中国科学院大学
University of Chinese Academy of Sciences

自然语言处理

第12讲 智能体

王 石 资康莉 刘 瑜

2026年春季课程

https://ictkc.github.io/teaching/

第十二讲
智能体（Agent）

Agent Smith in ***Matrix***

中国科学院大学
University of Chinese Academy of Sciences

目 录

什么是智能体?

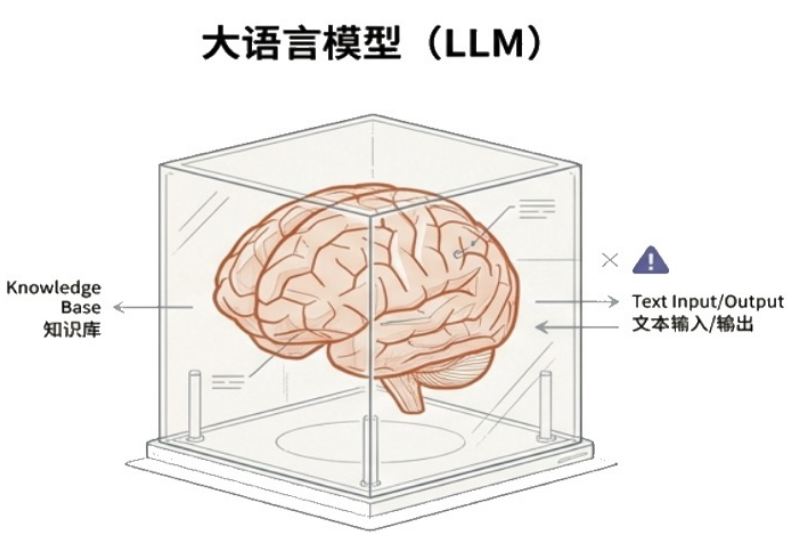
- （大模型驱动的）**智能体**是指依靠自身的感知、决策、记忆、执行模块，独立完成特定任务的大模型应用系统





智能体*v.s.*大模型

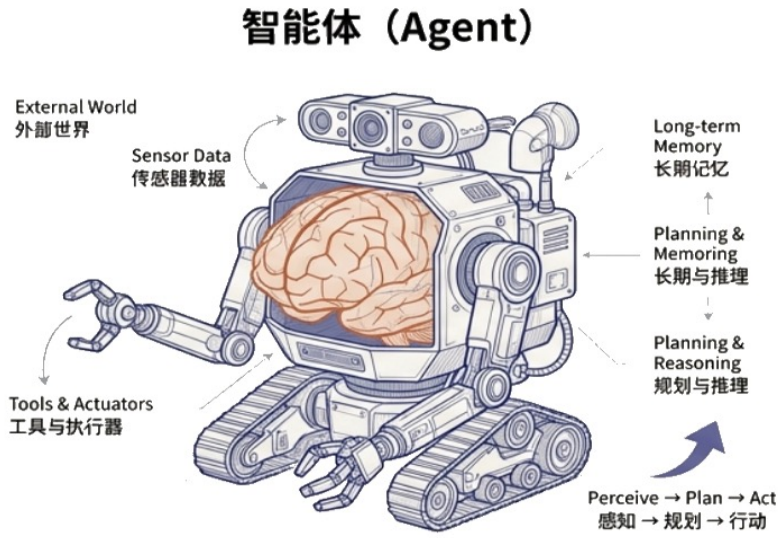
- 大模型和智能体：思想家和实干家



思想的巨人，行动的矮子

知道“如何订机票”。能写出完美的订票步骤，甚至模拟对话，但无法操作浏览器、填表或支付。它被困在文本的牢笼中。

无法真正帮你订到机票



配备手脚的实干家

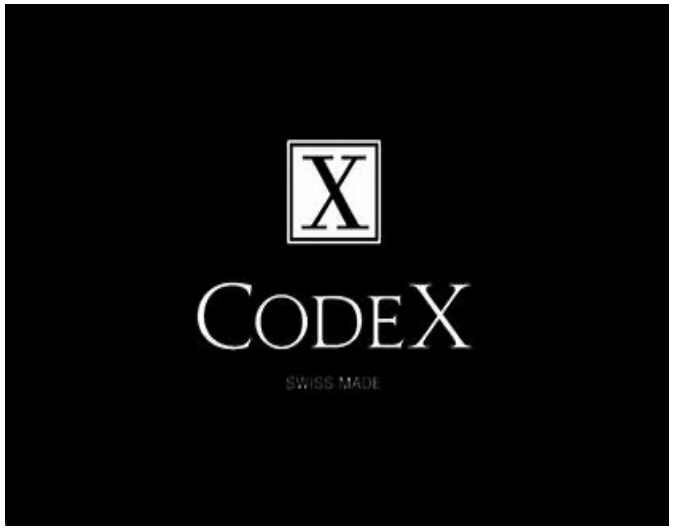
具有自主性与反应性。LLM是中央处理器，向外挂载感知器官、工具肢体、记忆系统。它不仅能“想”，更能“做”。

能够帮你真正订到机票

智能体背景

- 当今智能体技术发展迅速，如 OpenClaw、Codex、Claude Code 等，它们具备感知、推理、规划和工具调用能力，能处理更复杂的任务

openclaw



codex

claude code

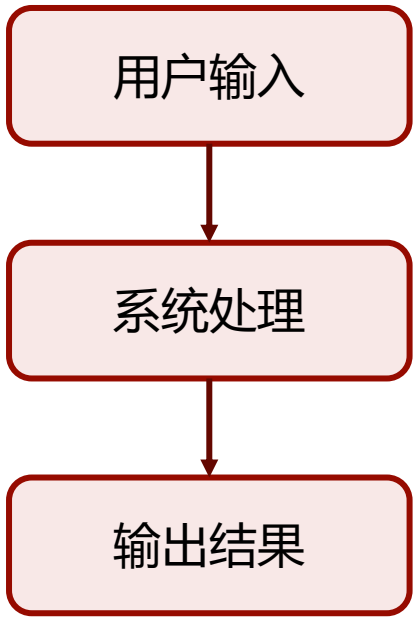
智能体背景

- 2025年，国务院颁发《关于深入实施“人工智能+”行动的意见》，首次将智能体纳入国家战略，明确提出发展“智能体即服务（Agent as a Service）”
- 2026年，国务院印发《关于推进服务业扩能提质的意见》，明确提出深入实施“人工智能+”行动，加快智能编程工具研发使用，支持采购大模型、智能体服务

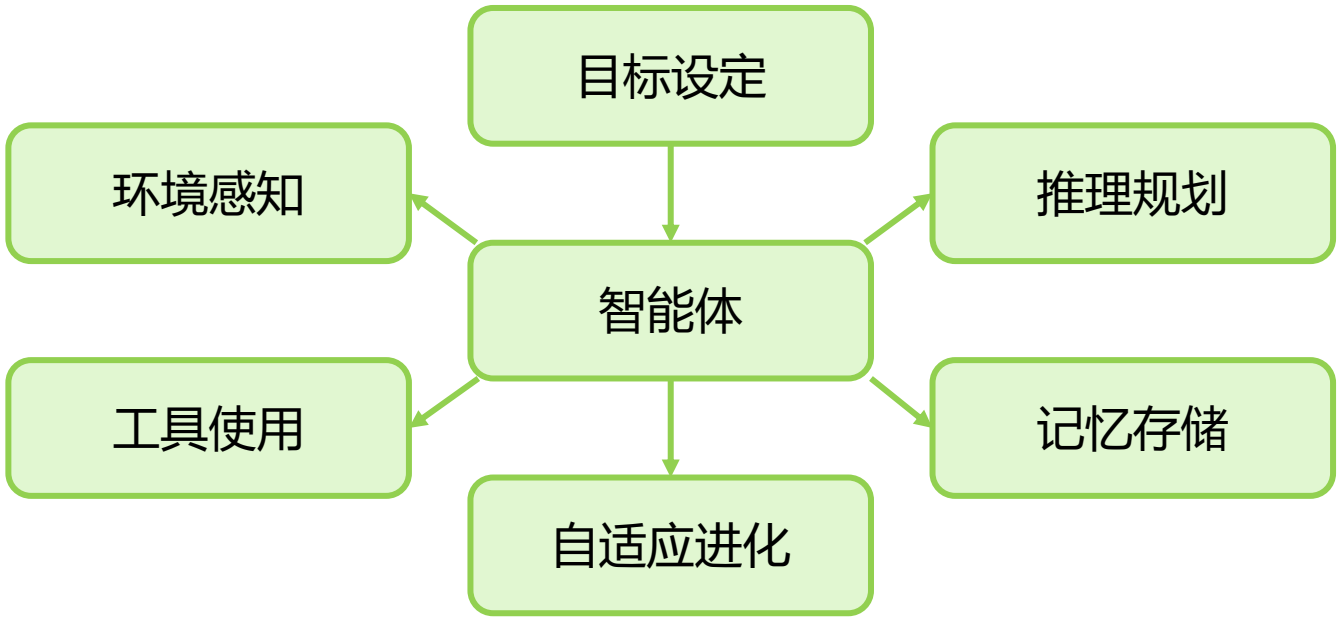
为什么需要智能体?

- 智能体代表人工智能从"语言理解工具"向"**自主行动体**"的进化

被动式系统

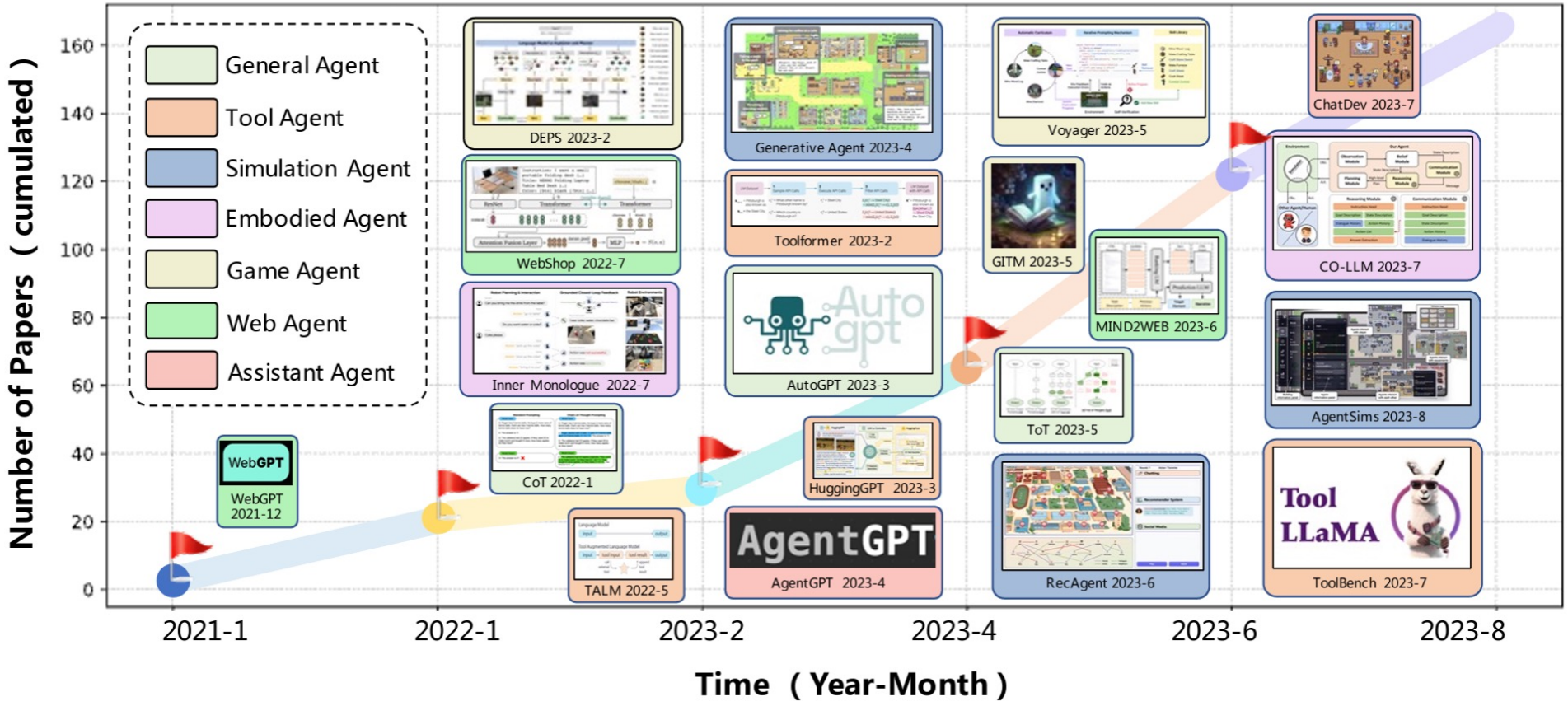


智能体



智能体发展

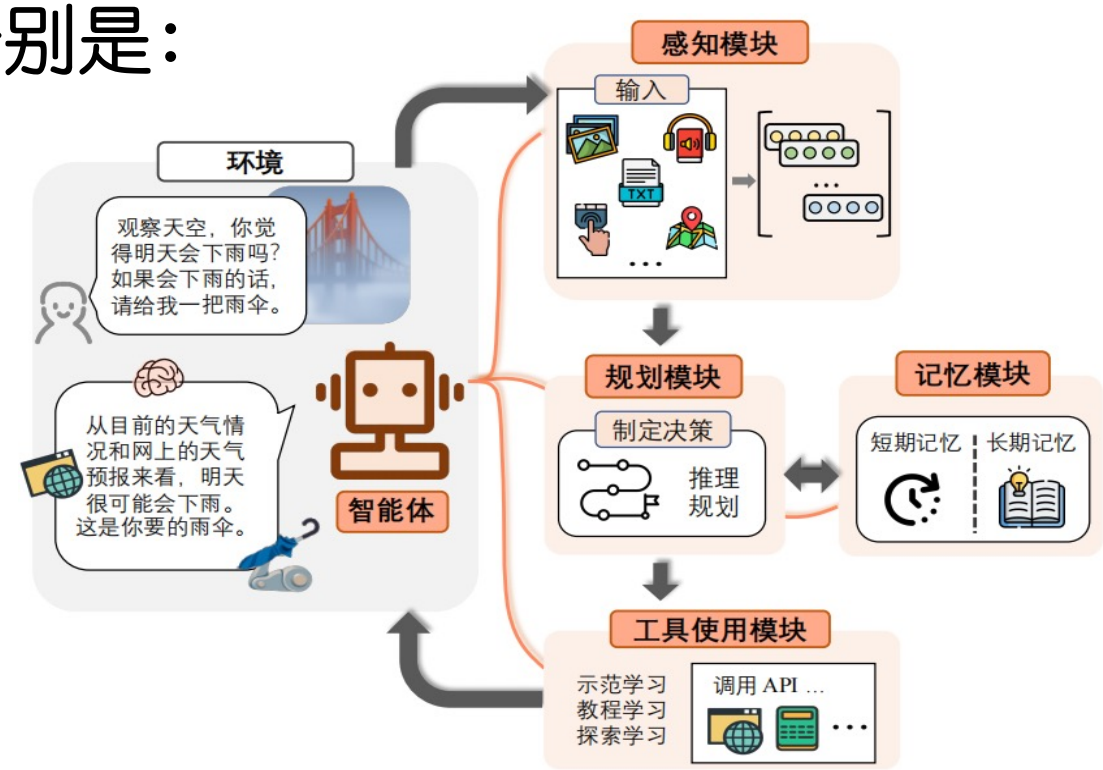
❑ LLM智能体近年来发展历程图



A Survey on Large Language Model based Autonomous Agents. 2023

智能体组成

- 如何构建智能体，使其能够像人一样解决真实世界上的问题?
- 智能体由四个模块组成，分别是：
  - 感知模块
  - 规划模块
  - 记忆模块
  - 工具调用模块

