Agent概述

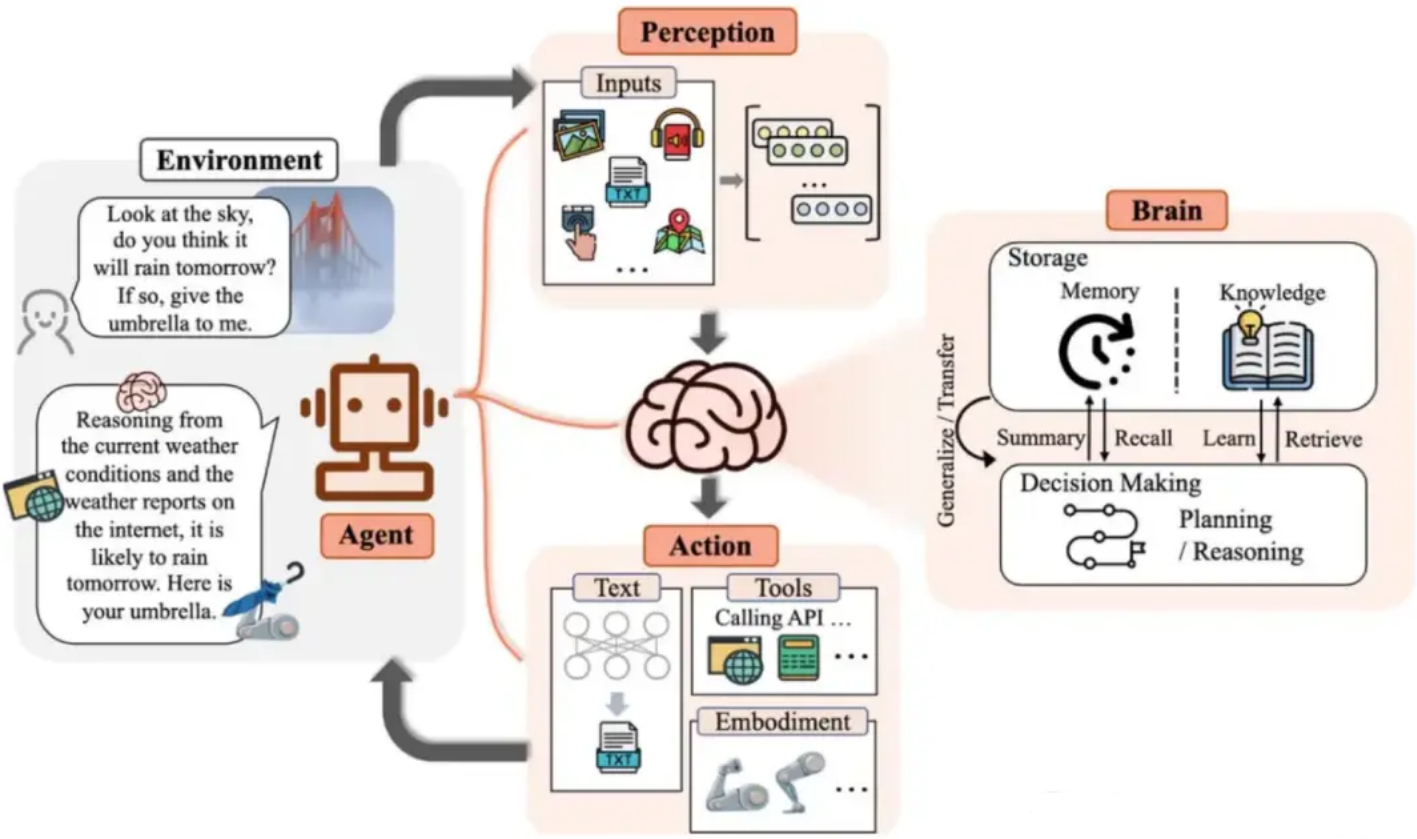
4.1 Agent 概述

> 大模型存在的固有问题：**无法主动更新自己的知识，导致出现事实幻觉**。RAG可以一定程度上缓解这个问题，让大模型先在本地知识库中进行搜索，检查一下提示中的信息的真实性，如果真实，再进行输出；如果不真实，则进行修正。但如果本地知识库找不到相应的信息，就应该调用工具进行外部搜索，这就需要使用 Agent（Agent 能调用的工具不止外部搜索，还包括数学工具、编程工具等等）。

Agent（智能体）是一种能够自主决策、采取行动以达到某种目标的实体。在人工智能领域，Agent 是基于大模型技术构建的智能实体，能够感知和理解环境，并采取行动以完成特定任务。

