AI大模型应用：NLP与大模型

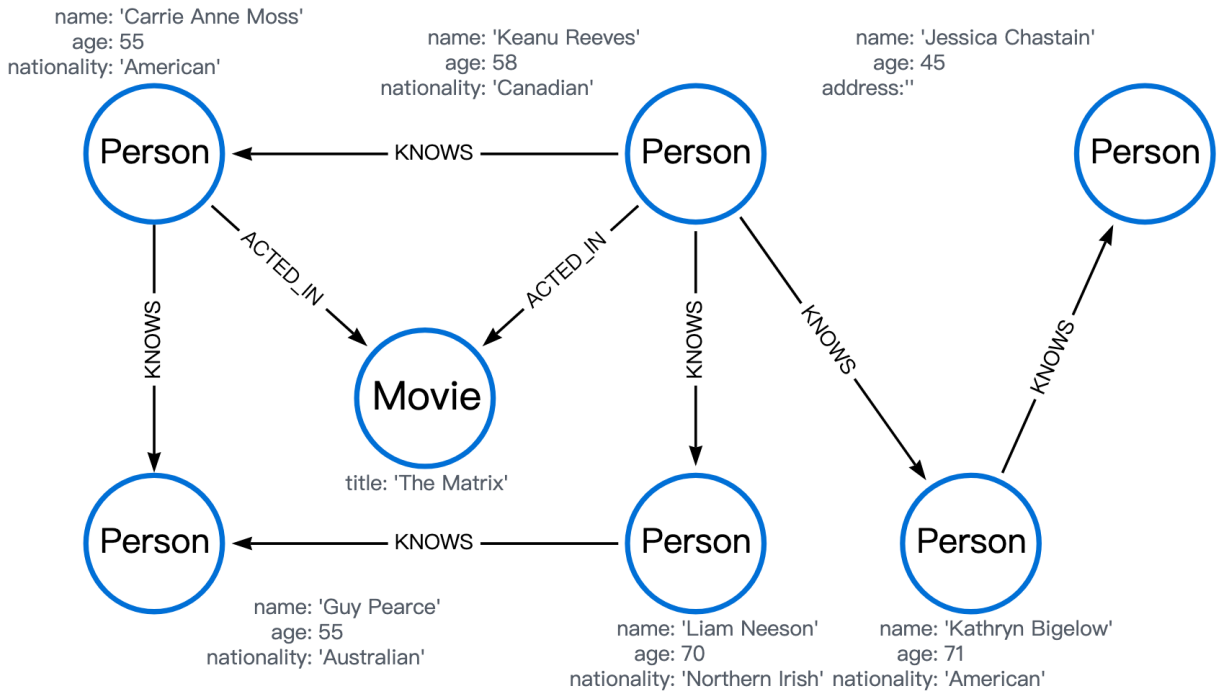
2025年

内部资料，请勿外传

图数据库：Neo4j

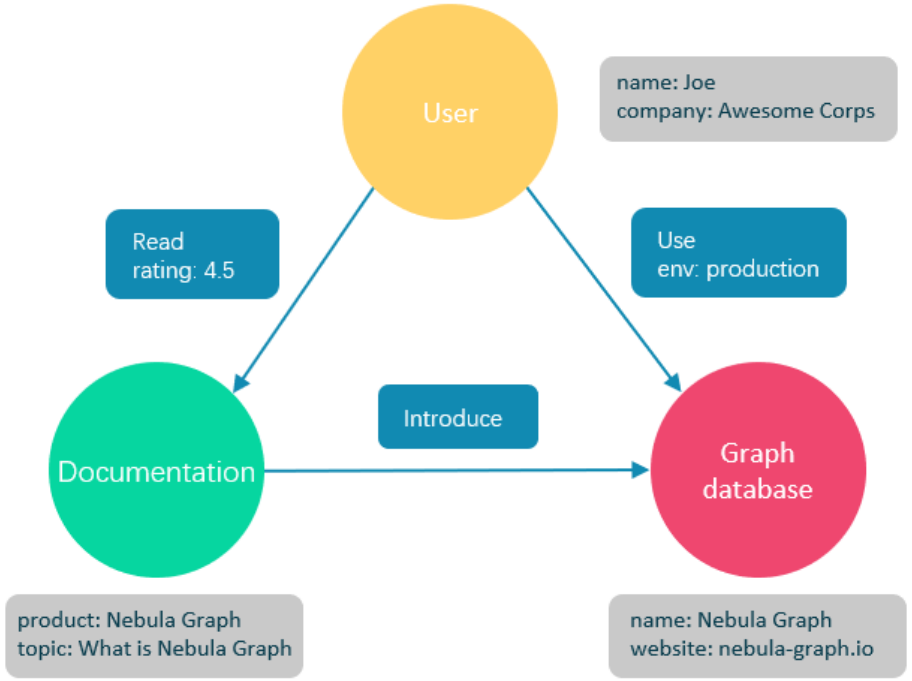
Neo4j是一种图数据库管理系统，它专注于处理图形数据模型。图数据库是一种用于存储和查询图形结构的数据库，其中数据以节点和边的形式表示，节点表示实体，边表示实体之间的关系。

- ✓ Neo4j使用一种名为Cypher的声明性查询语言，这种语专门设计用于图数据库。Cypher语法简洁，易于理解，许用户以自然的方式查询和操作图数据。
- ✓ Neo4j提供ACID（原子性、一致性、隔离性和持久性）务支持，确保数据的完整性和一致性。



图数据库： Nebula Graph

NebulaGraph是一个分布式、可扩展、高性能的图数据库，旨在有效地存储和检索庞大的图网络中的信息。图数据库，如NebulaGraph，专注于通过在标记的属性图中将数据表示为顶点（节点）和边缘（关系）来管理实体之间的关系。顶点和边缘都可以附有属性，并且顶点可以带有一个或多个标签。