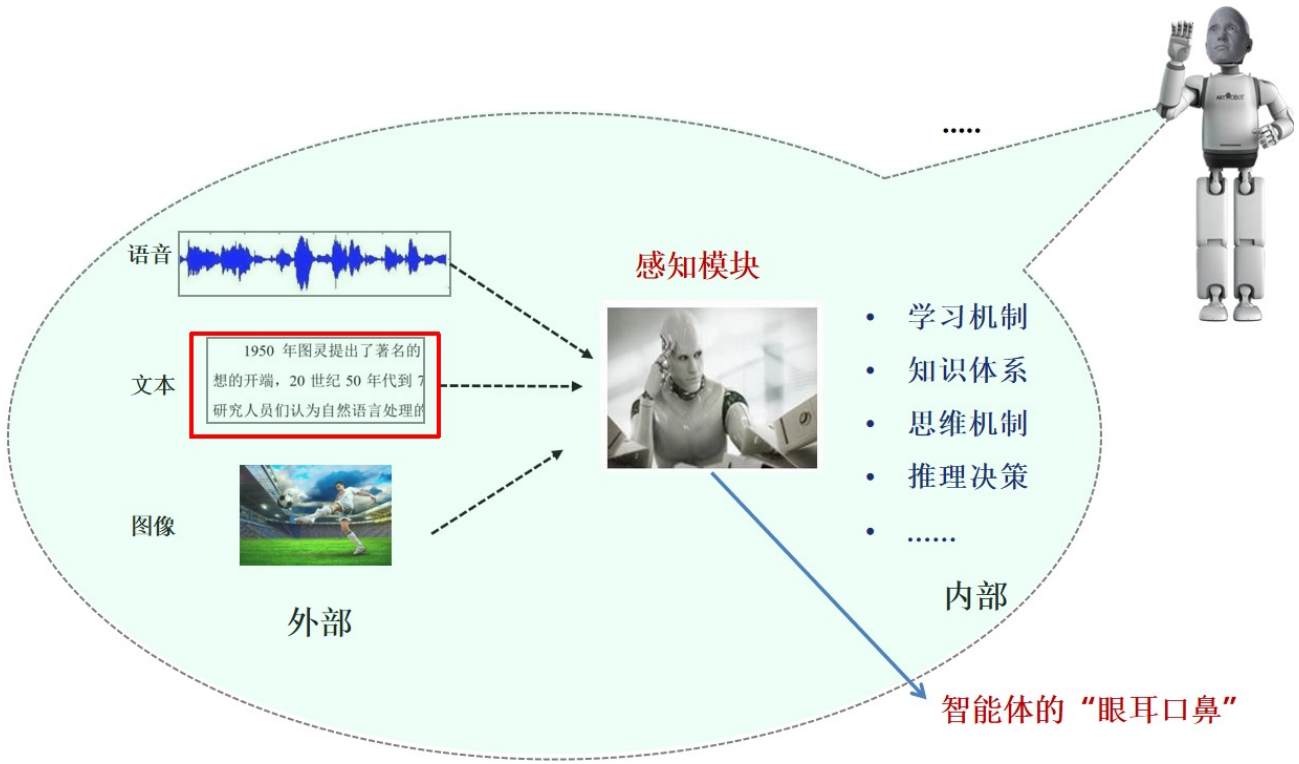


WebArena: A Realistic Web Environment for Building Autonomous Agents. 2023

目录

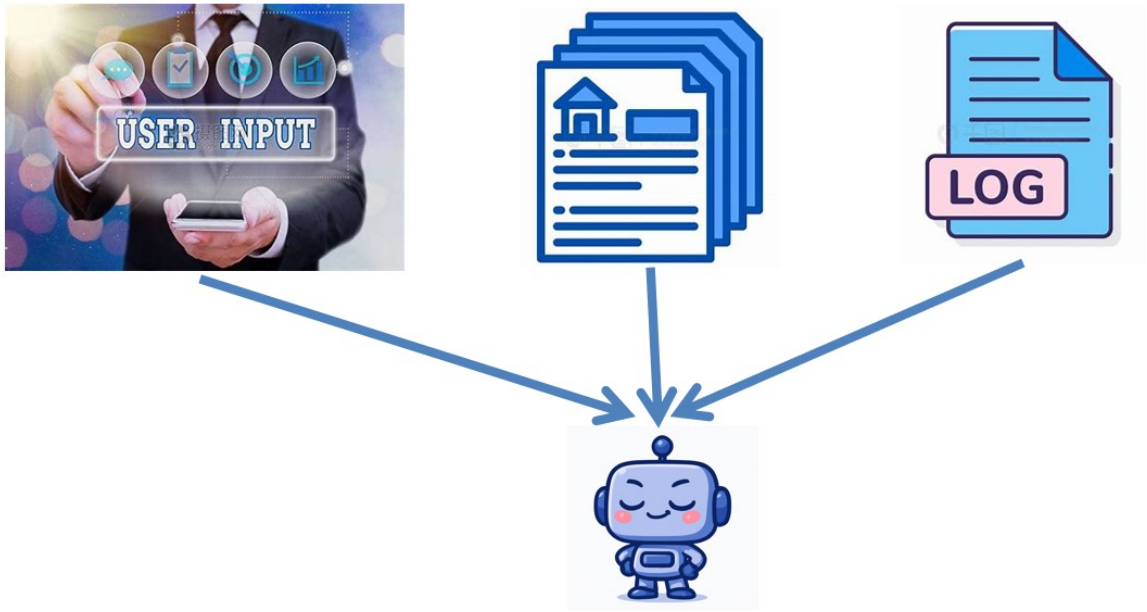
感知模块

- 感知模块是智能体与外部环境交互的首要环节，相当于**智能体的“感官”**。
  它收集周围环境的各种多模态信息，为智能体提供决策依据



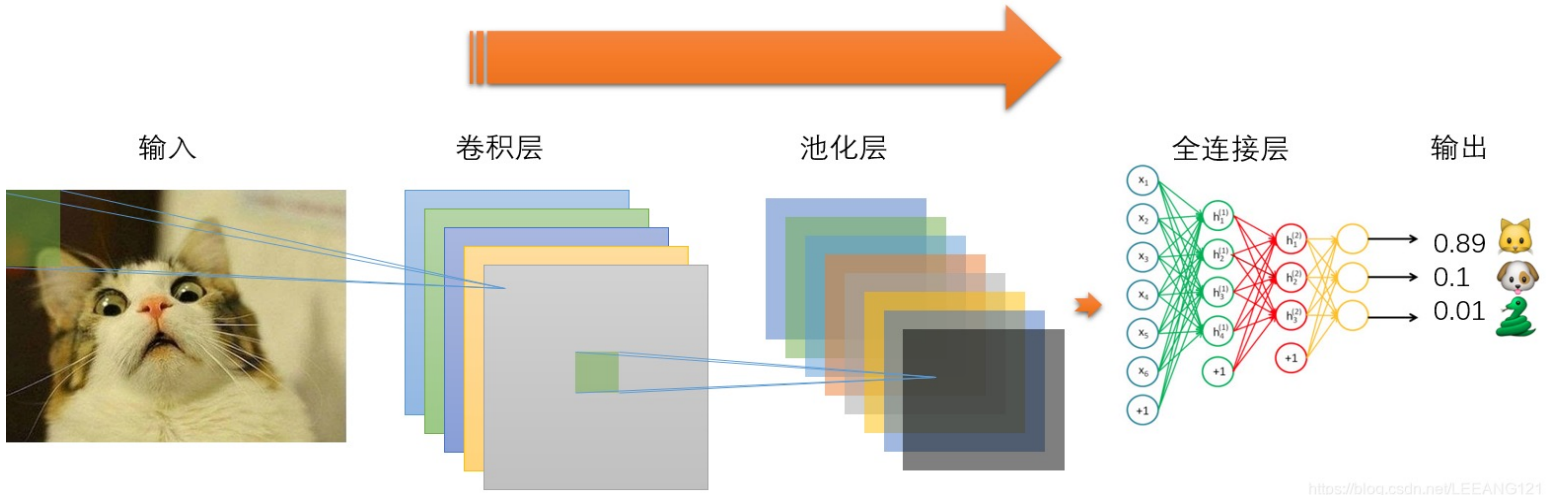
文本信息感知

- 文本信息主要包括用户输入，文档内容，日志/网页数据等非标准化文本
- 为了使得智能体理解文本内容，需要把这些文本转化为智能体大脑可识别的标准化信息，通常处理模型是基于Transformer架构的预训练模型



图像信息感知

- 图像是另一类重要的信息来源，比如照片、视频帧或扫描图像
- 图像感知主要利用视觉编码器，把视觉信息转化为向量表示，再和文本向量融合，实现跨模态理解



通过CNN提取图像特征

语音信息感知

- 语音感知的处理流程包括：
  - 将连续声波转为机器可计算的声学特征，如梅尔频谱、MFCC 声学特征，剥离无效波形信息，保留人声关键特征
  - 通过语音模型，把声学特征转写为自然语言文本
  - 将语音解析出的文本、情绪特征，与文本、图像感知信息统一编码

应用场景

- 给出两个典型案例：自动驾驶和智能安防

自动驾驶：通过摄像头、激光雷达和麦克风等设备收集道路图像、距离信息和周围声音，从而帮助汽车准确判断路况

智能安防：通过监控摄像头和麦克风获取图像和声音信息，结合文字描述，能够更准确地识别异常行为

目 录

规划模块

- 为什么智能体需要规划?

为部门组织一次跨城市技术研讨会

需要分为以下步骤执行:
1.明确需求、搜索并筛选合适城市与场地
2.比较交通与住宿成本
3.邀请嘉宾并协调时间
4.制定详细日程
5.安排预订场地与酒店
6.准备物料与宣传
7.应对突发情况（嘉宾变更、预算调整等）

规划模块

- 为什么智能体需要规划?

为部门组织一次跨城市技术研讨会

复杂任务通常不是单步完成，而是多步组织与规划。

需要分为以下步骤执行：
1.确定举办城市与场地
2.比较交通与住宿成本
3.邀请嘉宾并协调时间
4.制定详细日程
5.安排预订场地与酒店
6.准备物料与宣传
7.应对突发情况（嘉宾变更、预算调整等）

基本概念

- 规划（Planning）：根据交互的**环境**生成「**动作序列**」以最优/可行方式达成**目标**的过程
- 必要的项：E（**环境**）、g（**目标**）、p（**动作序列**），规划可表示为公式：

$$p = plan(E, g)$$

符号方法 → 与LLM结合

相关概念辨析

1. 规划VS搜索

搜索：强调在众多的可能性中找到一个可行或较优解，是规划常用的实现机制之一

规划：强调如何在当前状态，在满足约束的条件下到达目标状态，是目标驱动的任务求解

2. 规划VS推理

推理：强调从信息到结论的思维过程

规划：强调从目标到行动的组织过程，并考虑约束、反馈和调整

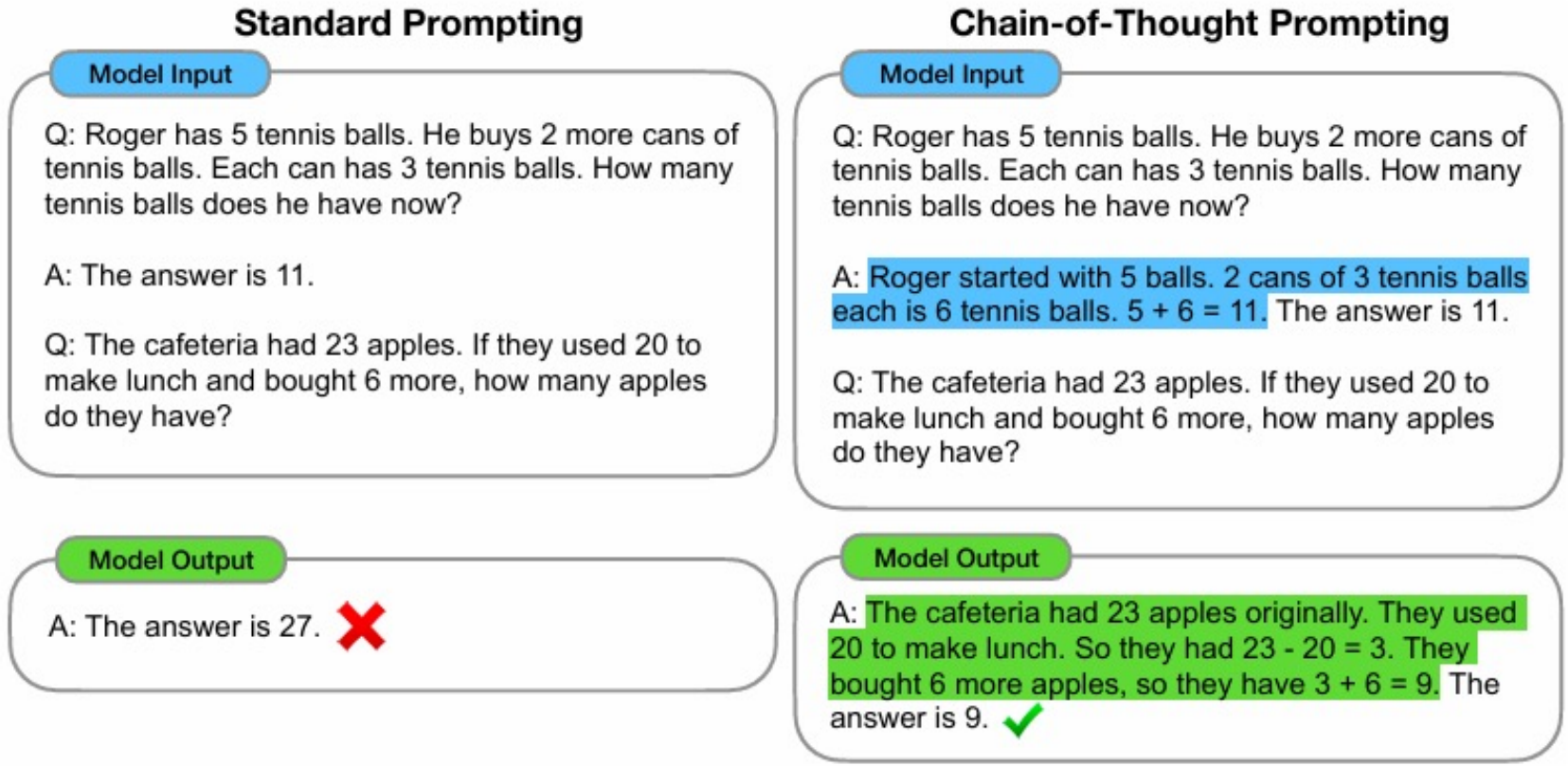
3. 规划VS决策

决策：强调在某一时刻从若干候选动作中做出选择

规划：强调面向未来的一系列行动安排

回顾CoT

- Chain-of-Thought引导大模型显式生成中间推理步骤



回顾CoT

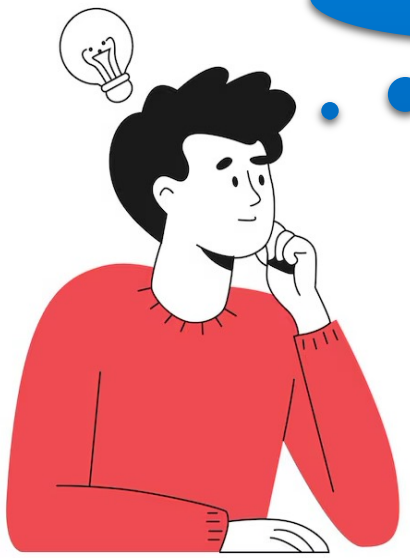
❑ 问题讨论

“Chain-of-Thought 已经能够让模型逐步思考，因此 CoT 本质上就等于智能体规划。”