从结构上来说，一个Agent包括三个部分，如下图所示：

- **Perception（输入）：** Agent通过文字输入、传感器、摄像头、麦克风等等，建立起对外部世界或环境的感知。
- **Brain（大脑）：** 大脑是Agent最重要的部分，包括信息存储、记忆、知识库、规划决策系统。
- **Action（行动）：** 基于Brain给出的决策进行下一步行动，对于Agent来说，行动主要包括对外部工具的API 调用，或者对物理控制组件的信号输出。



从功能的角度来看，Agent 就像一个多功能的接口，它能够接触并使用一套工具。根据用户的输入，Agent会规划出一条解决用户问题的路线，决定其中需要调用哪些工具，并调用这些工具。Agent = 大语言模型+规划+记忆+工具使用，具备以下关键能力：

- **规划（Planning）**：最核心最关键的部分，负责拆解复杂任务为可执行的子任务，并规划执行任务的流程。同时Agent还会对任务执行的过程进行思考和反思，决定是否继续执行任务，并改进决策策略。
  - **任务分解**：将复杂任务分解为可执行的子任务，让大模型逐步解决，例如将订外卖分解为选择餐厅+选择菜品两步。关键技术例如CoT、LLM+P等。
  - **反思**：Agent 通过完善过去的行动决策和纠正以前的错误来不断改进。关键技术例如React、Reflexion等。
- **记忆（Memory）**：包括短期记忆和长期记忆，用于存储会话上下文和业务数据等信息，来优化未来行为。
  - **短时记忆**：即上下文学习，由于受到Transformer上下文窗口长度的限制，它是短暂的和有限的。
  - **长期记忆**：则可对应为外部的向量数据存储，Agent 可在查询时引用，并可通过快速检索进行访问。
- **工具使用（Tools）**：通过调用外部工具（如API、插件）扩展Agent的能力，如文档解析、代码编译等。

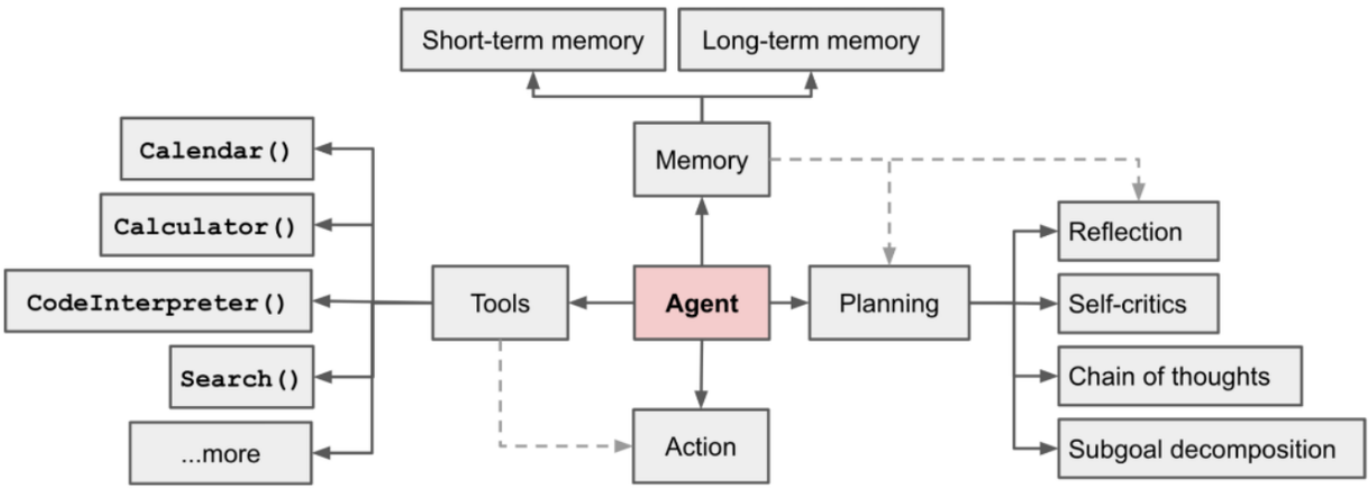


Fig. 1. Overview of a LLM-powered autonomous agent system.