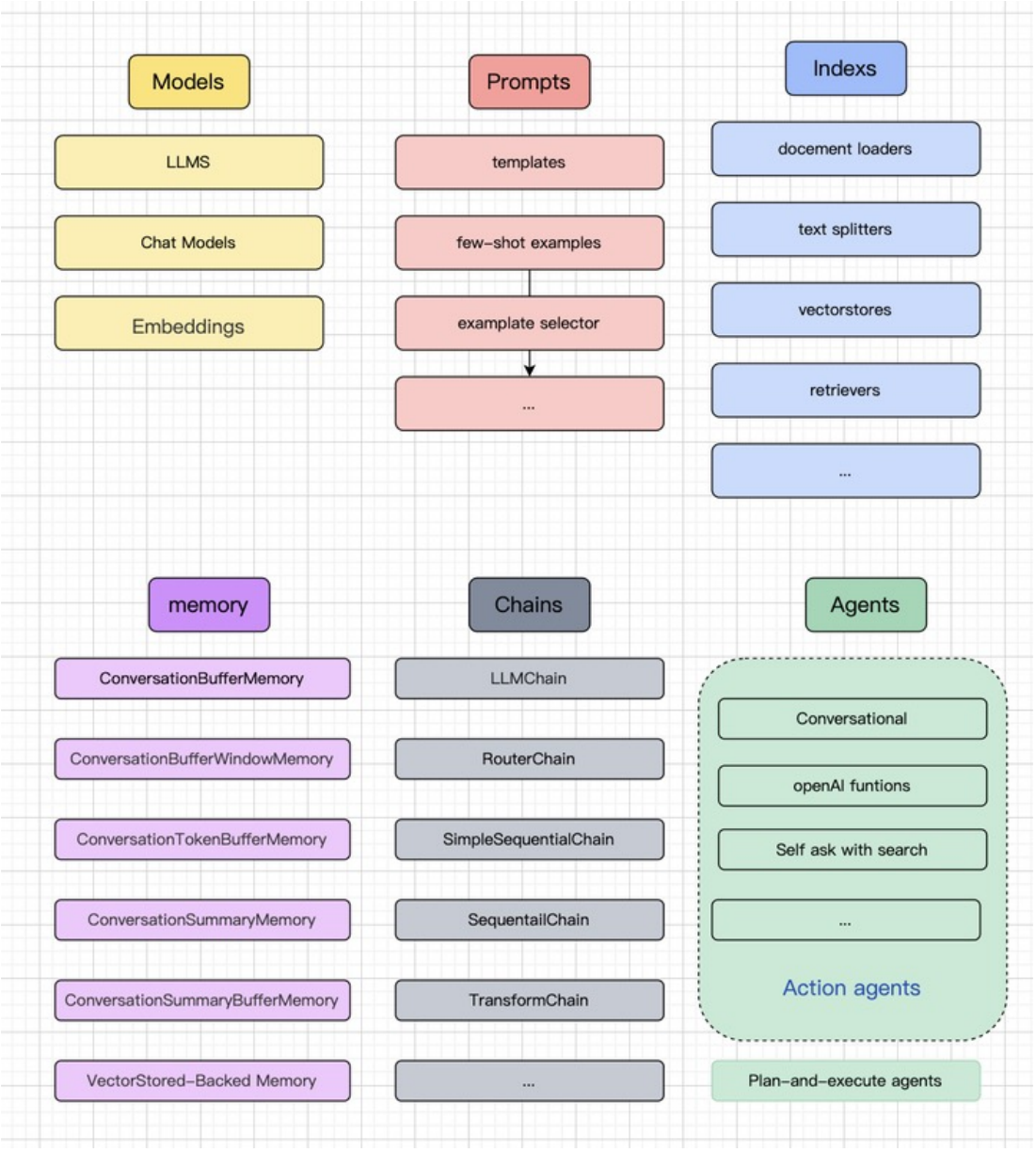


- ✓ NebulaGraph支持流行的编程语言，如Java、Python、C++和Go，旨在为开发人员提供友好的使用体验。其他客户端库正在开发中，以扩展语言支持。
- ✓ NebulaGraph查询语言（nGQL）是一种声明性的、兼容OpenCypher的文本查询语言，易于理解和使用，用于查询图数据。

大模型开发框架：LangChain

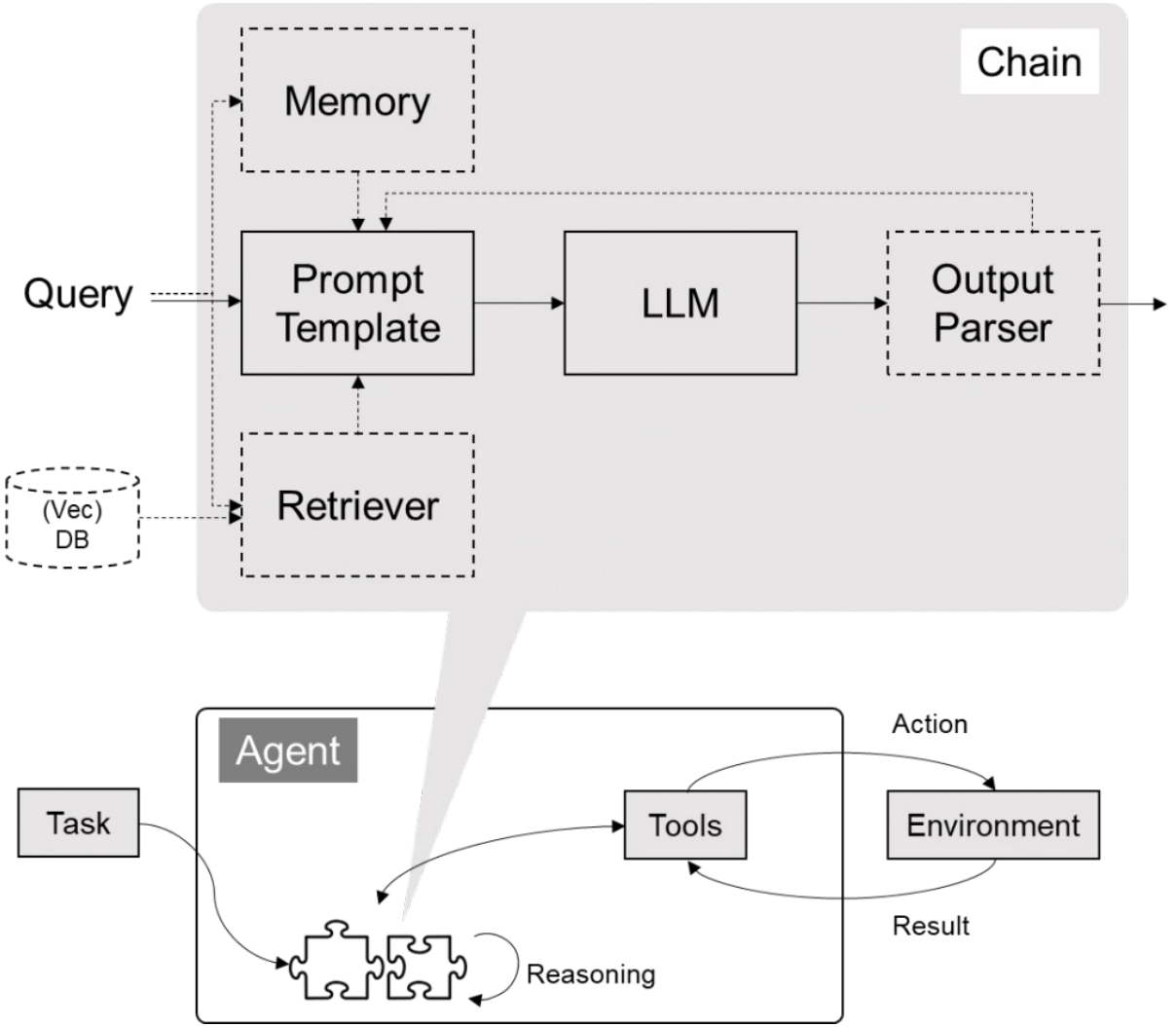
LangChain 是一个用于开发由语言模型驱动的应用程序的框架。我们相信，最强大和不同的应用程序不仅将通过 API 调用语言模型：

- ✓ 组件：LangChain 为处理语言模型所需的组件提供模块化的抽象。LangChain 还为所有这些抽象提供了实现的集合。这些组件旨在易于使用，无论您是否使用 LangChain 框架的其余部分。
- ✓ 用例特定链：链可以被看作是以特定方式组装这些组件，以便最好地完成特定用例。这旨在成为一个更高级别的接口，使人们可以轻松地开始特定的用例。这些链也旨在可定制化。



大模型开发框架： LangChain

- ✓ **集成和扩展**： LangChain 提供了一个框架，使开发者可以轻松地集成和扩展大型语言模型。通过 LangChain，开发者可以将 LLM 的强大能力与其他工具和数据源结合，创建更复杂和强大的 NLP 应用。
- ✓ **组件化设计**： LangChain 提供了各种组件，使开发者能够模块化地构建应用。这些组件包括模型管理、数据处理、任务调度、交互界面等。
- ✓ **增强的交互性**： 通过 LangChain，开发者可以实现更复杂的用户交互。例如，可以设计多轮对话系统、复杂的问答系统，或者根据用户输入动态调用外部 API 进行信息查询和处理。
- ✓ **提升性能和效率**： LangChain 通过优化模型调用、缓存机制、并行处理等技术手段，提升了 LLM 的性能和效率。