请判断该说法是否成立，并说明理由。

回顾CoT

“Chain-of-Thought 已经能够让模型逐步思考，因此 CoT 本质上就等于智能体规划。”
请判断该说法是否成立，并说明理由。



不成立！

1. CoT 是静态推理路径，规划是动态决策过程
2. CoT 不涉及行动与环境交互，规划需要根据环境变化与执行反馈进行调整
3. CoT 是固定推理，通常一次生成完成，很少自我修正；规划是可迭代、可反思的
4. CoT 是推理手段，可以作为实现规划的一种方式

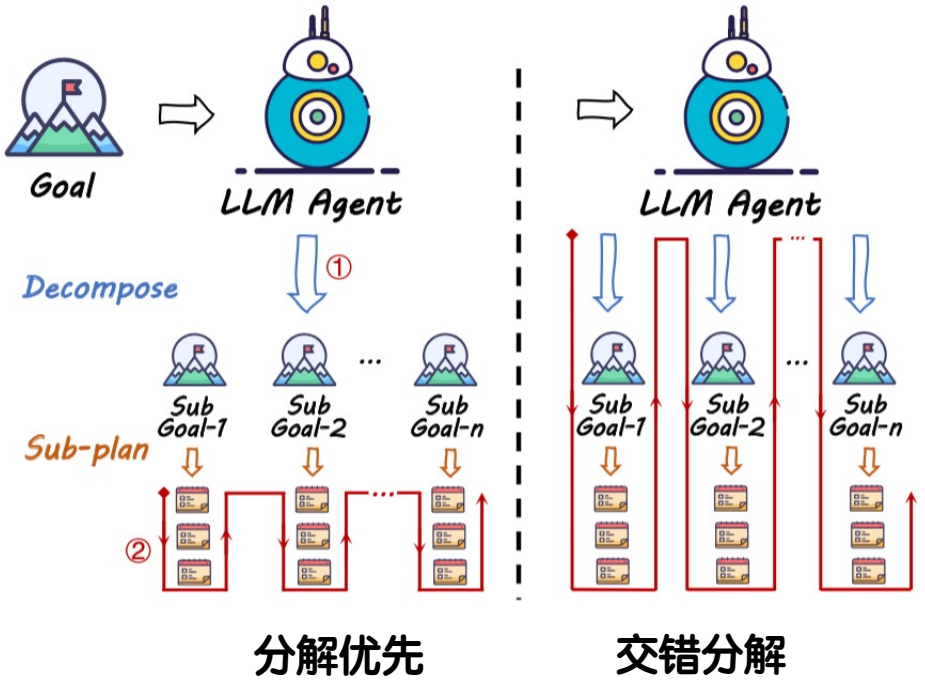
方法分类

- 基于任务分解的规划
- 基于搜索增强的规划
- 基于外部模块的规划
- 基于反馈修正的规划

1. 基于任务分解的规划

- **任务分解**两个关键步骤：
  1. 分解：将复杂任务分解成若干个子任务
  2. 子计划：为每个子任务制定计划

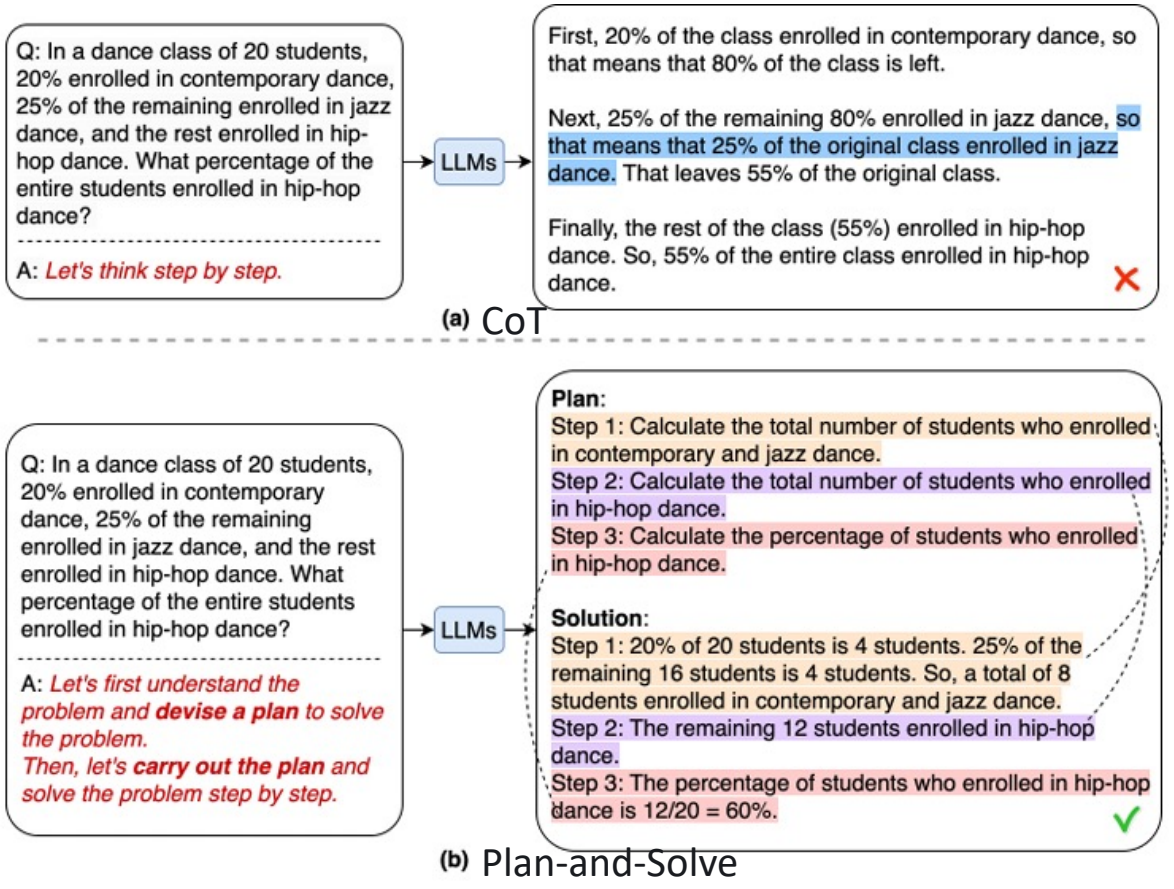
- 任务分解方法分为两类：
  - **分解优先**：将任务分解为子目标，依次为每个子目标制定规划
  - **交错分解**：在任务分解和子任务规划之间进行交错，每次只揭示当前状态的一个或两个子任务



Understanding the planning of LLM agents: A survey. 2024

分解优先式：Plan-and-Solve

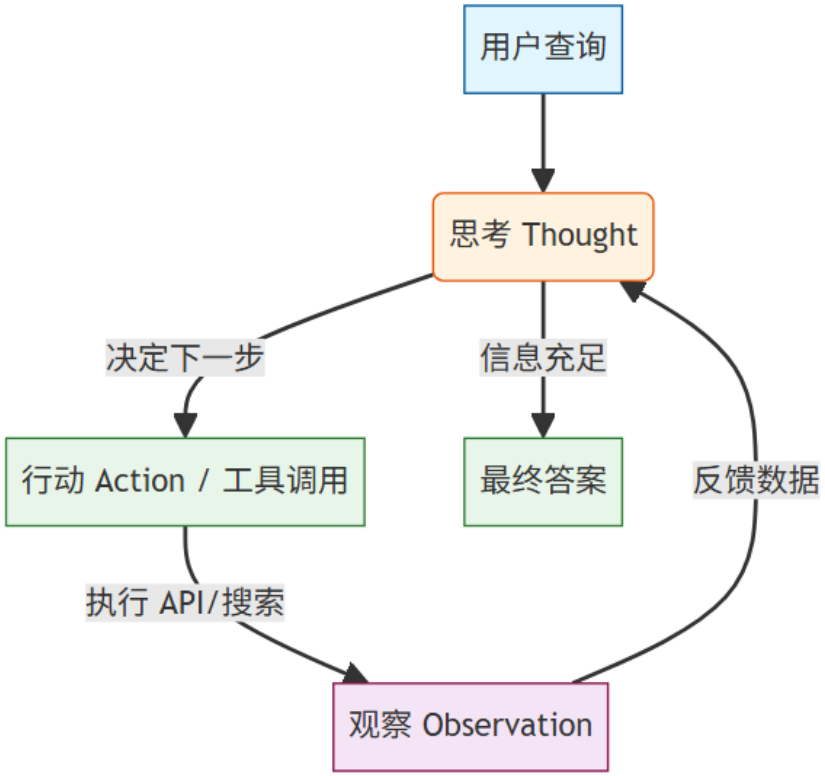
- Plan-and-Solve：
  - 设计计划将整个任务分解为较小的子任务
  - 按照计划执行子任务
- 避免了零样本CoT中手动生成多步推理示例的需求



Plan-and-Solve Prompting: Improving Zero-Shot Chain-of-Thought Reasoning by Large Language Models. 2023

交错分解式：ReAct

- ReAct 在推理和行动之间交替循环，直到模型任务已经获得最终答案才停止
- 整个过程由三个阶段组成：
  - **思考**：LLM 分析当前问题的状态，并决定接下来需要什么信息
  - **行动**：LLM 选择一个特定的工具（如搜索、计算器、天气API），并提供必要的输入参数
  - **观察**：系统执行该工具，并将原始输出结果返回给 LLM
- 基于新观察结果，模型生成一个新的思考，再次进入下一轮循环



ReAct: Synergizing Reasoning and Acting in Language Models. ICLR 2023

现任美国总统和苹果公司 CEO 年龄差多少?

- 思考 1：我需要查明现任美国总统，以及苹果的CEO。然后我需要他们的年龄。先从美国总统开始。
- 行动 1：搜索[现任美国总统 2026]
- 观察 1：[搜索结果： ...]

- 思考 2：好的，总统是唐纳德·特朗普，今年79岁。现在我需要找苹果的CEO。
- 行动 2：搜索[苹果 CEO 2026]
- 观察 2：[搜索结果： 蒂姆·库克...]

- 思考 3：蒂姆·库克是 CEO。我来查查他的年龄。
- 行动 3：搜索[蒂姆·库克 年龄]
- 观察 3：[搜索结果： 蒂姆·库克 65 岁...]

- 思考 4：总统 79 岁，库克 65 岁。差值是 79 - 65。我需要计算一下。
- 行动 4：计算器[79 - 65]
- 观察 4：[结果： 14]

- 思考 5：我已经得出了最终答案。
- 最终答案："他们的年龄差是 14 岁。"

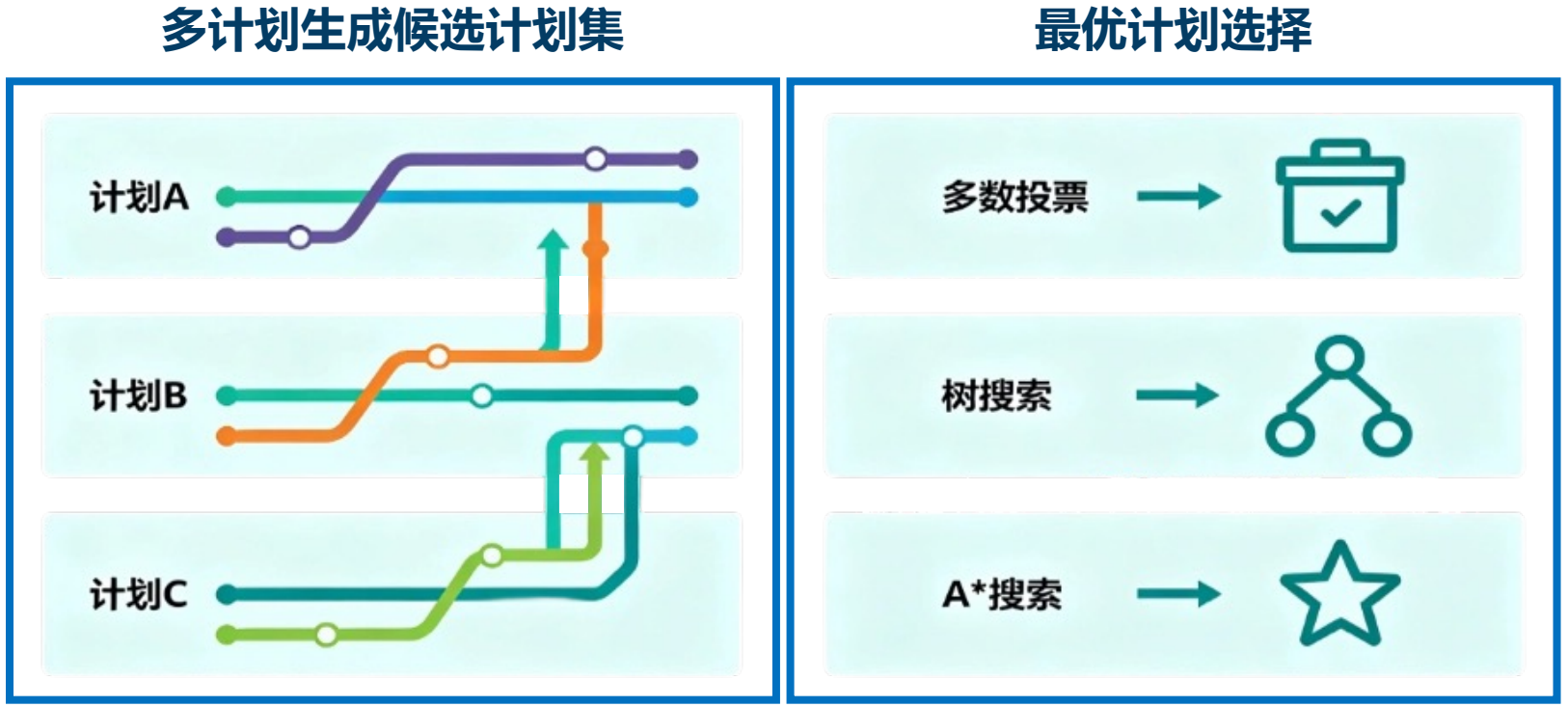
1. 基于任务分解的规划

❑ 优缺点总结

	优点	缺点
分解优先	子任务和原始任务关联性强，**减少了任务遗忘和幻觉**	预先设定的**子任务可能不准确**，需要额外调整
交错分解	可以根据环境反馈调整分解，**容错性更高**	交错分解**过长的轨迹**可能导致智能体在后续出现**幻觉**

