agent

= □□□+ memory + planning + tools □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

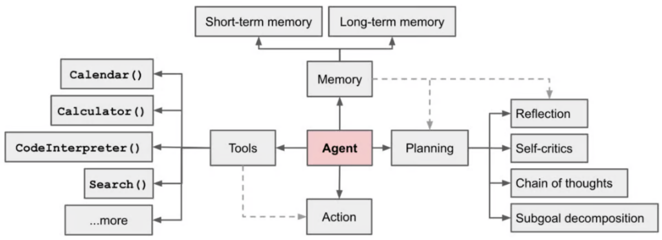
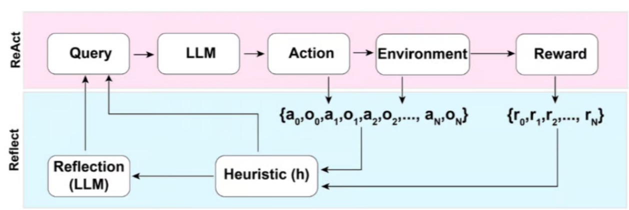


Fig. 1. Overview of a LLM-powered autonomous agent system.

Plan

□□□□
CoT chain of thoughts
ToT Tree of thoughts
LLM + P PDDL□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
ai□□□□□□
□□□
ReAct□
□□□□□□□□□□□□□□□□□□□
Reflexion
□□
□□□□□□□□□□□□REAct□□□□□□□□□□□agents□□□□□□□□□□□□□□□□□
□□□□□□□□□□



Reflexion框架的图解

□□□□□□□agent□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

- □□□□□□□□□□□□□□□□□□□□□□□□□□□
- □□□□□□□□□□□□□□□□□□□□□□□□□Agent□□□□□

□□□□

- □□□□□□□□□□□□
- □□□□□□□□□□□□□□□□□□□□□□□□agent□□□□□□□□□□□□□□□□□
□□

Chain of Hindsight
□□□□□□□□□□□□□□□ □□□□□□□□□□□□□□□□□□□□□□ □□□□□ □ □□□
□□ □□□□□□□

□□□□□□prompt□□□□□□□y□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□□□□□□□□□□□

□□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□□□□□□□0□5%□□□ttoken□□□□□□□□□□□□□□□□□□□□□□□□□□

CoH[illegible]Algorithm Distillation [illegible]

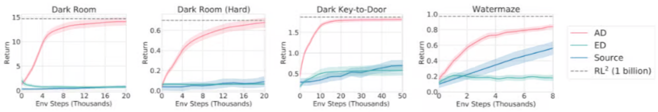
[illegible]Agent[illegible]

1. [illegible]
    - [illegible]
    - [illegible]AD[illegible]2-4[illegible]
    - [illegible]

1. [illegible]
    1. [illegible]RL[illegible]AD[illegible]
    2.

- 这样，AD通过 **行为克隆** 的方式，将强化学习过程中的学习历史和环境交互抽象成策略，从而获得一个可以在新的任务中快速适应的通用策略。

论文使用了三个基线进行对比：ED（专家蒸馏，使用专家轨迹而非学习历史的行为克隆）、源策略（用于生成UCB蒸馏的轨迹）、RL²（2017年提出的一种在线强化学习方法，作为上限进行比较）。ED依赖于专家的行为示范，而AD通过多个回合的历史数据进行蒸馏，因此AD不依赖于专家轨迹，而是依赖于**跨回合的学习历史。**在与RL²的对比中，AD通过 **离线强化学习** 达到了接近RL²的效果，且 **训练速度比其他基线方法快。**AD通过 **历史信息蒸馏** 的方式，显著提高了 **样本效率**，使得模型在较少的交互数据下就能得到较好的结果。



第二部分：Memory

可以把记忆存储在向量数据库中，需要的时候再使用ANN等MIPS方法进行**最大内积搜索**，从而以损失少量的精度来换取巨大的速度提升。

第三部分：Tool Use

给大模型配备工具可以显著扩展大模型的功能以及处理复杂任务的能力。

MRKL (Modular Reasoning, Knowledge, and Language)

MRKL是一种 **神经符号架构**（neuro-symbolic architecture），旨在将 **语言模型（LLM）与专家模块** 相结合。这些模块可以是神经网络（例如深度学习模型）或符号系统（例如数学计算器、货币转换器、天气API等）。在MRKL系统中，LLM充当 **路由器**（router），根据任务的需求将查询请求发送给最适合的专家模块。换句话说，LLM决定哪个外部工具（如计算器、API等）适合处理当前的问题。MRKL框架的一个实验表明，LLM在解决 **口头算术问题** 时比在 **明确陈述的算术问题** 上更难得到正确答案，尤其是在无法正确提取算式的参数时。这揭示了LLM的 **能力局限性**，即知道何时以及如何使用外部工具是非常关键的。

TALM (Tool Augmented Language Models)

TALM通过**微调（fine-tuning）**语言模型，使其学会如何 **调用外部工具API**（例如搜索引擎、计算器等）。这种方法的核心在于：当模型学会了如何调用API并利用外部工具时，能更好地处理和解决特定任务。TALM通过增加**API调用的注解**来扩展训练数据集，并评估这些新增的API调用注解是否能够提高模型输出的质量。换句话说，模型通过学习如何使用外部工具，提升其对复杂任务的处理能力。