大模型开发框架： LangChain

```python
from langchain.prompts import PromptTemplate

template = PromptTemplate.from_file("example_prompt_template.txt")
print("===Template===")
print(template)
print("===Prompt===")
print(template.format(topic='黑色幽默'))
```

```
------------------------

===Template===
input_variables=['topic'] template='举一个关于{topic}的例子'
===Prompt===
举一个关于黑色幽默的例子
```

```python
import os
os.environ["OPENAI_API_KEY"] = 'your apikey'
from langchain.prompts import PromptTemplate, FewShotPromptTemplate
from langchain.prompts.example_selector import LengthBasedExampleSelect


# These are a lot of examples of a pretend task of creating antonyms.
examples = [
    {"word": "happy", "antonym": "sad"},
    {"word": "tall", "antonym": "short"},
    {"word": "energetic", "antonym": "lethargic"},
    {"word": "sunny", "antonym": "gloomy"},
    {"word": "windy", "antonym": "calm"},
]
# 例子格式化模版
example_formatter_template = """
Word: {word}
Antonym: {antonym}\n
"""
example_prompt = PromptTemplate(
    input_variables=["word", "antonym"],
    template=example_formatter_template,
)

# 使用 LengthBasedExampleSelector来选择例子
example_selector = LengthBasedExampleSelector(
    examples=examples,
    example_prompt=example_prompt,
    # 最大长度
    max_length=25,
)

# 使用'example_selector'创建小样本提示词模版
dynamic_prompt = FewShotPromptTemplate(
    example_selector=example_selector,
    example_prompt=example_prompt,
    prefix="Give the antonym of every input",
    suffix="Word: {input}\nAntonym:",
    input_variables=["input"],
    example_separator="\n\n",
)

longString = "big and huge and massive and large and gigantic and tall

print(dynamic_prompt.format(input=longString))
```

提示词工程

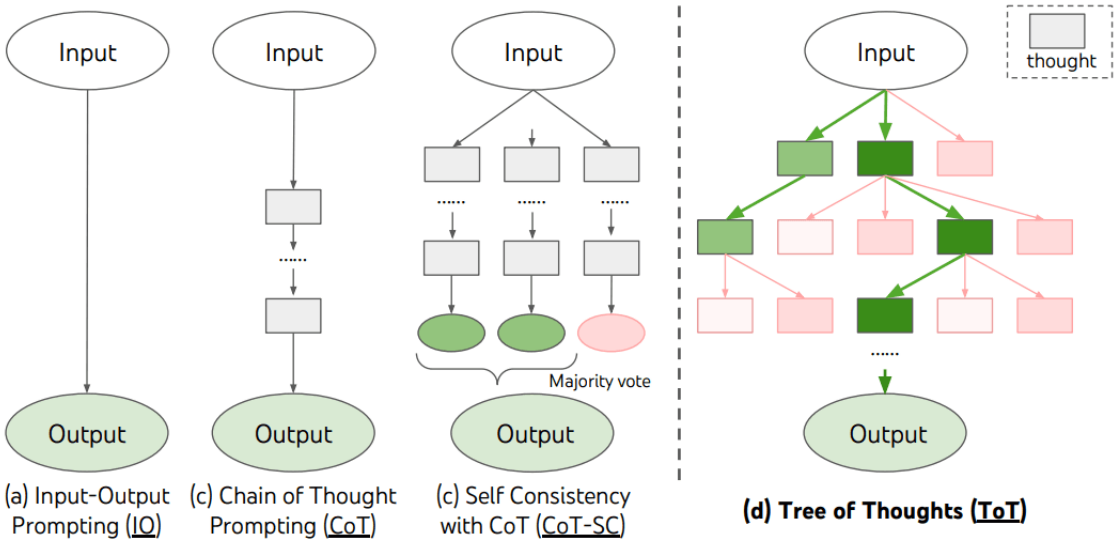
- 零样本提示
- 少样本提示
- 链式思考（CoT）提示
- 自我一致性
- 生成知识提示
- Prompt Chaining
- 思维树（ToT）
- 检索增强生成 (RAG)
- 自动推理并使用工具（ART）
- 自动提示工程师
- Active-Prompt
- 方向性刺激提示
- Program-Aided Language Models
- ReAct框架
- Reflexion
- 多模态思维链提示方法
- 基于图的提示

提示：

```
将文本分类为中性、负面或正面。
文本：我认为这次假期还可以。
情感：
```

输出：

```
中性
```



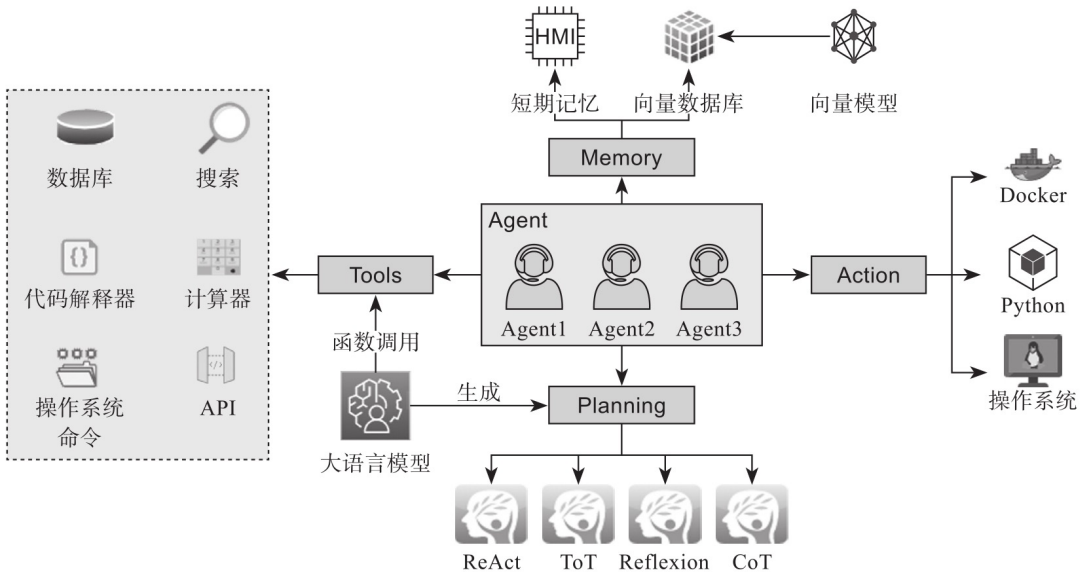
(a) Input-Output Prompting (IO) (c) Chain of Thought Prompting (CoT) (c) Self Consistency with CoT (CoT-SC) **(d) Tree of Thoughts (ToT)**

https://www.promptingguide.ai/zh/techniques

Agent智能体

AI Agent（或简称为Agent）是建立在大语言模型之上的智能应用，是将人工智能与特定场景深度结合的重要方式。Agent模仿人类“思考-行动-观察”的规划模式，具备自主思考和自主决策的能力，能够适应环境的变化，自主学习和改进，完成用户设定的目标。