Agent开发框架概览

- **低代码框架：**无需代码就能在线完成Agent开发，
  - 扣子coze（字节）：https://www.coze.cn
  - 通义千问（阿里）：https://tongyi.aliyun.com/qianwen

    - 文心智能体（百度）：https://agents.baidu.com/center
    - 元器智能体（腾讯）：https://yuanqi.tencent.com/agent-shop
    - Dify: https://dify.ai/zh
    - Fastgpt: https://fastgpt.cn
- **基础框架**：利用大模型原生能力进行Agent开发
    - function calling：https://platform.openai.com/docs/guides/function-calling
- **代码框架**：
    - Langchain：https://www.langchain.com/
    - LangGraph：https://langchain.ac.cn/langgraph
    - LlamaIndex：https://docs.llamaindex.ai/en/stable/
- **多智能体框架**：
    - CrewAI：https://www.crewai.com/
    - Swarm：https://github.com/openai/swarm
    - MegaGPT：https://github.com/openai/swarm

🏆 **开放式问题：谈谈你对Agent的理解?**

> 这个问题准确来说，应该是谈谈你对基于大模型的Agent的理解（之前在介绍强化学习时，也有Agent的概念，本章中讲的Agent特指基于LLM的Agent）。

在Agent诞生之前，有两种方式能使机器智能化：

- **基于规则的方法**：将人类指令转化成机器能理解的规则符号，这需要有丰富经验的人类专家，并且容错很低。
- **基于强化学习的方法**：构建策略模型和奖励模型，需要大量的数据进行训练。

随着大模型的诞生，人类利用其在逻辑推理、工具应用、策略规划等方面的能力，构建以大模型为核心的Agent系统，极大的提升了机器的智能化程度。当然，为了进一步提升Agent的性能，还提出了CoT等规划方法、引入记忆和工具模块，使得Agent越来越逼近人类的思考方式。

从人机合作的角度出发，Agent 改变了人机合作的方式。截至现在，主要有三种模式：

- **人类主导**：代表是**SaaS+AI模式**，人类完成大多数工作，而AI只负责完成特定任务。例如AI只负责实现人脸识别、OCR等能力，嵌入到人类操作的SaaS软件中，其他功能AI不参与。

- **AI作为人类助手**：代表是**Copilot模式**，AI可以随时辅助人类完成各种任务，不再局限于特定的功能。
- **AI主导**：代表**Agent模式**，人类只负责提出需求，在AI负责完成的过程中，可能需要人类进行进一步的描述需求、点评AI生成内容质量、矫正AI理解等。而Agent正式通往AGI（Artificial General Intelligence）的必经之路。

对比维度	传统对话式 AI	基于 LLM 的 AI Ager
交互方式	单轮或有限多轮对话	支持多轮交互，任务持续
能力边界	仅限于预设回复、有限理解	可动态调用工具，灵活执行
知识更新	静态（训练时固定）	可实时联网，获取最新
执行复杂任务能力	很弱	强，支持规划、执行、反馈、修

4.2 Agent 分类

按照工作模型可以分为单Agent、多Agent和混合Agent三种。

👍
- **单Agent**：
  - **特点**：由一个独立的智能体构成，所有的决策和执行都集中在一个智能体上，没有与其他智能体的协调和通信需求，适用于单一任务或相对简单的任务。
  - **优点**：不需要处理多个智能体之间的协调问题，也不需要额外的资源来管理多个智能体。
  - **缺点**：难以处理复杂、多变的环境，并且如果Agent出现故障，整个系统都将瘫痪。
  - **应用场景**：比如专门用于进行市场分析调研的Agent。


- **多Agent**：
  - **定义**：多个Agent协同工作，相互交流信息，共同完成更复杂的任务或目标。多个智能体在分布式环境中独立运行，每个智能体可以自主决策，需要处理智能体之间的通信、协调和竞争等问题。
  - **优点**：能够处理复杂、动态和多变的环境，可以完成单个智能体难以完成的任务。多个智能体之间可以相互协作，即使部分智能体出现故障系统仍然可以正常工作，鲁棒性强。能根据环境和任务需求动态调整，具有可拓展性。
  - **缺点**：需要大量的通信和协调来确保智能体之间的同步和协作。
  - **应用场景**：比如一家公司就可以视为一个多Agent系统，由Agent来扮演产品经理、UI设计师、研发工程师、测试人员、项目经理等角色。

- **例子**：斯坦福小镇

🥖
- **混合Agent：**
  - **定义**：Agent系统和人类共同参与决策过程，交互合作完成任务，强调的是人机协作的重要性和互补性。这种系统通常包含一个或多个智能体，以及与人类用户的交互接口。
  - **优点**：通过人类的参与，混合Agent系统可以更好地处理复杂和多变的任务，提高任务完成的质量和效率，灵活地调整人类和智能体的角色和任务分配，提供更个性化和人性化的服务。
  - **缺点**：开发难度和复杂度较高。
  - **应用场景：**医生和Agent可以共同进行病情诊断，Agent负责快速分析病人的医疗记录、影像资料等，提供初步的诊断建议；而医生则可以基于Agent的分析结果和自己的专业知识和经验，做出最终的诊断决定。