentExecutor
#支持保存对话记录
memory = ConversationBufferMemory(memory_key="chat_history")
#智能体执行
agent_executor = AgentExecutor(agent=agent, tools=tools, verbose=True, memory=memory)
agent_executor.invoke({"input": "hi, i am bob"})["output"]
agent_executor.invoke({"input": "whats my name?"})["output"]
agent_executor.invoke({"input": "what are some movies showing 9/21/2023?"})["output"]
```

AI写代码

2.3.4、对话型Agent返回答案全流程

```
对话型Agent返回：
> Entering new AgentExecutor chain...
Thought: Do I need to use a tool? No
Final Answer: Hi Bob, nice to meet you! How can I help you today?
> Finished chain.
> Entering new AgentExecutor chain...
Thought: Do I need to use a tool? No
Final Answer: Your name is Bob!
> Finished chain.
> Entering new AgentExecutor chain...
Thought: Do I need to use a tool? Yes
Action: Search
Action Input: Movies showing 9/21/2023
['September 2023 Movies:  The Kill Room • The Creator • Expend4bles • Bottoms • PAW Patrol: The Mighty Movie • Dumb Money • A Haunting in Venice,
'August 2023 welcomed some exciting stories, as were Teenage Mutant Ninja Turtles: Mutant Mayhem, the sci-fi action horror movie Meg 2: The ...',
'Domestic Box Office For September 2023 ; 7, Gran Turismo, - ; 8, Oppenheimer, - ; 9, PAW Patrol: The Mighty Movie, - ; 10, Teenage Mutant Ninja T
'Movie (+ IMAX). April 7 (Friday). Paint · Chupa (Netflix) How to Blow Up a Pipeline (Theaters) Joyland (Theaters) One True Loves (Theaters) Showi
Do I need to use a tool? No
Final Answer: It looks like there are several movies coming out in September 2023, including The Kill Room, The Creator, Expend4bles, Bottoms, PAW
> Finished chain.
'
It looks like there are several movies coming out in September 2023,
including The Kill Room, The Creator, Expend4bles, Bottoms,
 PAW Patrol: The Mighty Movie, Dumb Money, A Haunting in Venice,
 Teenage Mutant Ninja Turtles: Mutant Mayhem, Gran Turismo, Oppenheimer,
Paint, Chupa, How to Blow Up a Pipeline, Joyland, One True Loves, Showing Up,
Booked: First Day In, Botched, The Challenge: USA, The Dead Files,
The Wonderful Story of Henry Sugar, The Amazing Maurice, M3gan,
The Pale Blue Eye, Dog Gone, My Big Fat Greek Wedding 3, and The Nun II.
```

AI写代码

2.4、文档搜索ReAct

文档搜索ReAct支持从文档中先搜索，将搜索的答案和问题进一步进行整合回答。

2.4.1、使用LangChain LCEL实现文档搜索Agent

```
from langchain.agents import AgentType, Tool, initialize_agent
from langchain.agents.react.base import DocstoreExplorer
from langchain.docstore import Wikipedia
from langchain.llms import OpenAI
#使用维基百科作为文档搜索的输入源
docstore = DocstoreExplorer(Wikipedia())
tools = [
    Tool(
        name="Search",
        func=docstore.search,
        description="useful for when you need to ask with search",
    ),
    Tool(
        name="Lookup",
        func=docstore.lookup,
        description="useful for when you need to ask with lookup",
    ),
]
llm = OpenAI(temperature=0)
#初始化智能体
react = initialize_agent(tools, llm, agent=AgentType.REACT_DOCSTORE, verbose=True)
#用户问题
question = "Author David Chanoff has collaborated with a U.S. Navy admiral who served as the ambassador to the United Kingdom under which Presiden
#智能体执行
react.run(question)
```

AI写代码

2.4.2、文档搜索Agent返回答案全流程