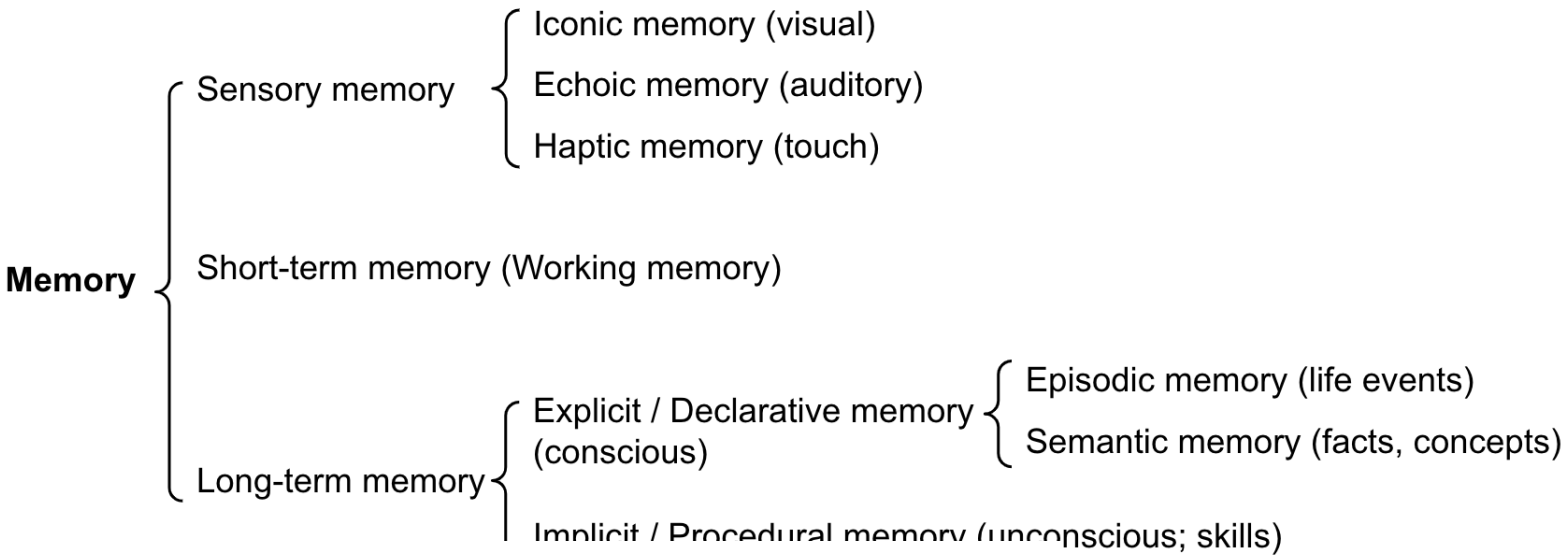
与大语言模型的对话应用不同，Agent的突出特点是主动性，在行为上表现为多步操作、多角色会话、多轮迭代、反复修正答案以及调用外部资源的能力。

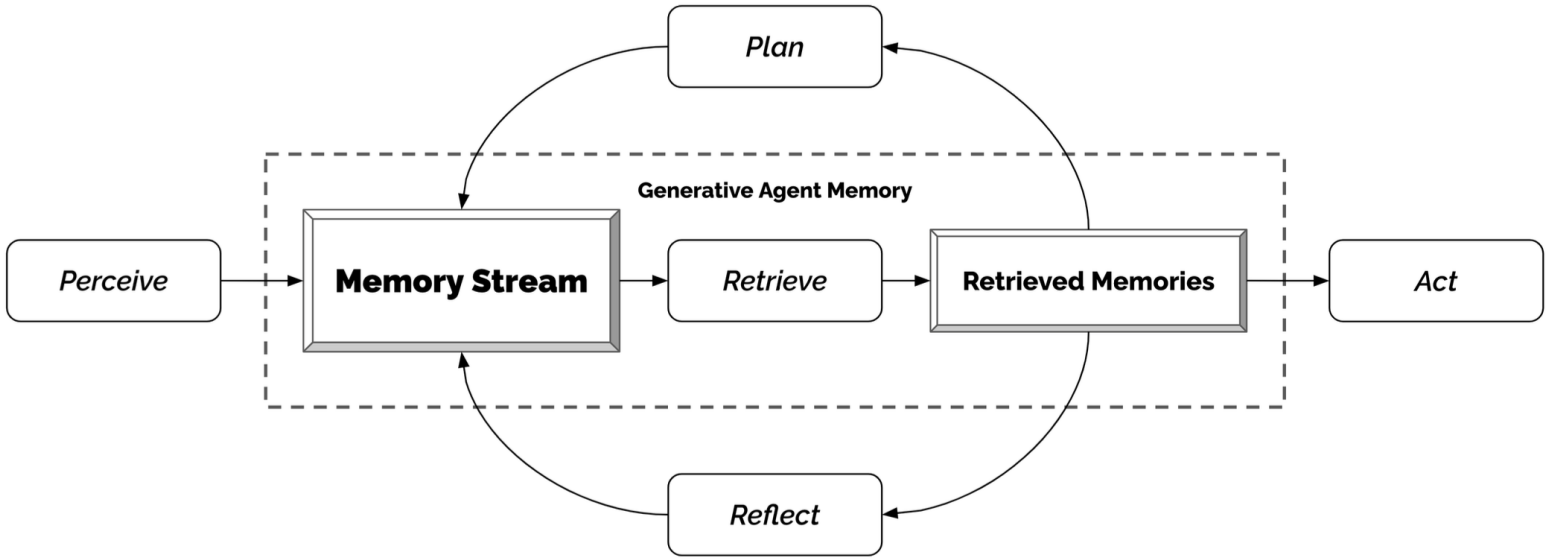


Agent智能体

What's an AI agent?

agent : take actions to achieve a goal
= perception (observation)
+ action (decision)
+ feedback (reward)
+ learning (policy)
(+ model / simulator w/ planning)





https://lilianweng.github.io/posts/2023-06-23-agent/

Agent智能体

Agent这个框架包含多个部分，分别是**规划（Planning）、记忆（Memory）、工具（Tools）、动作（Action）等**，分别介绍一下：

✓ **规划（Planning）：**主要包括**子目标分解、反思与改进。**其中子目标分解主要指的是Agent可以将大型任务分解为较小，可管理的子目标，从而有效地处理复杂的任务；而反思和改进指的是Agent可以对过去的行动进行自我批评和自我反思，从错误中学习并改进未来的步骤，从而提高最终结果的质量。

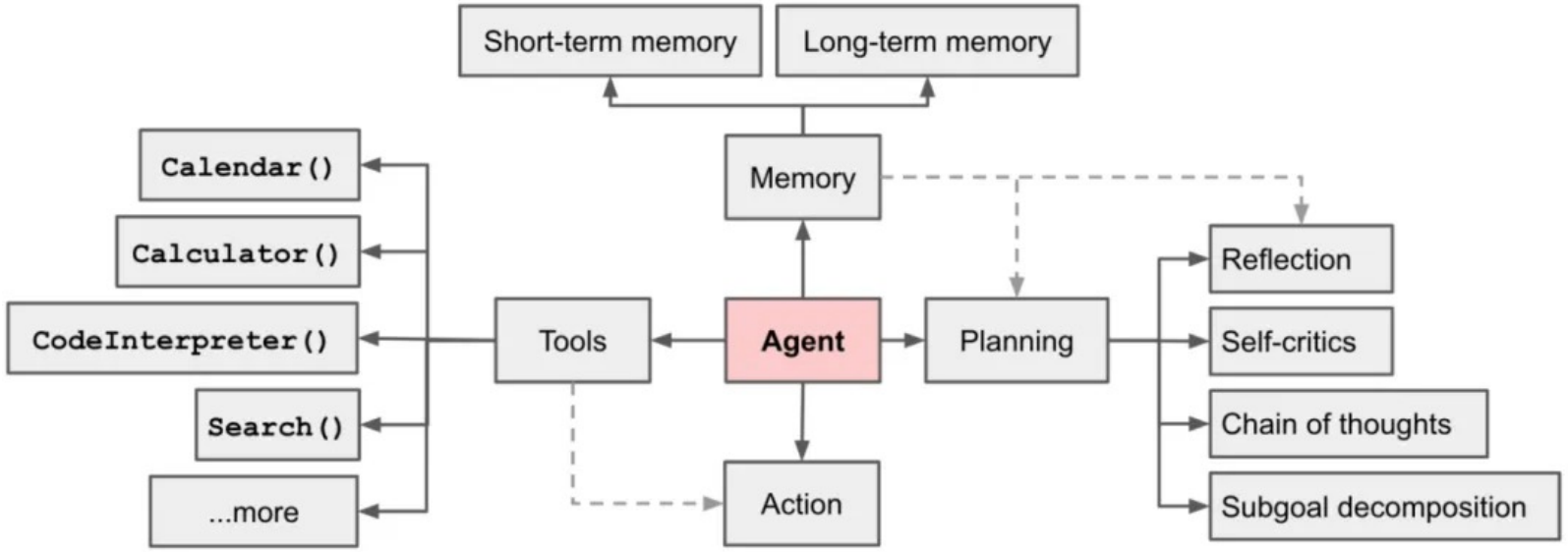
✓ **记忆（Memory）：**分为**短期记忆**和**长期记忆。**其中短期记忆是指的将所有的上下文学习（比如Prompt Engineering、In-Context Learning）都看成是利用模型的短期记忆来学习；而长期记忆为Agent提供了长期存储和召回信息的能力，它们通常通过利用外部的向量存储和快速检索来存储和召回信息。