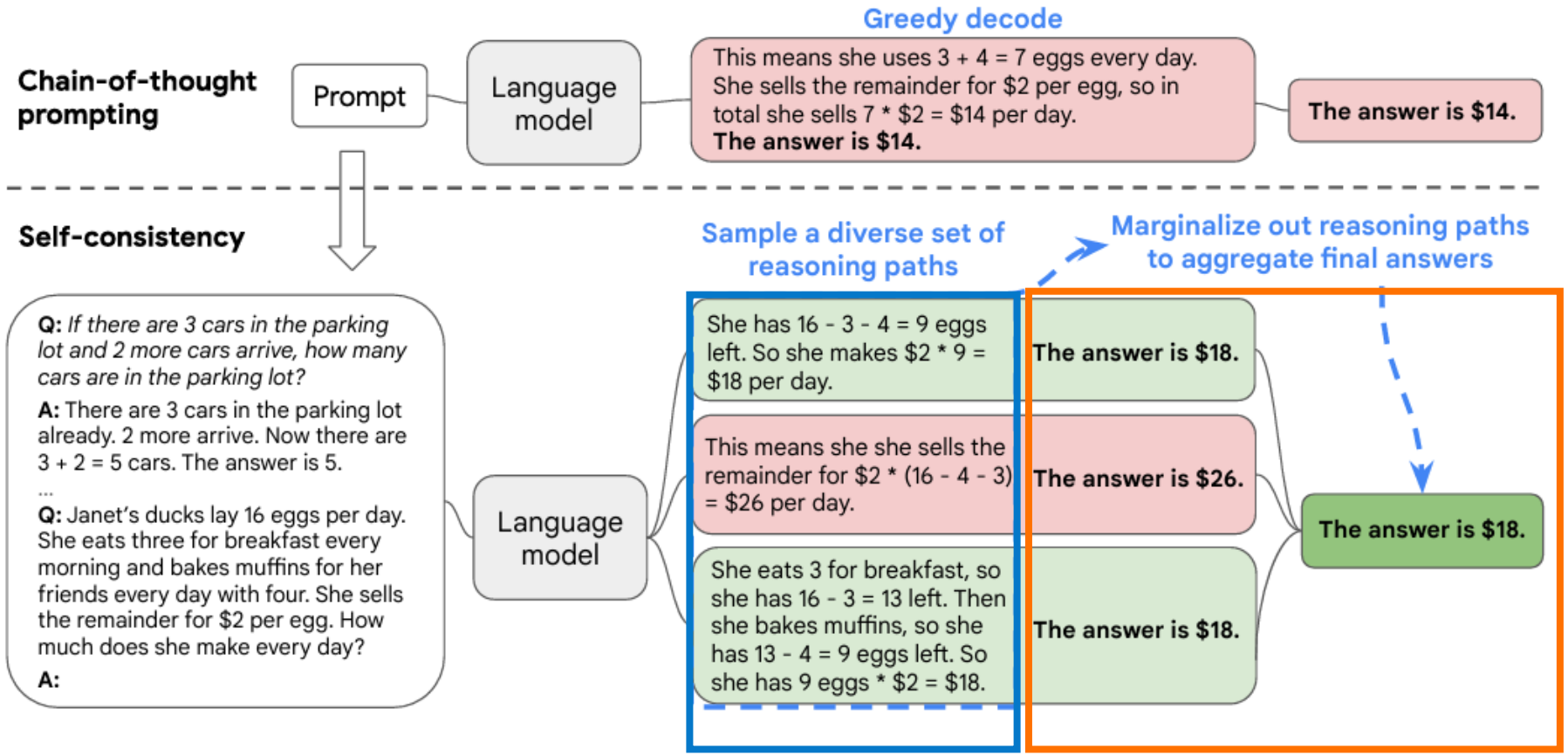
2. 基于搜索增强的规划

- 基于搜索增强的方法为任务**生成多个计划**并使用与任务相关的搜索算法**选择一个**计划执行



CoT-SC

- CoT-SC是对CoT方法的改进，采用多次采样的思想



多计划生成方式：

- **解码时通过采样获得多个不同的推理路径**

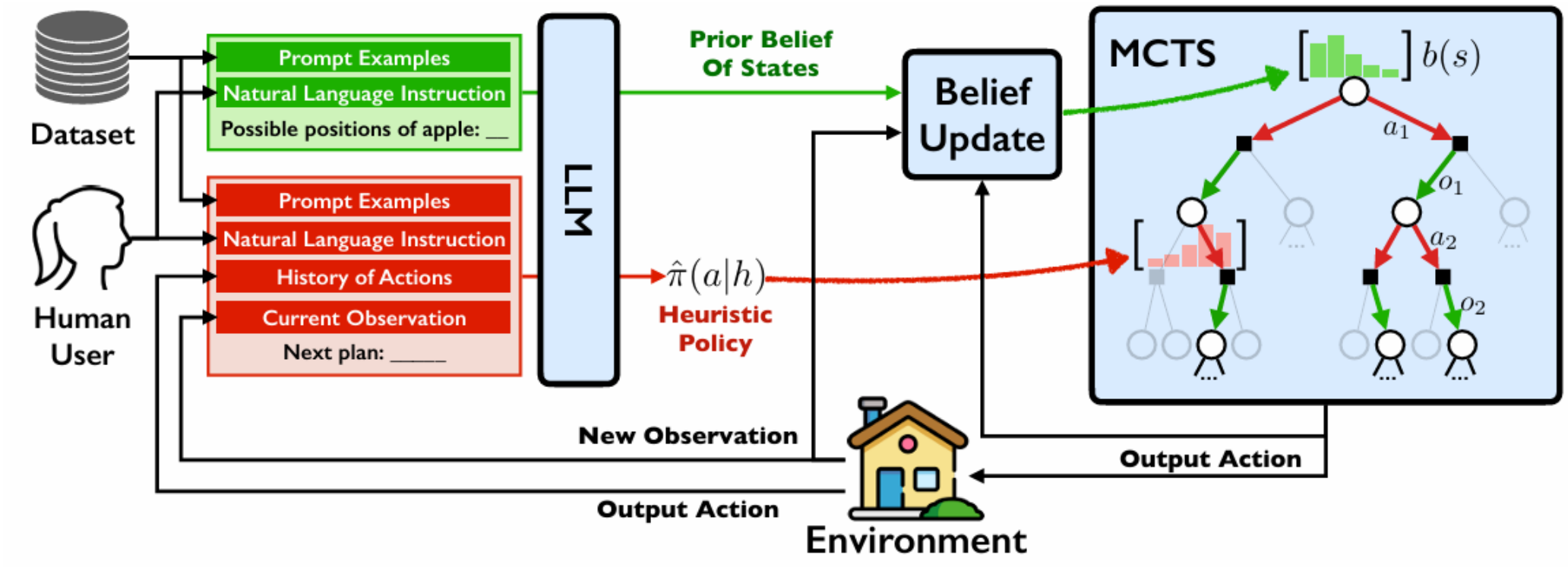
最优计划选择方式：

- **多数投票选择一致性最高的结果**

Self-Consistency Improves Chain of Thought Reasoning in Language Models. ICLR 2023

LLM-MCTS

- 通过LLM和MCTS的协作实现了多计划生成和最优计划选择



Large Language Models as Commonsense Knowledge for Large-Scale Task Planning. NeurIPS 2023

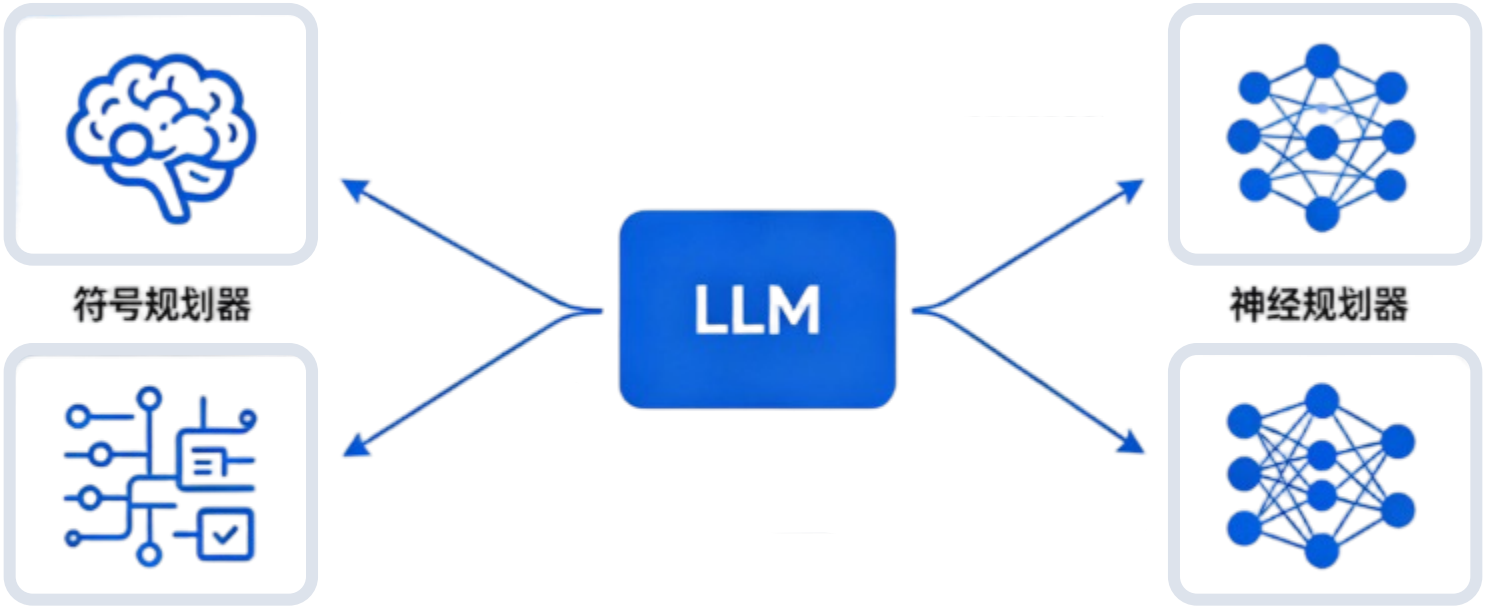
2. 基于搜索增强的规划

❑ 优缺点总结

	优点	缺点
搜索增强	可扩展性更强，能在庞大的搜索空间中更广泛地探索潜在解决方案	• 计算需求增加，成本更高 • 对LLM的能力的依赖性高 • LLMs的随机性质增加了选择的随机性，可能影响所选方案的一致性和可靠性

3. 基于外部模块的规划

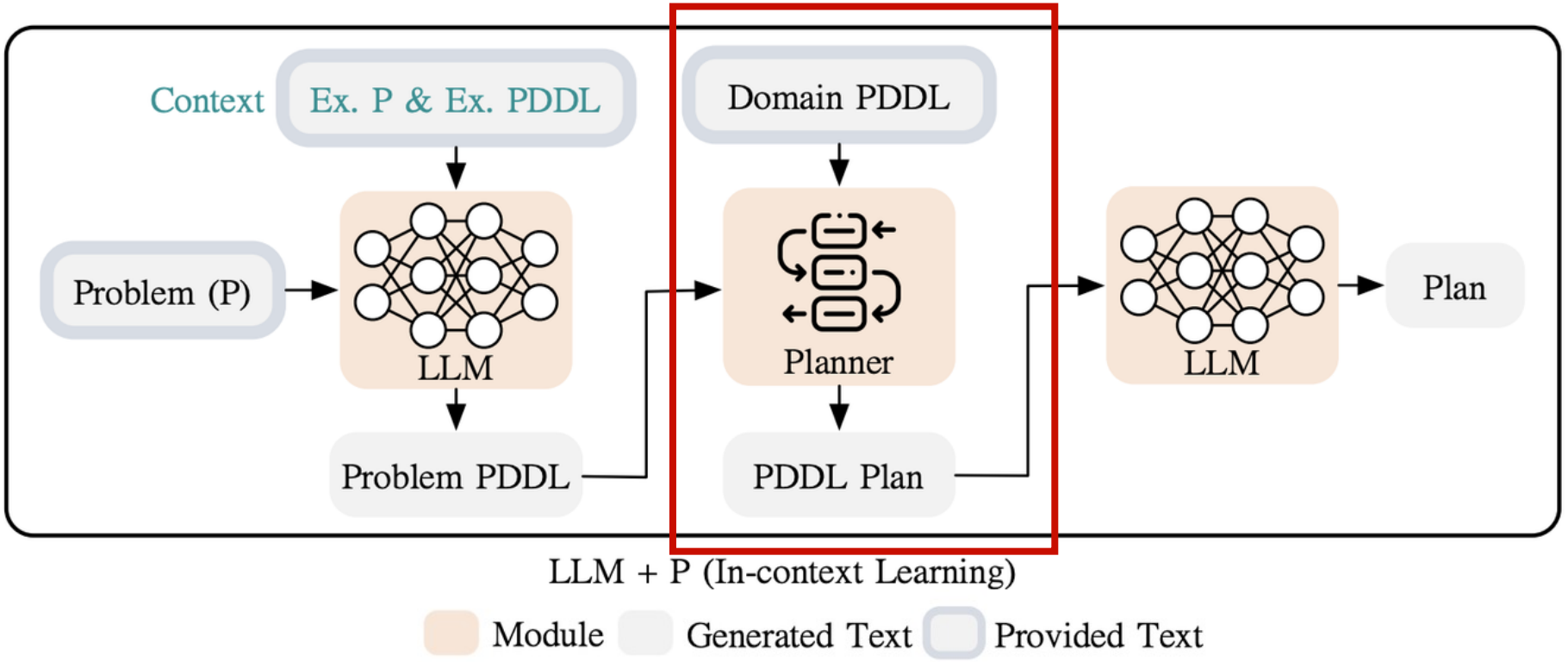
- 基于外部模块的方法将LLM和外部规划器集成，根据引入的规划起分为符号规划器和神经规划器



LLM与外部规划器集成规划

符号规划器：LLM+P

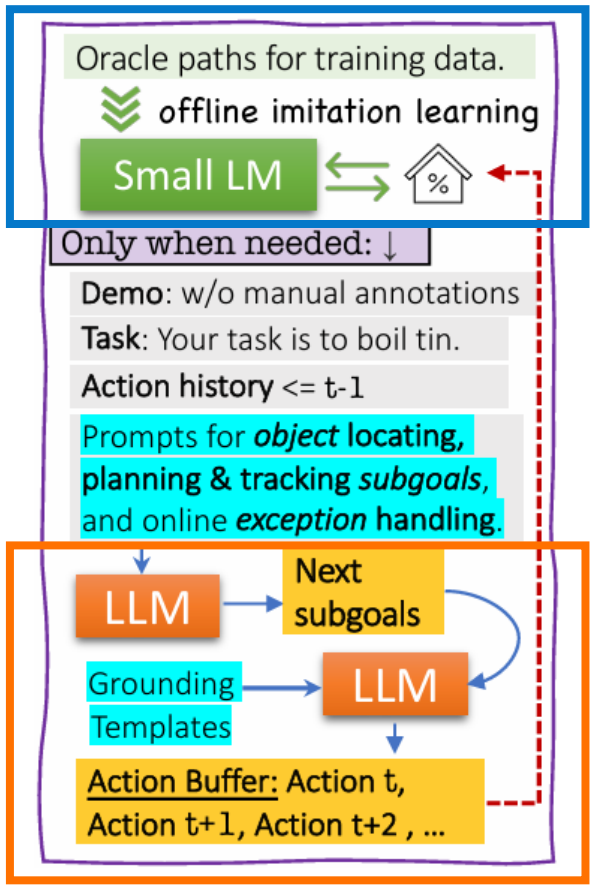
- 将自然语言描述转化为一个用规划领域定义语言（PDDL）编写的文件，然后利用**传统规划器**找到解决方法



LLM+P: Empowering Large Language Models with Optimal Planning Proficiency. 2023

神经规划器：SwiftSage

- 利用认知心理学中的双过程理论，将规划过程分为慢思考和快思考



SWIFT 模块：

一个小型的编码-解码语言模型，通过模仿学习对其进行微调，用于模拟人类的直觉思维。它能够快速解码出下一个动作，适合简单且直接的任务

快思考VS慢思考

SAGE 模块：

使用类似于 GPT-4，模拟深度分析的推理过程。SAGE 模块分为两个阶段：**规划阶段和执行阶段**。规划阶段负责生成高层次的任务建议，执行阶段则将这些建议转化为可执行的具体操作

SwiftSage: A Generative Agent with Fast and Slow Thinking for Complex Interactive Tasks. NeurIPS 2023