```
> Entering new AgentExecutor chain...Thought: I need to search David Chanoff and the U.S. Navy admiral, find the ambassador to the United Kingdom,
Action: Search[David Chanoff]
Observation: David Chanoff is a noted author of non-fiction work. His work has typically involved collaborations with the principal protagonist of
Thought: David Chanoff has collaborated with a U.S. Navy admiral who served as the ambassador to the United Kingdom. I need to search U.S. Navy ad
Action: Search[U.S. Navy admiral]
```

```
Observation: Admiral of the Navy was the highest-possible rank in the United States Navy, prior to the creation of fleet admiral in 1944. The rank

Action: Search[U.S. Navy admiral ambassador to the United Kingdom]
Observation: Could not find [U.S. Navy admiral ambassador to the United Kingdom]. Similar: ['United States Secretary of the Navy', 'List of United
Thought: I can search William J. Crowe and find the ambassador to the United Kingdom.
Action: Search[William J. Crowe]
Observation: William James Crowe Jr. (January 2, 1925 – October 18, 2007) was a United States Navy admiral and diplomat who served as the 11th cha
Thought: William J. Crowe was the ambassador to the United Kingdom under President Bill Clinton, so the answer is Bill Clinton.
Action: Finish[Bill Clinton]
> Finished chain.
'
Bill Clinton
```

AI写代码

- 用户问题：
  作者大卫·查诺夫曾与一位美国海军上将合作，他曾在哪任总统的领导下担任驻英国大使?

```
智能体执行过程：
Step1：
我需要寻找大卫·查诺夫和美国海军上将，找到驻英国大使，然后找到他们所服务的总统。
Step2:
大卫·查诺夫 (David Chanoff) 是一位著名的非小说类作家。
 他的作品通常涉及与相关作品的主要主角的合作。
他的合作者包括： Augustus A. White、Joycelyn Elders、
Đoàn Văn Toại、William J. Crowe、Ariel Sharon、Kenneth Good
 和 Felix Zandman。 他还为《华盛顿邮报》、《新共和》和
《纽约时报杂志》撰写过有关文学史、教育和外国等广泛主题的文章。
他出版了十二本书以上。
Step3:
大卫·查诺夫曾与一位担任驻英国大使的美国海军上将合作。
 我需要寻找美国海军上将并找到驻英国大使。
Step4:
在 1944 年设立舰队上将之前，海军上将是美国海军的最高军衔。
该军衔被认为至少相当于五星级海军上将，
其中乔治·杜威海军上将是 唯一被任命为军衔的军官。
Step5:
该段没有提及驻英国大使。 我需要寻找美国海军上将驻英国大使。
Step6:
找不到[美国 海军上将驻英国大使]。
类似：[“美国海军部长”、“美国海军四星上将名单”、
“雷蒙德·A·斯普鲁恩斯”、“第一海军杰克”、
“英美关系”、“ 珍珠港袭击”、“美国革命之子成员名单”、
“威廉·J·克罗”、“肯尼思·布雷斯韦特”、“杰出服务奖章（美国陆军）”]
Step7:
我可以搜索威廉·J·克罗并找到驻英国大使。
Step8:
小威廉·詹姆斯·克罗（William James Crowe Jr.，1925年1月2日－2007年10月18日）
是一位美国海军上将和外交官，
曾担任罗纳德·里根总统和乔治·H·W·布什总统领导下的第十一任参谋长联席会议主席，
并担任美国驻华大使。 英国和比尔·克林顿总统领导下的情报监督委员会主席。
Step9:
威廉·J·克罗是比尔·克林顿总统领导下的驻英国大使，所以答案是比尔·克林顿。
```

AI写代码

- 返回答案： 比尔·克林顿

参考文档：

https://python.langchain.com/docs/get_started/introduction

2023年新生代大模型Agents技术,ReAct,Self-Ask,Plan-and-execute,以及AutoGPT, HuggingGPT等应用 - 知乎

https://python.langchain.com/docs/modules/agents/agent_types/self_ask_with_search

LangChain大模型应用开发指南-从最基础的链式结构到ReAct对话解构 - 知乎

小白入门大模型：LangChain - 知乎

https://zhuanlan.zhihu.com/p/62