✓ **工具（Tools）：**Agent通过学会调用外部API来获取模型权重（通常在预训练后很难修改）中缺少的额外信息，包括当前信息，代码执行能力，访问专有信息源等。

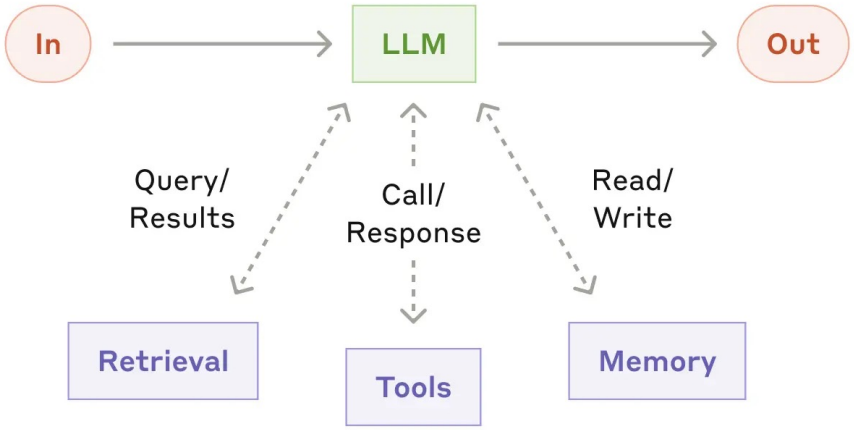
✓ **动作（Action）：**根据上述的规划、记忆、工具，大模型才能决策出最终需要执行的动作是什么。

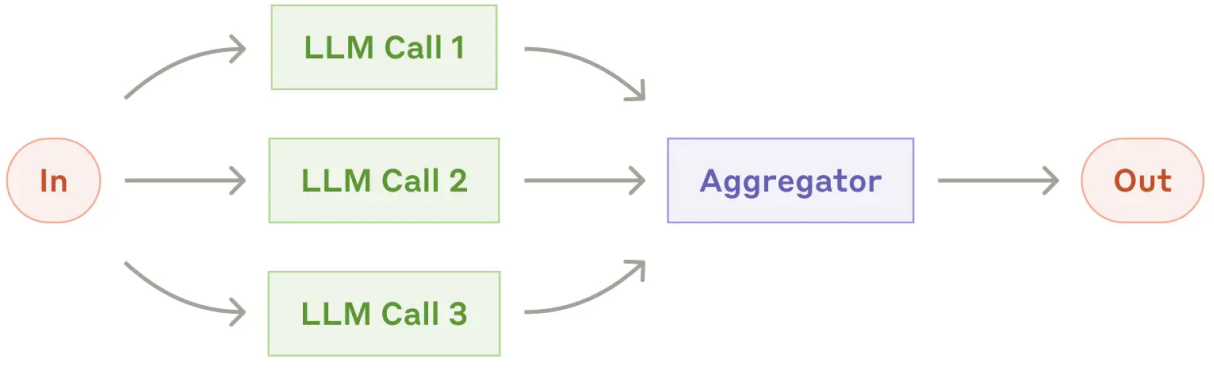
Agent组成

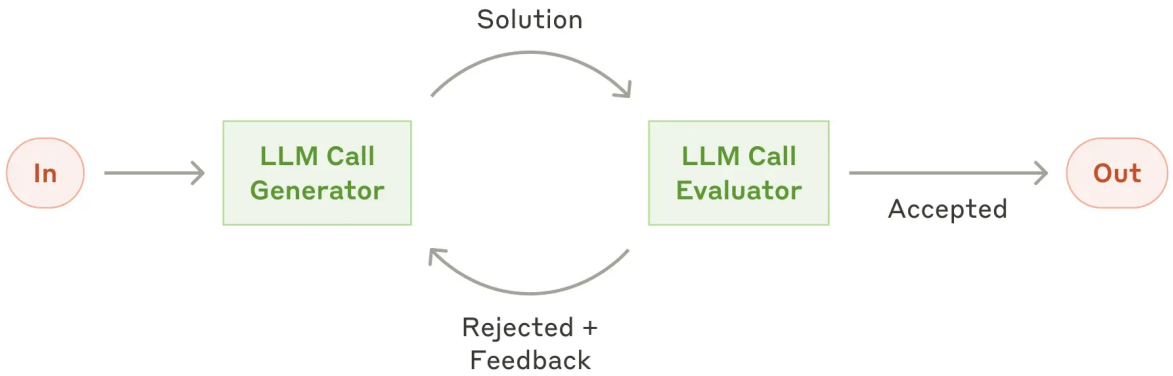
Planning组件思考问题的过程中，有提示词工程(Prompt Engineering)的理论和实践支持，也有Action组件和Tools组件执行外部命令的反馈，还有大语言模型理解Planning组件提出的问题而推理给出的回应。这几个因素综合起来，让Agent模仿出人类“思考—行动—观察—然后再思考”的多次迭代的流程，使得Agent具备了主动决策和处理事务的能力，形成一种智体。