3. 基于外部模块的规划

❑ 优缺点总结

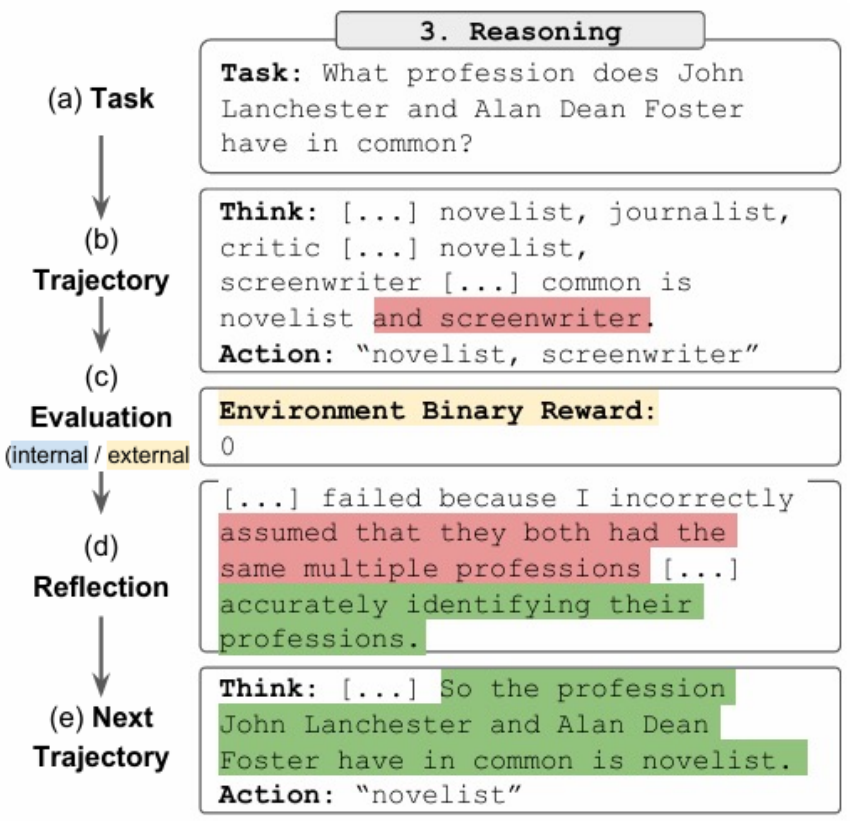
	优点	缺点
外部模块	• 外部规划器能够显式建模，提升推理过程的稳定性、可解释性 • 大模型具备较强的语义理解与知识泛化能力	• 外部规划器通常依赖人工设计的规则或状态空间，构建成本较高 • LLM 与规划器之间的信息交互不够紧密

4. 基于反馈修正的规划

- 类似于反馈，鼓励LLM反思失败并改进计划

Reflexion

❑ 借鉴人类反思的过程，使用语言反馈信号来帮助agent学习



对于推理任务

任务目标：判断两个人的共同职业

智能体思考：共同职业是小说家和编剧

环境反馈：环境给出奖励反馈

进行反思：纠正错误，重新分析

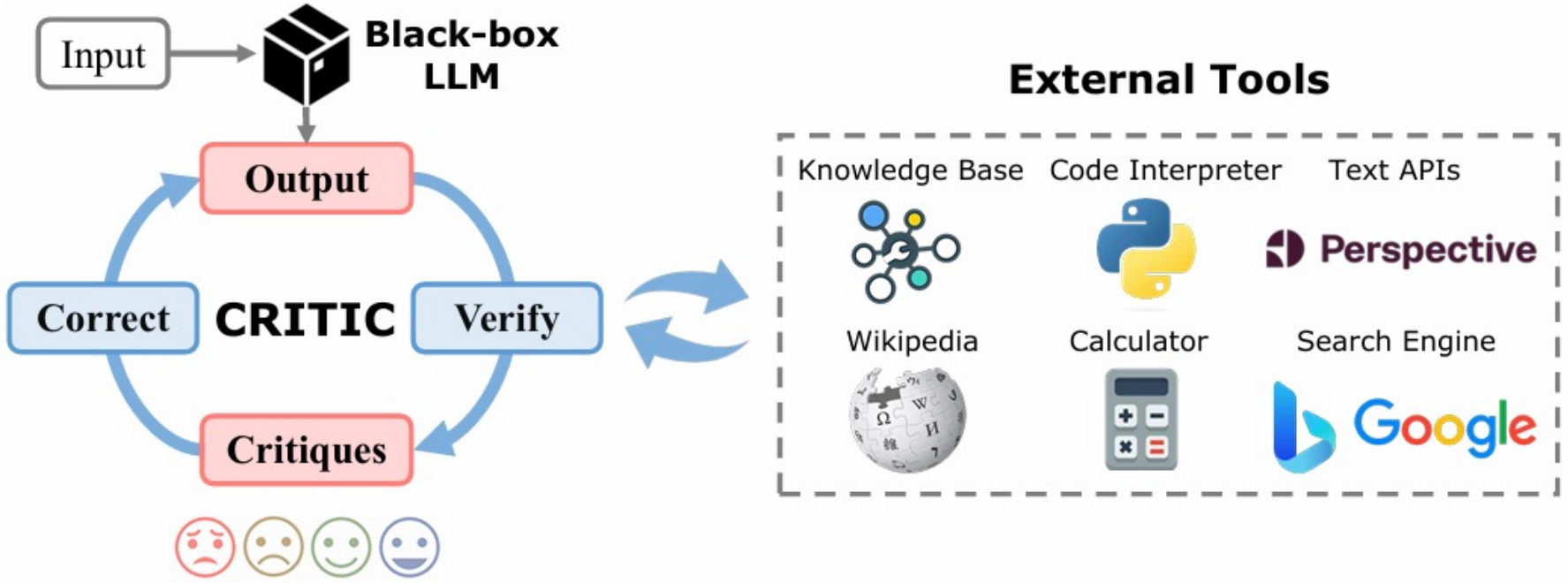
错误修正：发现共同职业只有小说家

Reflexion: Language Agents with Verbal Reinforcement Learning. NeurIPS 2023

CRITIC

- 借鉴了人类的认知和批判性思维，不断验证和修正模型的输出

第一步与外部工具互动验证结果并提出批评；第二步根据这些批评对结果进行修正



CRITIC: Large Language Models Can Self-Correct with Tool-Interactive Critiquing. ICLR 2024

4. 基于反馈修正的规划

❑ 优缺点总结

	优点	缺点
反馈修正	• 模型能够根据执行结果动态修正错误，提升复杂环境下的适应能力 • 反思结果可作为长期或短期记忆，增强后续决策的一致性与连续性	• 反思过程依赖大语言模型自身生成，容易受到幻觉 • 文本形式的更新缺乏理论上的收敛保证，无法证明持续反思一定能够引导 Agent 达到目标

目 录

记忆模块

❑ 为什么智能体需要记忆?

LLM 天生无状态

- 每次调用完全独立
- 参数部署后静态固化
- 无法跨调用保留信息

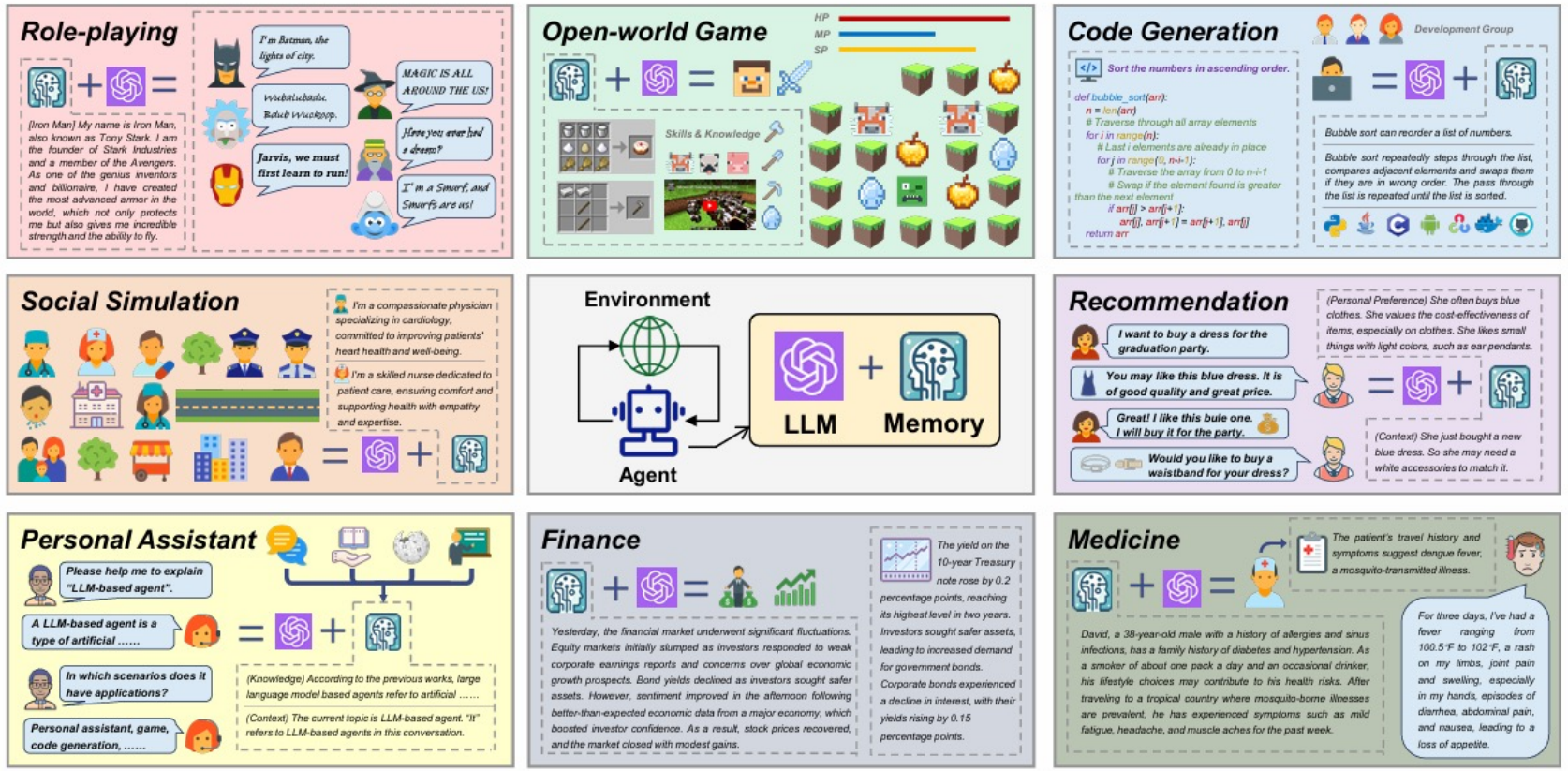
真实任务要求连续性

- 跨轮对话理解
- 跨任务经验积累
- 多步执行状态管理

记忆是 Agent 从「单次调用」变成「持续智能」的关键 —— 使其能够持续适应、跨时间演化

记忆模块

- 记忆贯穿所有智能体应用场景



A Survey on the Memory Mechanism of Large Language Model based Agents. 2024

基本概念

❑ 智能体记忆是智能体跨时间步保留和利用信息的持久认知状态，使静态的 LLM 获得跨时间适应与持续进化的能力

记忆的生命周期

写入	$\mathcal{M}_{t+1}^{\text{form}} = F(\mathcal{M}_t, \phi_t)$	从原始交互产物中选择性地提炼有价值的内容
演化	$\mathcal{M}_{t+1} = E(\mathcal{M}_{t+1}^{\text{form}})$	整合冗余、解决冲突、丢弃遗忘低价值信息、重组结构
读取	$m_t^i = R(\mathcal{M}_t, o_t^i, \mathcal{Q})$	根据当前观察和任务构造查询，返回相关记忆

相关概念辨析

	LLM记忆	RAG	上下文工程	智能体记忆
本质	模型内部的 信息保持机制	访问静态外部知识	单次上下文资源调度	持久的自演化 认知状态
是否跨任务持续	否	知识库持续， 但检索不跨任务关联	否	是
是否随交互演化	否	否	否	是
关注点	模型能"装下"多少	模型能"查到"什么	怎么"打包" 这一次输入	智能体"记住"了什么
与智能体记忆的区分	只优化模型表征容量， 不涉及记忆生命周期	知识库静态， 不随交互自我组织	只管当前调用的信息 排布，不跨任务持续	—

记忆的功能类型

❑ 三种记忆功能类型：

- **工作记忆（Working Memory）**：任务中的临时工作台，管理有限上下文信息，任务结束后清空 → 核心问题：保留什么？
- **事实记忆（Factual Memory）**：跨任务持久化的用户和环境信息 → 核心问题：世界是什么样的？
- **经验记忆（Experiential Memory）**：任务中积累的能力型知识，使智能体持续进化 → 核心问题：我该怎么做？

工作记忆解决容量问题，事实记忆实现知识跨任务延续，经验记忆则从经验中学习

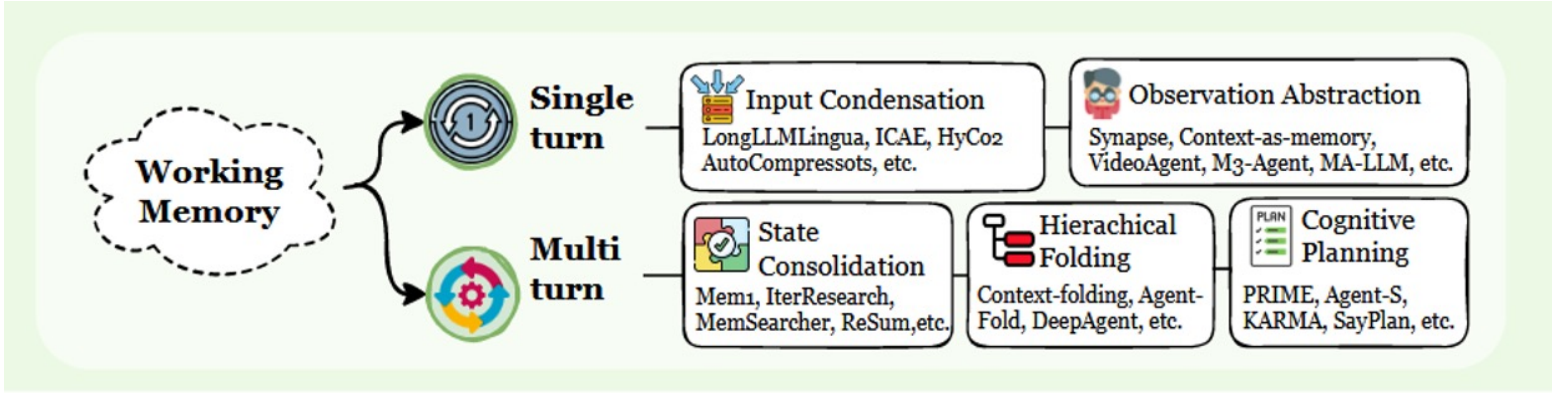
工作记忆

- **核心问题**：复杂任务需要几十步执行，但上下文窗口有限——中间状态、工具输出、推理轨迹同时竞争空间
  - **单轮工作记忆**：一次调用内，如何在有限 token 预算下保留最关键信息
  - **多轮工作记忆**：跨多步执行，如何管理不断累积的历史信息

特点：

轻量实时，任务结束后清空，

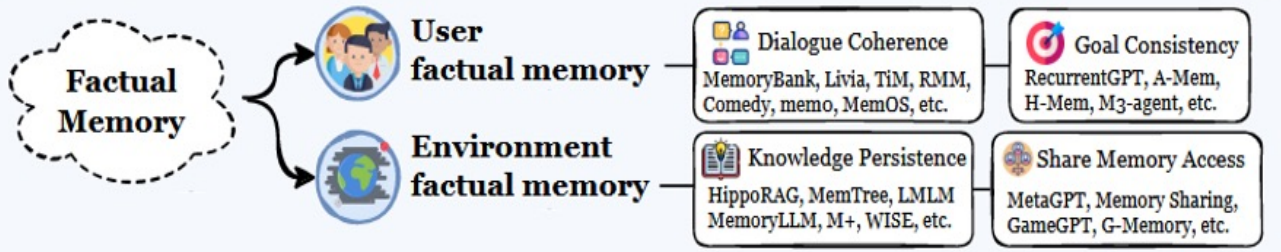
不污染长期记忆



事实记忆

- **核心问题**：工作记忆任务结束后清空，但关于用户和环境的稳定事实需要跨任务持久化
  - **User Factual（用户事实）**：维持人机交互的一致性
  - **Environment Factual（环境事实）**：维持与外部世界的一致性