Toolformer

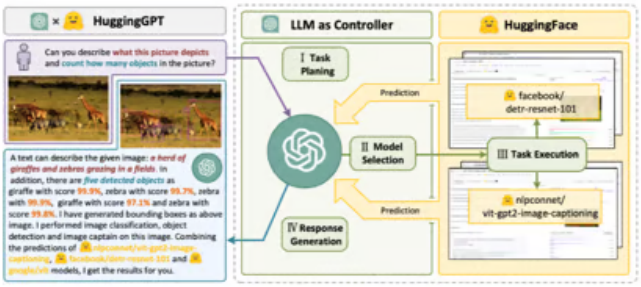
Toolformer是另一种微调语言模型的技术，目的是让语言模型学会 **使用外部工具API**，类似于TALM。不同之处在于，Toolformer框架更侧重于 **基于模型输出质量来选择合适的API调用。**通过微调，使得模型能够基于实际情况（例如任务的具体要求）自动选择和调用外部工具，从而提高任务完成的效率和准确性。

ChatGPT Plugins 和 OpenAI API Function Calling

- **ChatGPT Plugins**：ChatGPT的插件机制使得用户可以通过安装插件扩展ChatGPT的功能。这些插件通常由 **第三方开发者提供**，允许ChatGPT访问外部的 **服务或工具**（例如查询天气、计算机算式、执行交易等）。
- **OpenAI API Function Calling**：类似于插件，但功能更为 **自定义**，通过API调用的方式，允许开发者定义函数，使得LLM可以与这些函数进行交互，完成特定任务。

HuggingGPT

HuggingGPT是一个框架，它通过将 **ChatGPT** 作为任务规划器来 **选择合适的模型**，这些模型可以来自HuggingFace平台。HuggingGPT会根据 **模型描述** 来选择合适的任务执行模型，然后基于执行结果来 **总结回应。**



HuggingGPT的工作原理

该系统包括四个阶段：

- 任务规划：LLM作为大脑，负责将用户请求解析为多个任务，每个任务有4个关联的属性：任务类型、ID、依赖项、参数，通过few-shot的方式，指导LLM进行任务解析和规划。指令如下（中文版）：

```
AI助手可以将用户输入解析为多个任务：[{"task": 任务, "id": 任务_id, "dep": 依赖任务_id, "args": {"text": 文本, "image": 图片_URL, "audio": 音频_URL, "video": 视频_URL}}]。其中，"dep"字段表示当前任务依赖于先前任务生成的新资源。特殊标签"-task_id"指代依赖任务生成的文本、图片、音频和视频资源。任务必须从以下选项中选择：{{ 可用任务列表 }}。任务之间存在逻辑关系，请注意它们的顺序。如果用户输入无法解析，您需要回复空的JSON格式。以下是几个参考案例：{{ 示例演示 }}。聊天历史记录作为{{ 聊天历史 }}。从聊天历史中，您可以找到用户提到的资源路径来规划任务。
```

- 模型选择：LLM可以根据任务类型将请求分配给相应的专家模型，这些模型一般限制为多项选择题的类型。然后，LLM提供一个可供选择和使用的模型列表。由于上下文长度的限制，LLM需要基于任务类型进行过滤。

```
根据用户请求和调用命令，AI助手帮助用户从模型列表中选择一个合适的模型来处理用户请求。AI助手仅输出最合适模型的模型ID。输出必须严格遵循JSON格式："id": "id", "reason": "选择该模型的详细原因"。我们为您提供了可供选择的模型列表{{ 候选模型 }}，请选择一个模型。
```

- 任务执行：专家模型执行并记录特定任务的结果。

```
根据用户输入和推理结果，AI助手需要描述处理过程和结果。前面的阶段可以形成如下内容：

- 用户输入：{{ 用户输入 }}
- 任务规划：{{ 任务 }}
- 模型选择：{{ 模型分配 }}
- 任务执行：{{ 预测结果 }}

你必须首先直接回答用户的请求。然后描述任务过程，并以第一人称的方式向用户展示分析和模型推理结果。如果推理结果包含文件路径，必须告知用户完整的文件路径。
```

- 响应生成：LLM接收执行结果并向用户提供汇总结果。

挑战

1、有限的上下文长度：历史信息、指令的细节、API call的内容、模型的回复，其实都需要放到上下文中，尤其是像self-reflection这种方式依赖于从过去的错误中学习，缺乏长期记忆会导致Agent分心，从而无法专注于目标，导致不可预测的行为，所以需要比较长的上下文窗口，虽然现在有一些方法比如外挂向量知识库的方式解决试图这个问题，但也不如完全将这些信息都包括在上下文中效果好。

2、长期计划和任务分解：对于比较长期和复杂的任务而言，LLM还做不到像人类那样，从错误中学习和修正自身行为。

3、过于依赖自然语言：当前LLM仍然存在幻觉、输出内容不标准等问题，所以对模型输出进行解析和后处理也是一个关注点。

参考：Weng, Lilian. (Jun 2023). "LLM-powered Autonomous Agents". Lil'Log.