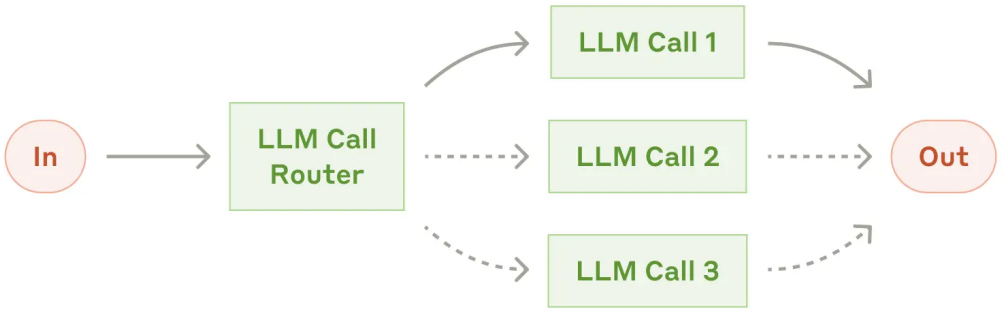


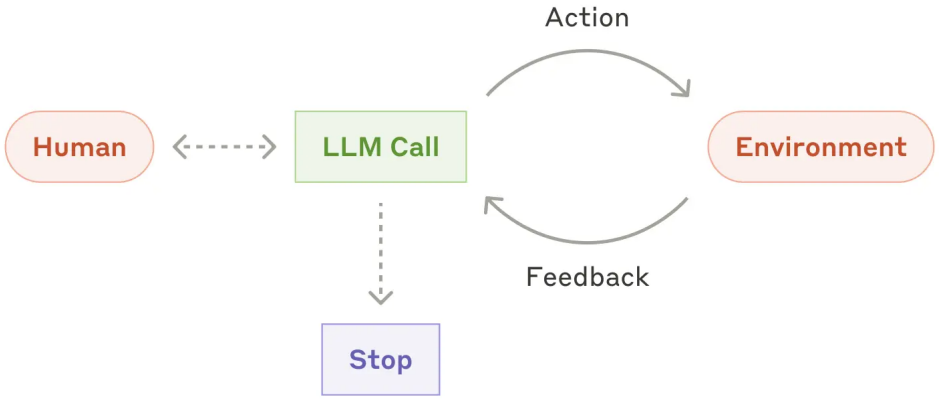
Agent落地方式

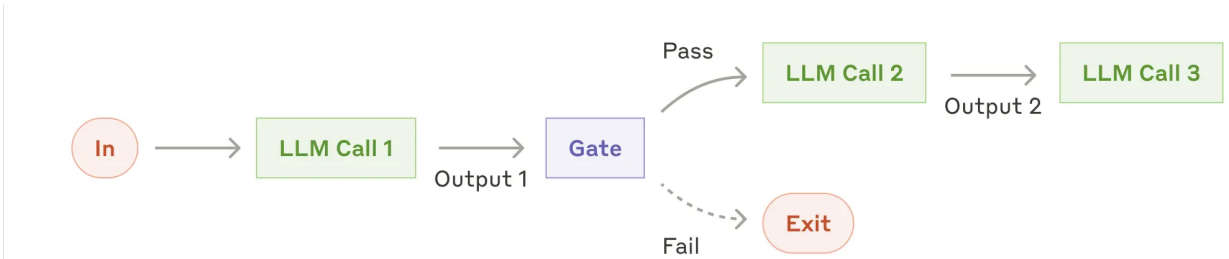






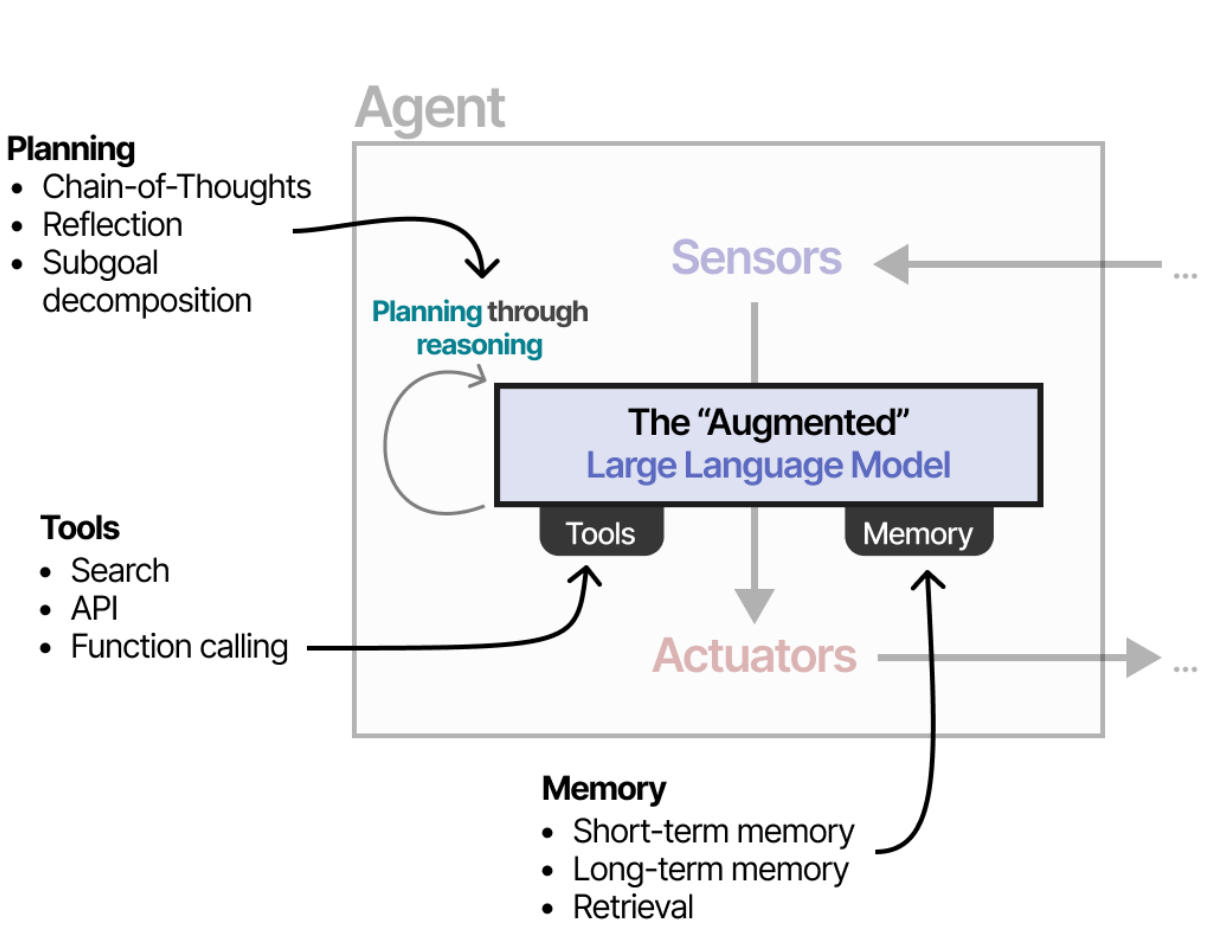


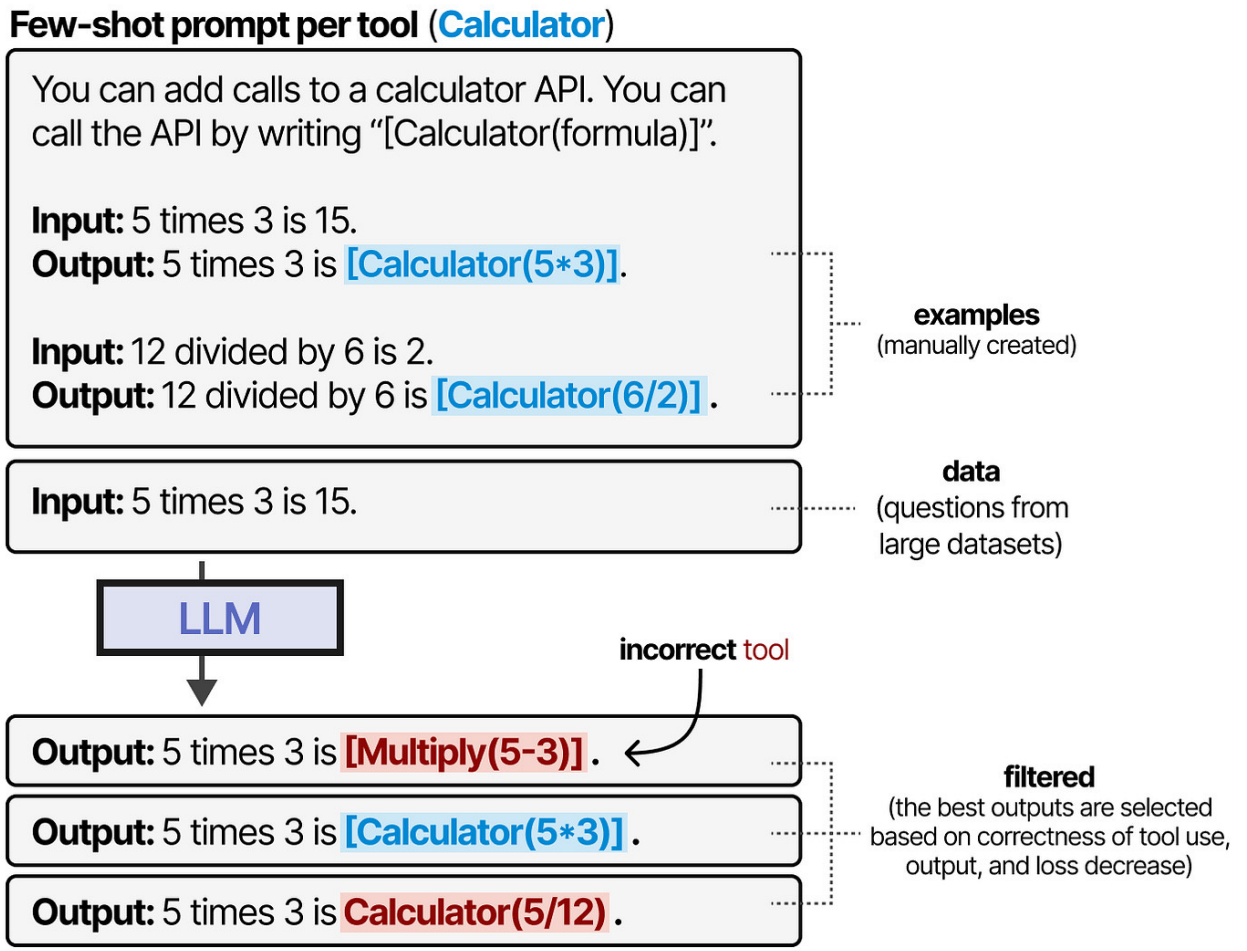