事实记忆保障智能体交互中的三个基本属性：连贯性（上下文衔接）、一致性（跨时间不自相矛盾）、适应性（根据用户特征个性化调整）

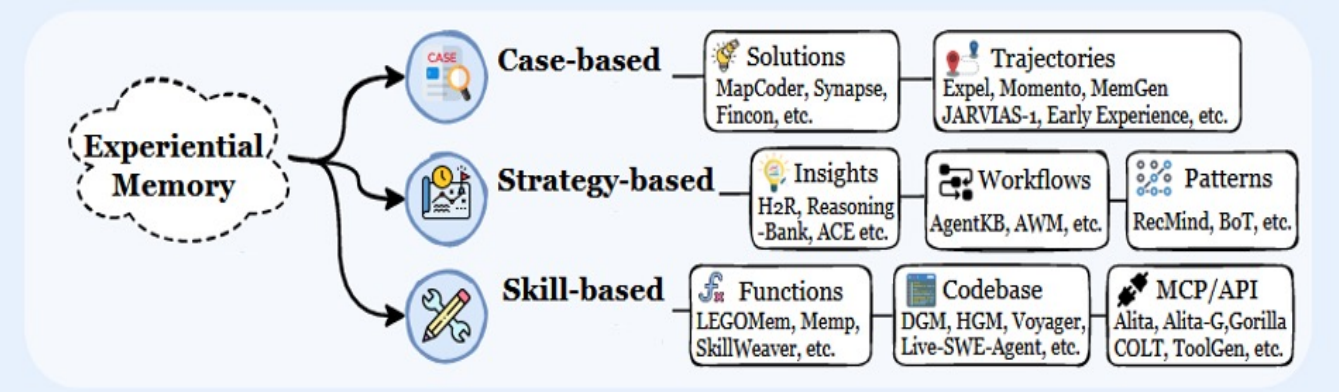


经验记忆

❑ **核心问题：** Agent 还需要从历史任务执行中学会"怎么做得更好"

- **Case-based（案例型）**：存储具体任务轨迹，保留情境与解法的原始对应关系
- **Strategy-based（策略型）**：从案例中蒸馏出可迁移的高层知识语义表示
- **Skill-based（技能型）**：将经验封装为可调用的可执行能力

三个抽象层次形成递进关系，实际系统越来越倾向混合使用：案例作为事实基底，策略提供规划逻辑，技能处理具体执行，三者协同配合。



记忆的存储形式

❑ 三种记忆存储形式：

- **Token-level 记忆**：显式可见的离散单元，可独立访问和修改。
- **参数化记忆（Parametric）**：存储在模型参数中，自动使用但难以直接读取
- **隐式记忆（Latent）**：存在于模型内部隐状态或连续表示中，可以在推理过程中或跨交互周期持续存在并更新

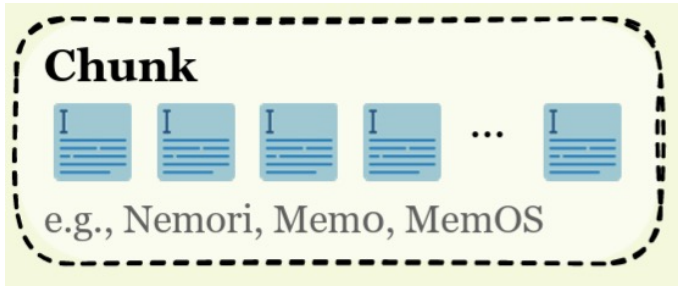
Token-level：像笔记本，可看可改。 **Parametric**：像肌肉记忆，会用但难说清楚。 **Latent**：像大脑神经激活，看不见但影响判断

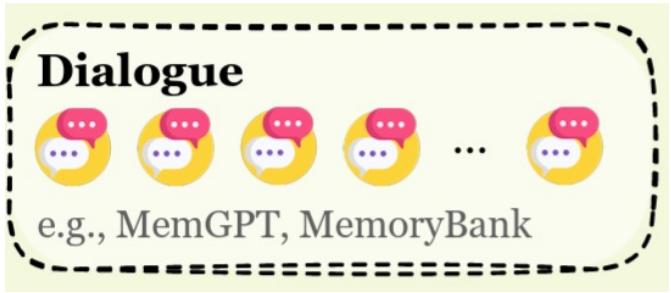
Token-level 记忆

❑ 定义

Token-level 记忆：

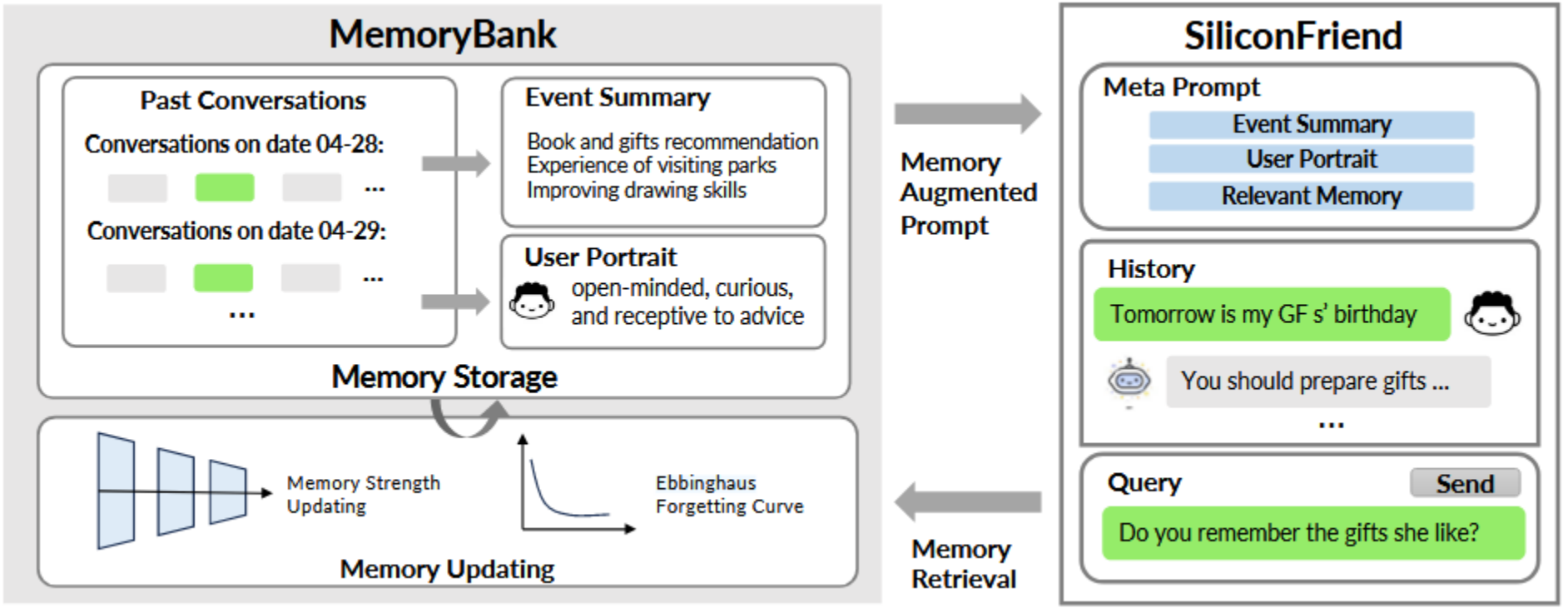
以显式离散单元组织的记忆，可被独立访问、修改和重构。这些单元对外部可见，可以结构化形式持久存储。





MemoryBank

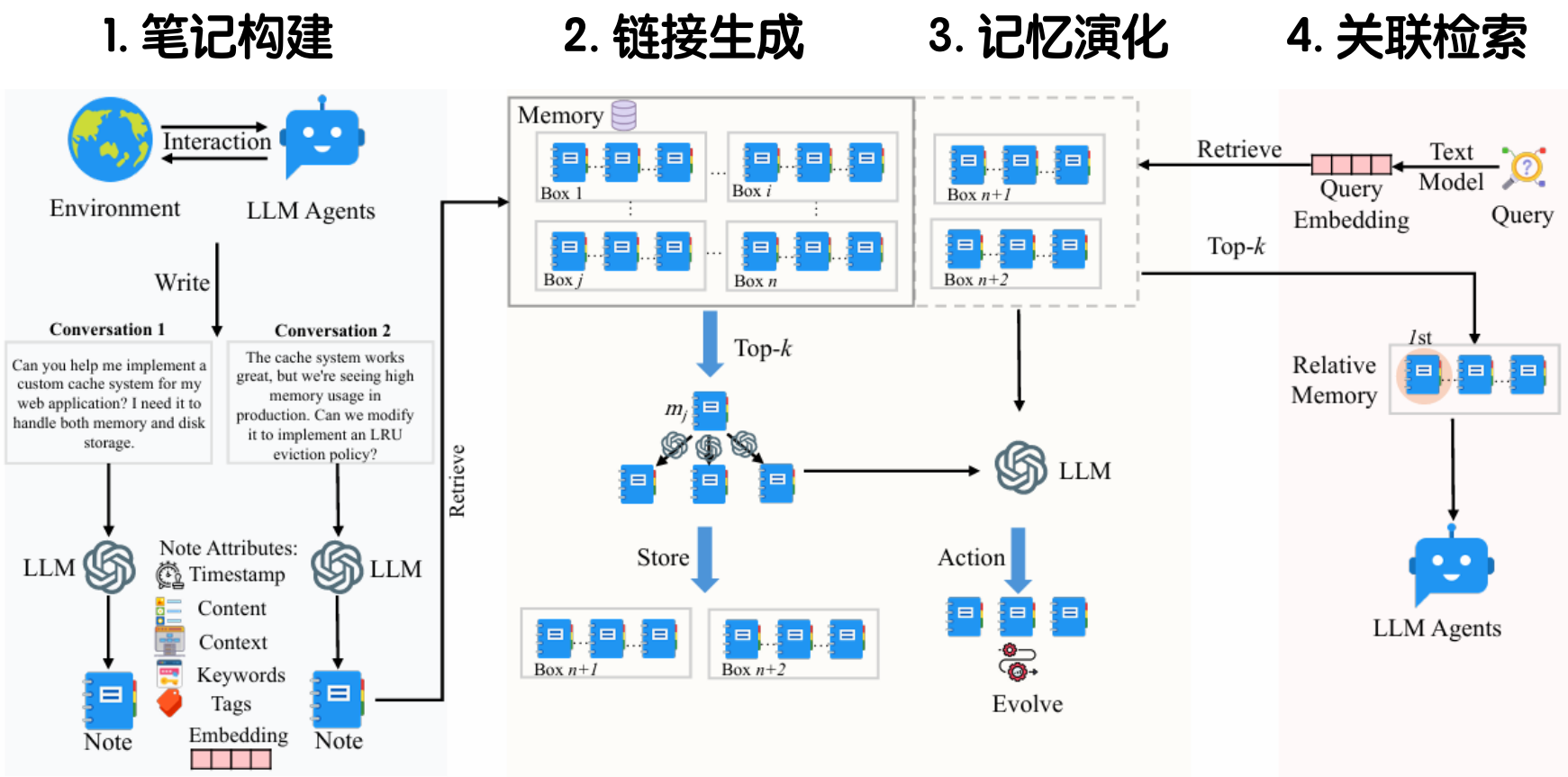
- 将长期对话转化为事件摘要和用户画像，并通过记忆更新与检索机制，为后续对话提供个性化上下文



MemoryBank: Enhancing Large Language Models with Long-Term Memory. AAAI 2024

A-Mem

- 借鉴**卡片盒笔记法**，构建一种结构化的记忆系统



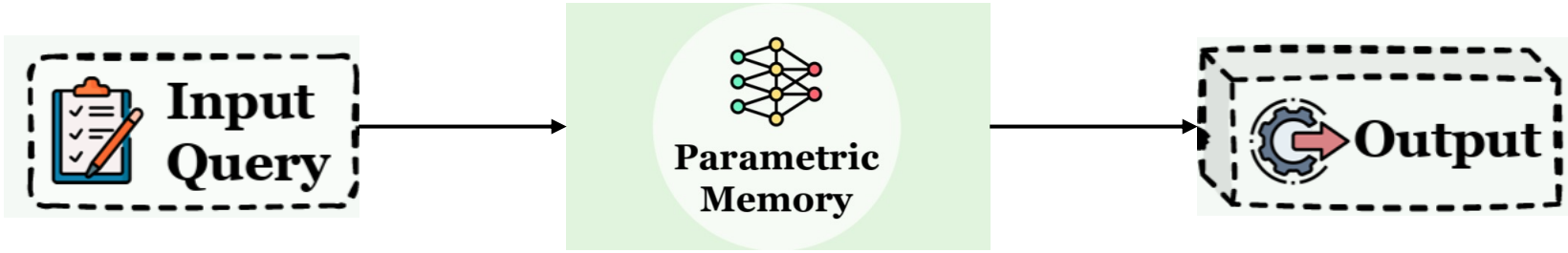
A-MEM: Agentic Memory for LLM Agents. NeurIPS 2025

参数化记忆

❑ 定义

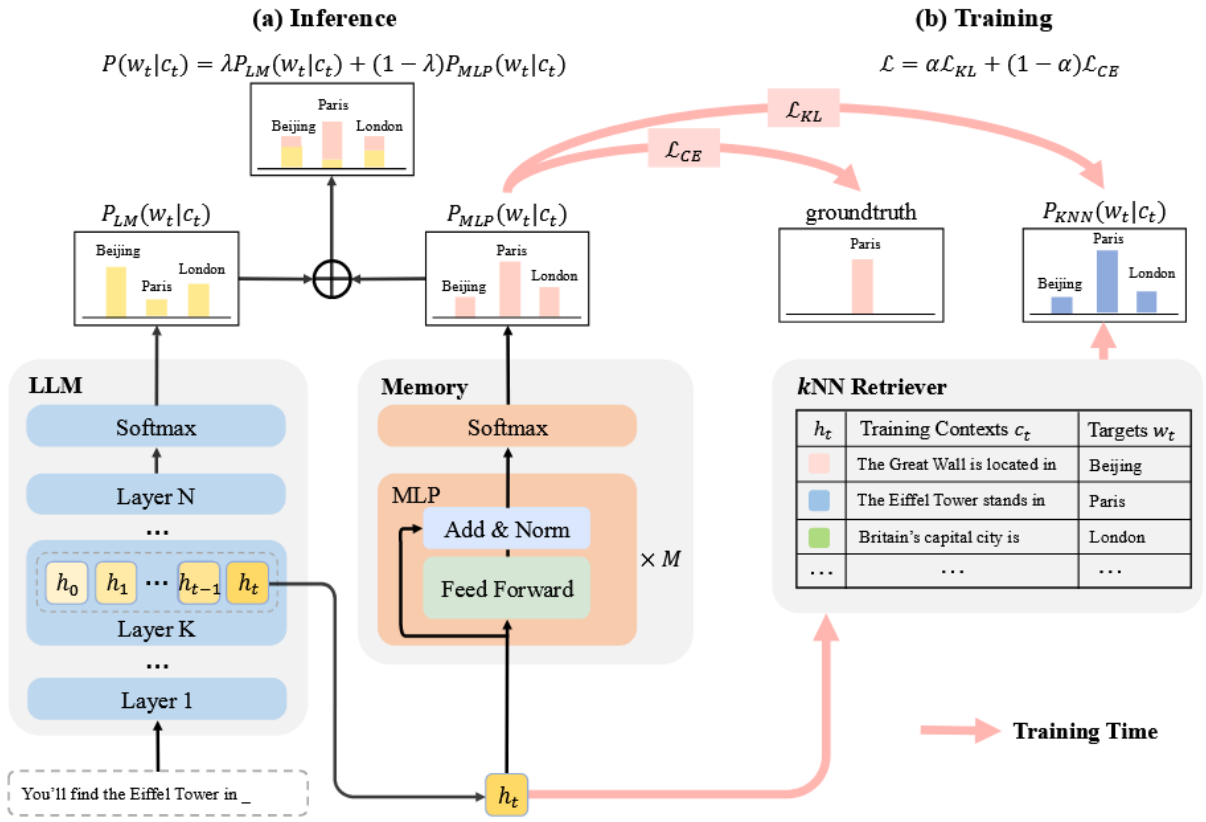
参数化记忆（Parametric）：

存储在模型参数中的记忆，信息通过参数空间的统计模式编码，在前向计算时被隐式访问。



MLP Memory

- 将记忆从 LLM 解码器中解耦出来，使用一个预训练的、可微分的外部记忆模块



- 推理阶段
  - LLM 输出分布 与 MLP Memory 输出分布通过概率插值融合
  - 无需访问任何外部文档库

- 训练阶段
  - MLP 通过 KL 散度模仿 kNN 检索器的输出分布
  - 同时用交叉熵损失对齐真实标签
  - LLM 主干全程冻结，仅训练 MLP模块