https://www.anthropic.com/engineering/building-effective-agents

Agent与Tools

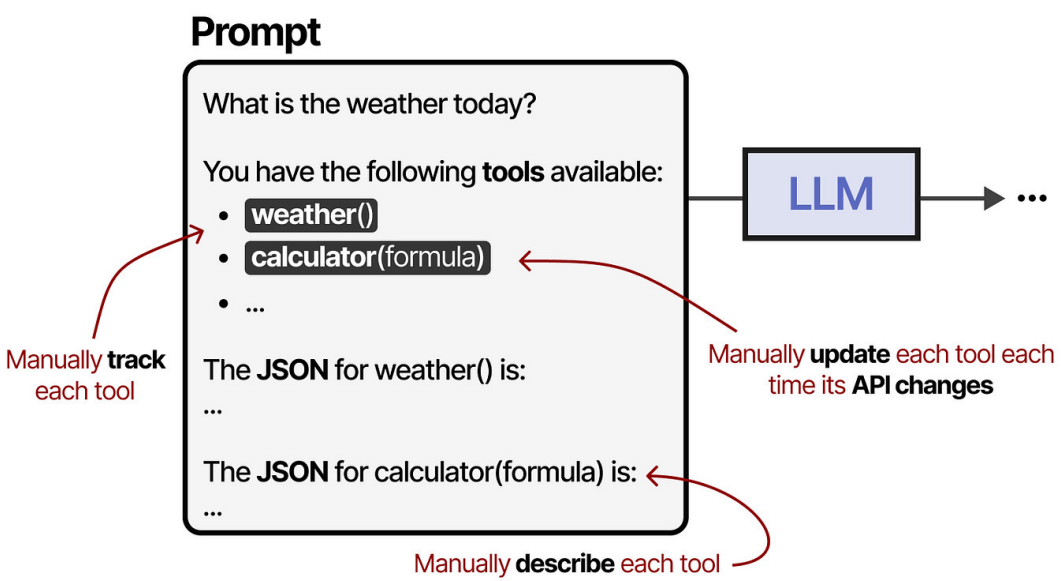


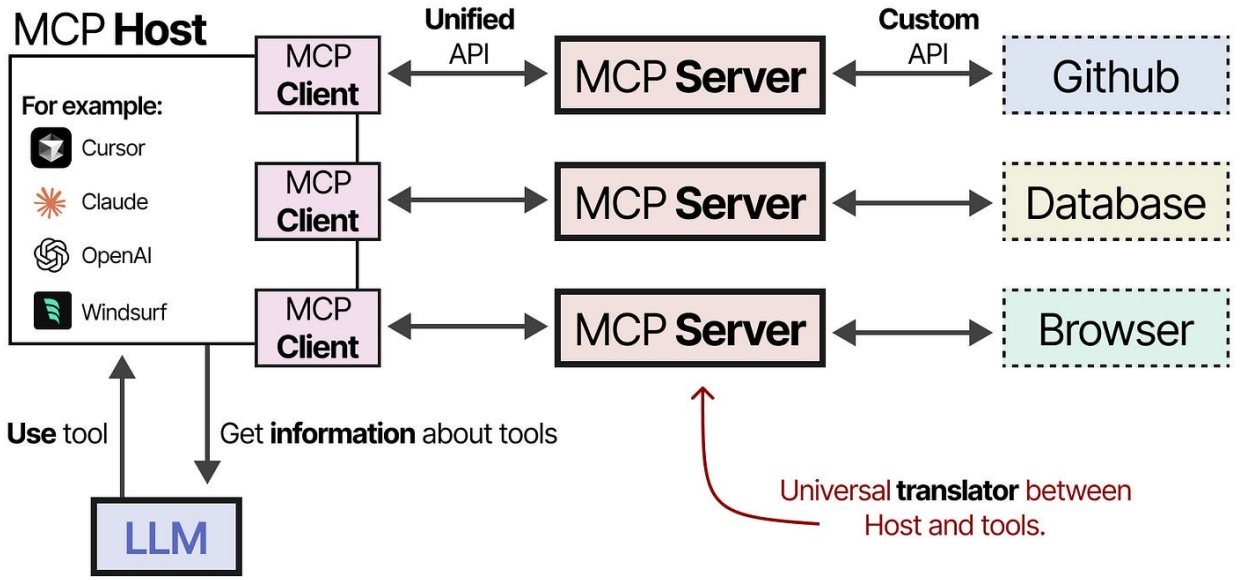


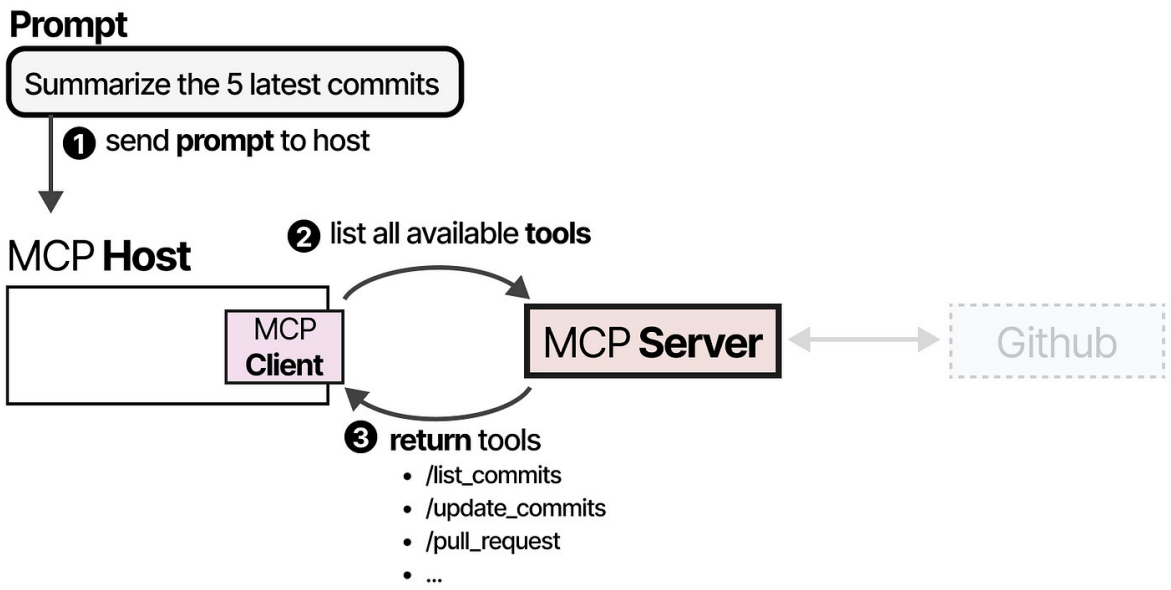
https://newsletter.maartengrootendorst.com/p/a-visual-guide-to-llm-agents