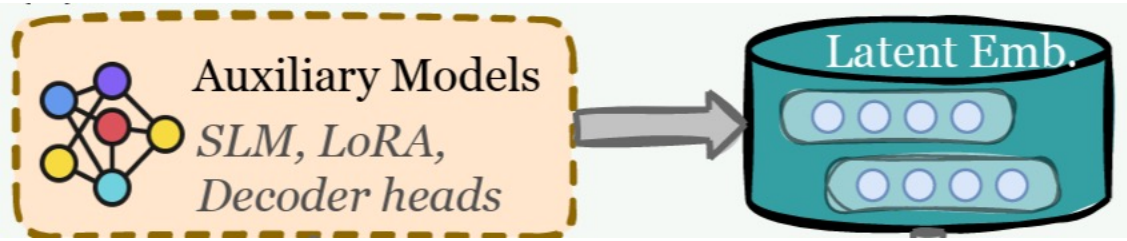
MLP Memory: A Retriever-Pretrained Memory for Large Language Models. ICLR 2026

隐式记忆

❑ 定义

隐式记忆（Latent）

隐含在模型内部表示中的记忆（如 KV cache、activations、hidden states、latent embeddings），而非以显式的、人类可读的文本的形式存储



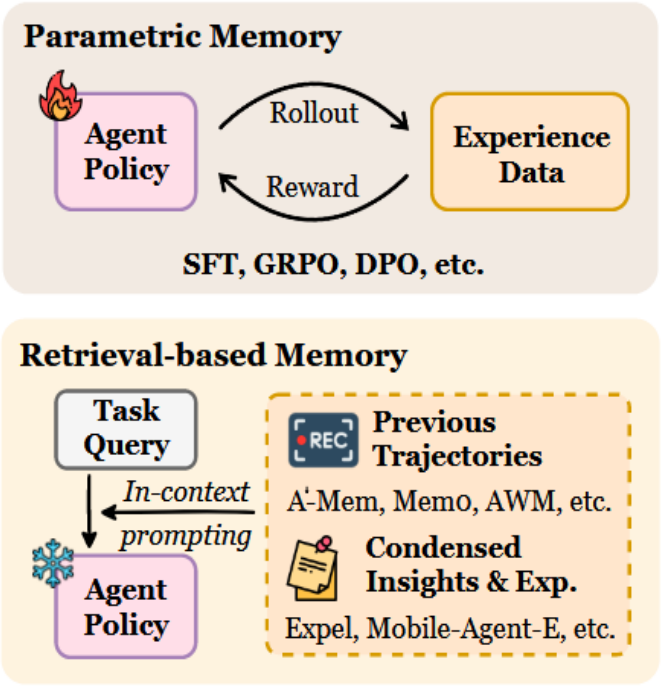
MemGen

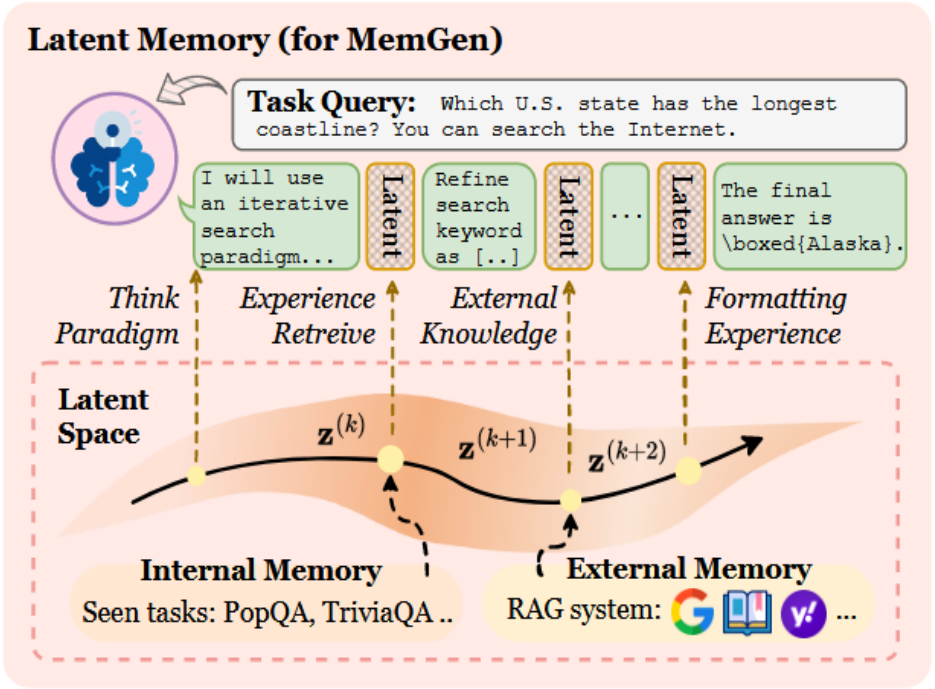
❑ 主要动机

参数化记忆：改参数内化经验，但导致灾难性遗忘

检索式记忆：存数据库避免遗忘，但检索-注入管道僵硬，与推理割裂

共同缺陷：人在思考时记忆自然涌入推理，而非停下来去"查"——现有方案都做不到





MemGen: Weaving Generative Latent Memory for Self-Evolving Agent. ICLR 2026

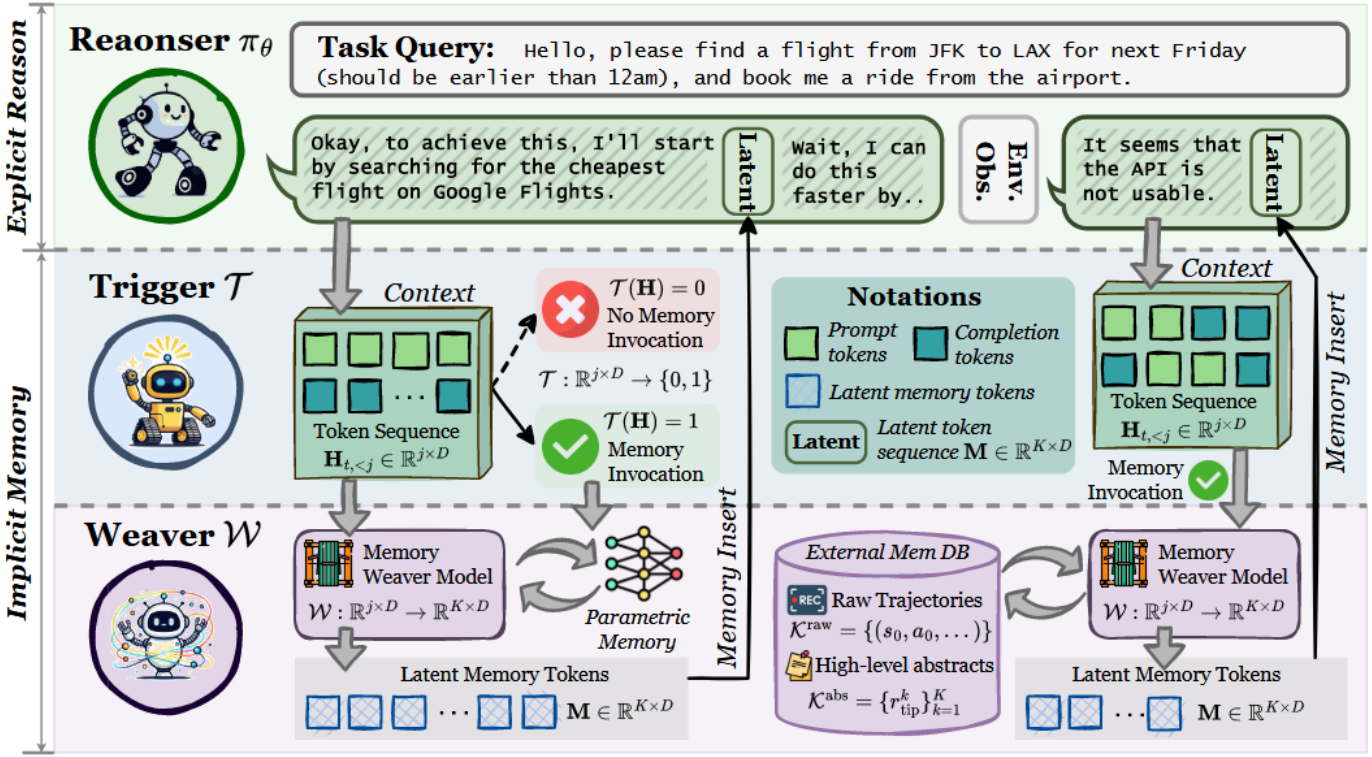
MemGen

- 记忆不是推理的前置输入，而是在推理过程中按需生成、即时编织的隐式信号

显式推理

记忆触发

记忆编织



MemGen: Weaving Generative Latent Memory for Self-Evolving Agent. ICLR 2026

存储形式总结

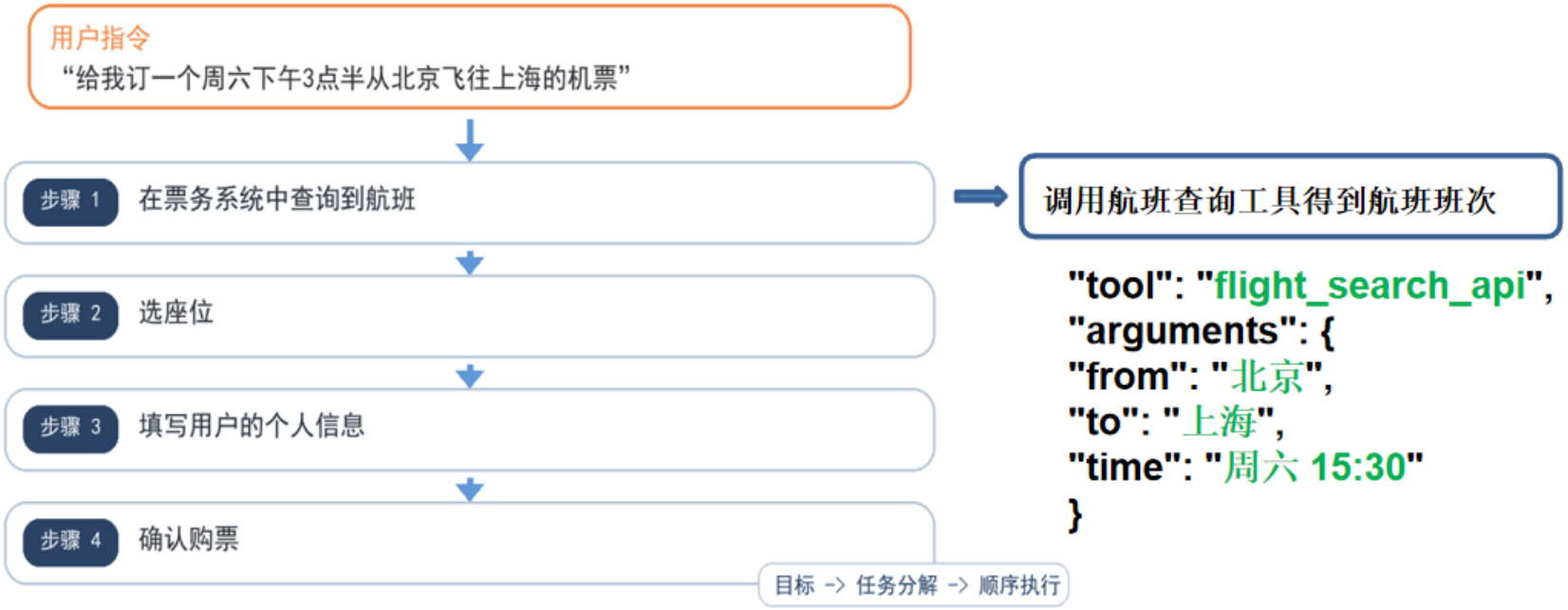
	Token-level 记忆	参数化记忆	隐式记忆
特征	可读、可编辑，支持即时增删改	隐式编码在权重中，抽象且可泛化	连续隐向量，人类不可读，机器原生
优势	透明可审计，决策可追溯；长期稳定不遗忘；即插即用不改模型参数	知识深度内化，泛化能力强；端到端推理流畅；性能增益通常最大	高表达容量，信息损失小；天然隐私保护；推理高效，便于多模态融合
缺点	检索质量制约效果；无法利用记忆间隐含关联；规模大时检索代价高	更新慢需重新训练；灾难性遗忘严重；记忆内容不可解释	不可读不可调试；记忆内容无法人工干预；可解释性最低
适应场景	多轮对话、个性化、推荐系统、法律/金融/医疗等高风险领域	角色扮演、数学/编程等推理密集型任务、需要从根本上获得新能力的任务	多模态 Agent、端侧/边缘部署、隐私敏感场景

中国科学院大学
University of Chinese Academy of Sciences

目 录

工具调用模块

- 工具调用模块是智能体的核心执行中枢，负责连接大模型与外部真实世界



什么是工具?

- 任何通过外部手段增强 LLM 的方法都可被称为“工具”，有时也被称为函数
- 检索增强生成 (RAG) 本质上是工具学习的一个特例（搜索引擎作为工具）

搜索引擎

Google, Bing
实时信息检索

代码解释器

Python, Shell
程序执行与调试

数学工具

计算器, Wolfram
精确数值计算

数据库 / KG

SQL, 知识图谱
结构化数据查询

多模态工具

图像生成/语音
识别

Web API

天气, 地图, 邮件
在线服务调用

为什么需要工具?

❑ LLM 的固有局限

幻觉 (Hallucination)
生成看似合理但事实错误的内容，参数化知识有限且无法更新

实时性缺失
训练数据存在截止日期，无法获取当前信息（天气、股价等）

精确计算不足
复杂数学运算、代码执行、专业领域推理能力受限

❑ 工具增强的六大收益

知识获取
搜索引擎、数据库

专业增强
数学工具、代码解释器

自动化效率
邮件、购物

交互增强
多模态、翻译

可解释性
过程透明

鲁棒性
降低输入敏感度

如何使用工具?