Agent与MCP

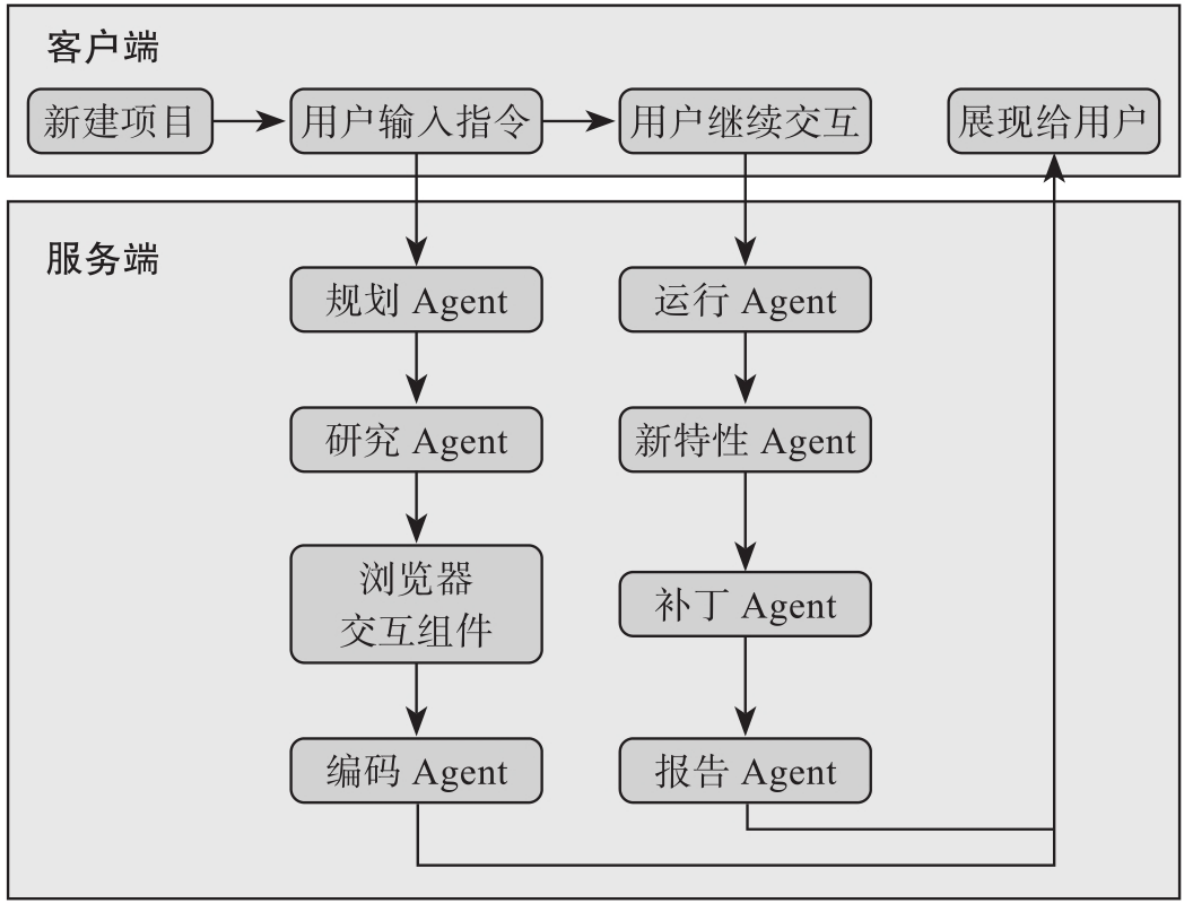
https://newsletter.maartengrootendorst.com/p/a-visual-guide-to-llm-agents







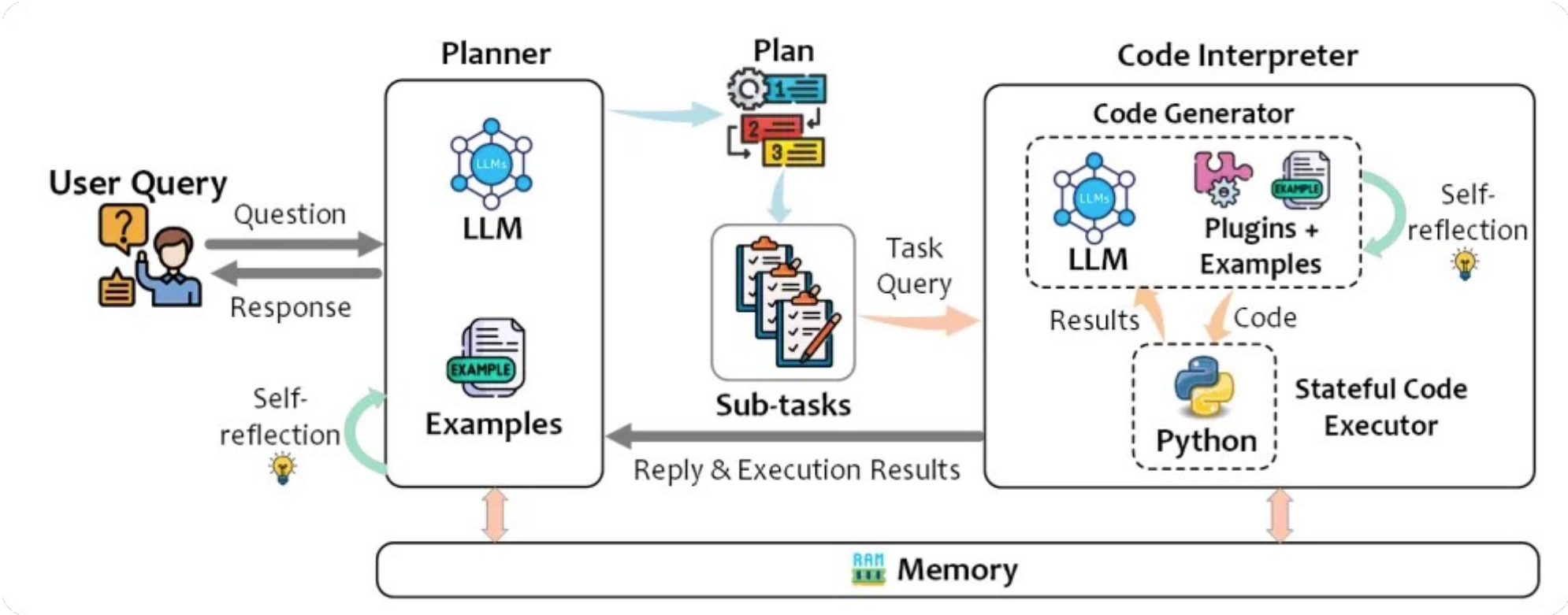
Agent框架案例： Devika



Devika是一个AI智体软件，用于软件辅助开发场景，可以理解为人类输入指令，Agent将指令分解为操作步骤，并自主制订计划、编写代码以实现给定的目标。Devika利用大语言模型、规划和推理算法以及Web浏览能力来智能化地开发软件。它改变传统构建软件的方式，可以在最少的人工指导下承担复杂的开发任务，其能力包括创建新功能、修复错误，甚至从头开始开发整个项目。

https://github.com/stitionai/devika

Agent框架案例： TaskWeaver



TaskWeaver 是一个以代码为中心的智能代理框架，用于无缝规划和执行数据分析任务。这个创新框架通过代码片段解释用户请求，并高效协调各种插件（以函数形式存在）来执行数据分析或工作流自动化任务。

✓ 有状态的对话 - TaskWeaver 设计为支持有状态的对话，这意味着你可以在多个聊天回合中与内存中的数据进行交互。

✓ 代码验证 - TaskWeaver 设计为在执行前验证生成的代码。它可以检测生成代码中的潜在问题并自动修复它们。

https://github.com/microsoft/TaskWeaver