- 大模型本身不会真的执行工具，它只给出使用工具的决策

模型看到的

```
{
  "type": "function",
  "name": "get_weather",
  "description": "Retrieves current weather for the given location.",
  "parameters": {
    "type": "object",
    "properties": {
      "location": {
        "type": "string",
        "description": "City and country e.g. Bogotá, Colombia"
      },
      "units": {
        "type": "string",
        "enum": ["celsius", "fahrenheit"],
        "description": "Units the temperature will be returned in."
      }
    },
    "required": ["location", "units"],
    "additionalProperties": false
  },
  "strict": true
}
```

模型返回的

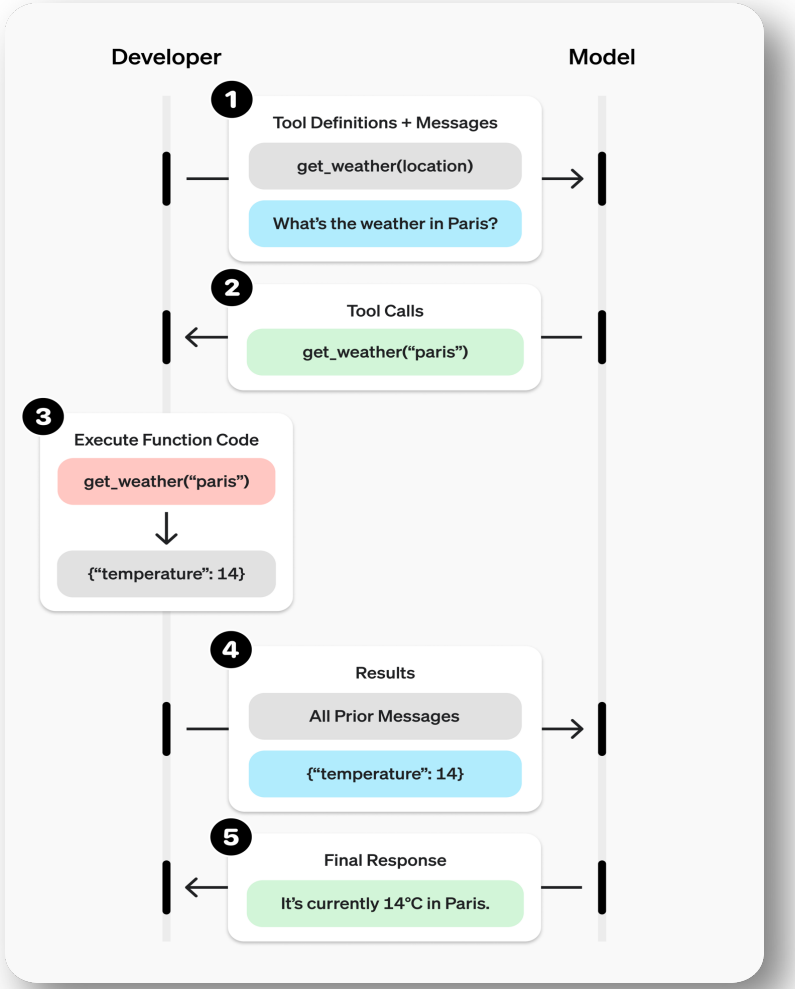
```
{
    "id": "fc_12345xyz",
    "call_id": "call_12345xyz",
    "type": "function_call",
    "name": "get_weather",
    "arguments": "{\"location\":\"Paris, France\"}"
},
```

模型给出使用工具的意图，以及使用的方式（name, arguments）

如何使用工具?

- 从开发者角度看，工具调用流程主要包含五个步骤：

1. **开发者**向**模型**发出请求，附带可以调用的工具描述
2. **开发者**收到**模型**发送的工具调用请求
3. **开发者**使用来自工具调用的输入，在应用程序端执行代码
4. **开发者**将工具执行结果发送给**模型**
5. **开发者**接收**模型**的最终响应（**或下一次工具调用请求**）



如何判断工具使用的好坏?

对最终任务完成度进行评估

通过在真实环境，或模拟的沙箱环境中实际执行工具，关注模型的最终响应，即任务的最终结果。

- **Pass Rate**：限定步数内任务成功完成的比例
- **Win Rate**：与基线方法的解法两两对比胜率
- **BLEU / ROUGE-L**：对最终响应的传统 NLP 评估指标

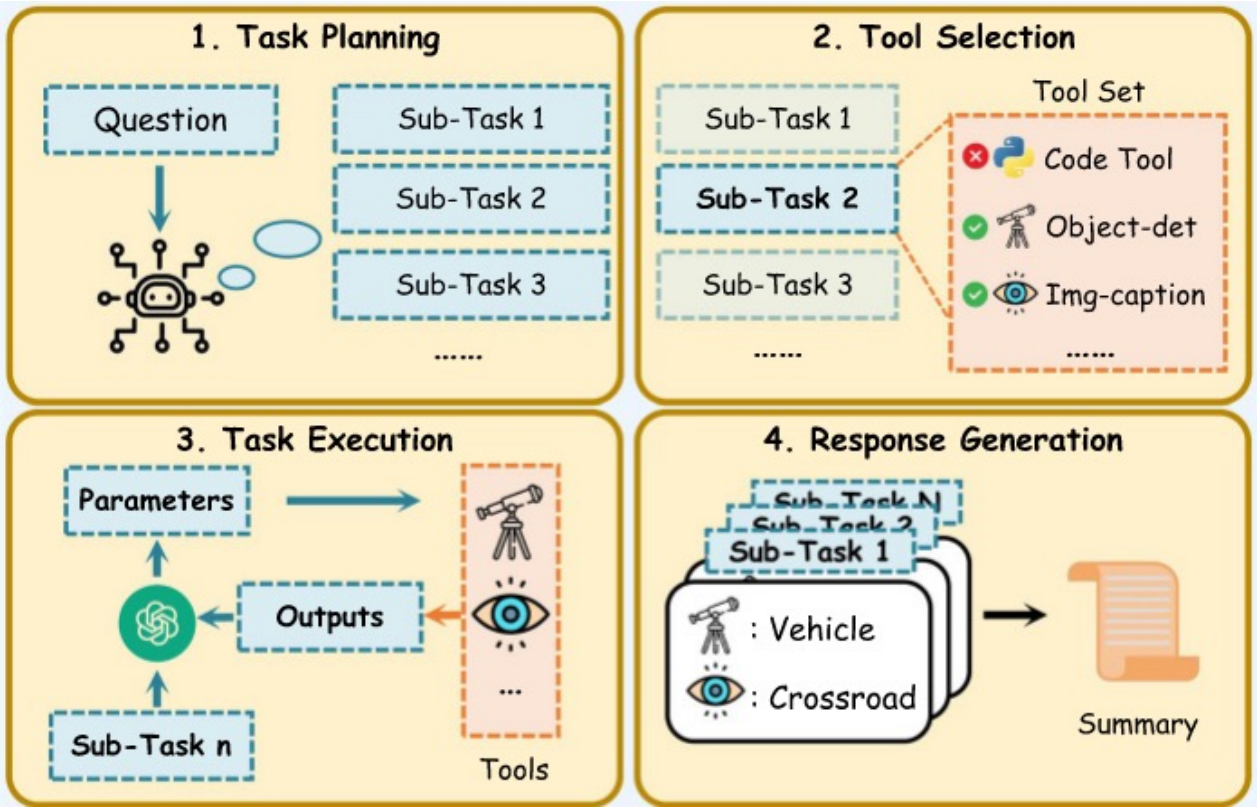
对工具使用过程评估

具体关注工具使用过程中各个阶段表现。

- **ROUGE-L（任务规划）**：预测计划与参考计划的最长公共子序列
- **Tool Selection Accuracy（工具选择）**：工具选择准确率
- **Recall@K / NDCG@K（工具选择）**：top-K 工具是否包含 / 多大程度匹配 ground truth 工具
- **AST 子树匹配（工具选择&调用）**：把 API 调用解析成抽象语法树，逐参数和ground truth匹配
- **Parameter Error Rate（工具调用）**：参数填充错误率
- **Levenshtein Distance（工具调用）**：参数值的字符级编辑距离

核心工作流

❑ 四阶段流水线（模型角度）



1. **任务规划：** 理解意图，将复杂问题分解为子任务
2. **工具选择：** 从庞大的工具库中匹配合适的工具
3. **工具调用：** 提取参数并生成符合规范的 API 调用请求
4. **响应生成：** 综合工具返回的结果与内部知识，生成最终回答

核心阶段一：工具选择

- 为了解答这些问题，模型从众多候选工具中选择合适的工具

两种工具选择方式：

检索式方法

用检索器从大规模工具库中快速筛出 Top-K

适用于工具数量庞大的场景

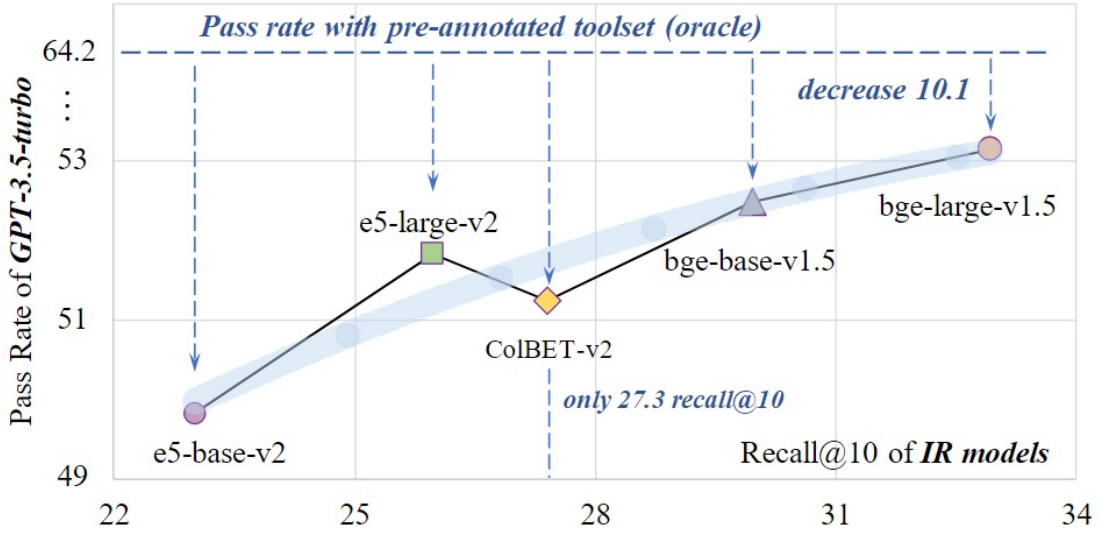
生成式方法

LLM 从候选工具列表中直接生成工具

用于工具数量有限的场景，或用于工具检索后的二次精选

检索式方法： TOOLRET

- 从传统信息检索到工具检索面临**领域迁移问题**



TOOLRET中query和目标工具之间的ROUGE-L词汇重叠只有0.06，而传统IR数据集高得多

	Ours	NQ	MSMARCO	HotpotQA	MTEB
# Average number of targets for an input query.	**2.17**	1.00	1.00	2.00	2.57
# ROUGE-L overlap between query and targets.	**0.06**	0.31	0.34	0.11	0.27

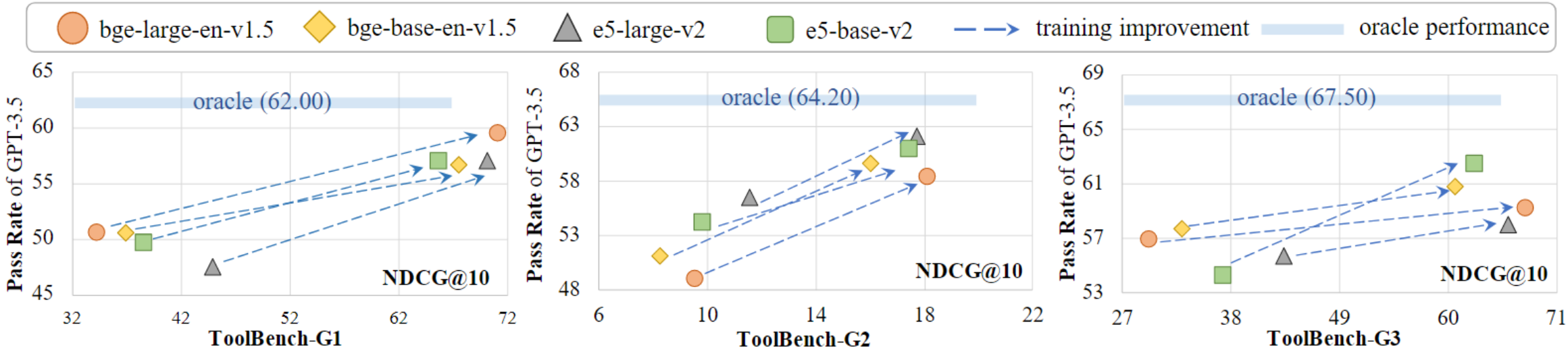
工具检索benchmark TOOLRET

工具检索是一个独立的、需要专门优化的研究问题，不能直接套用通用检索器

Retrieval models aren't tool-savvy: Benchmarking tool retrieval for large language models. ACL 2025

检索式方法： TOOLRET

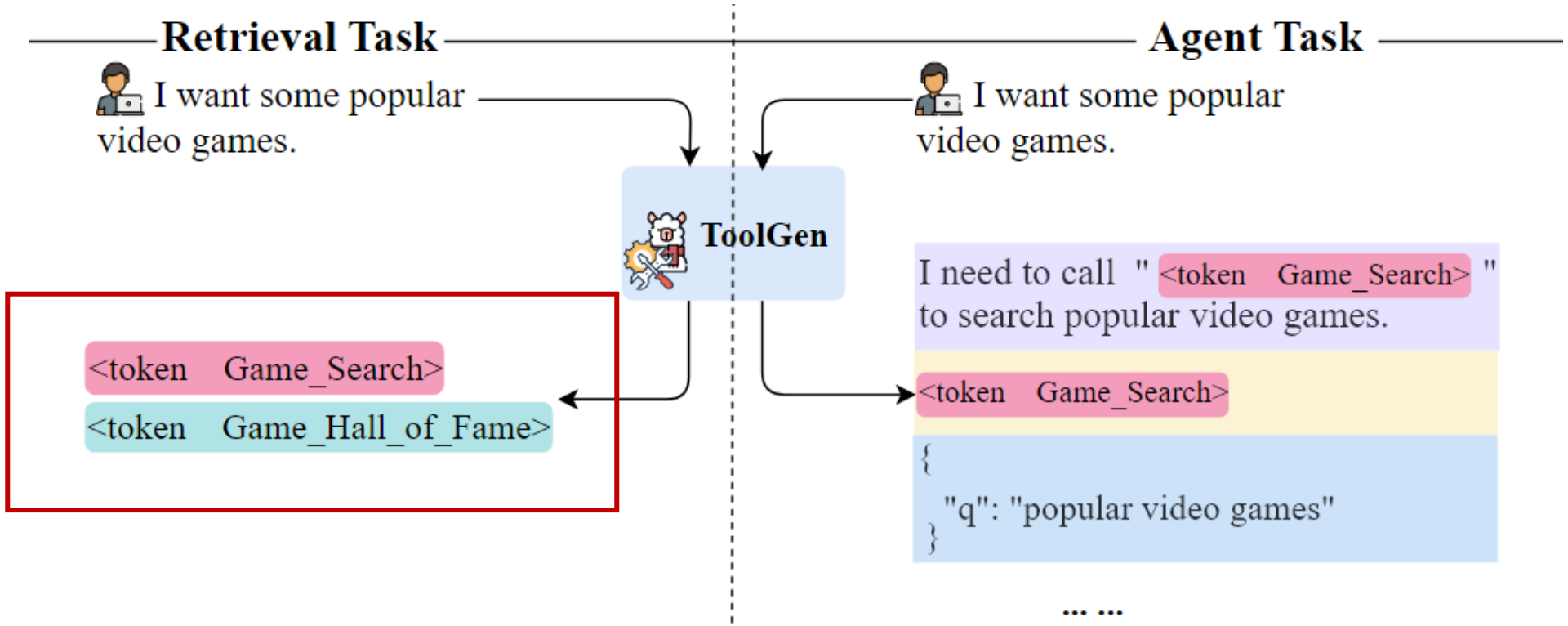
- 针对工具检索任务，TooLRET**专门训练 信息检索 IR 模型**
- 收集超过20万个训练实例，每个实例均由一个查询q和一组目标工具$\mathcal{T}$组成
- 在训练时也加入检索器原本检索到的错误工具作为负样本，以更好的区分正确和错误的选择



Retrieval models aren't tool-savvy: Benchmarking tool retrieval for large language models. ACL 2025

生成式方法： ToolGen

- 检索变为生成，将每个工具作为特殊 token 加到 LLM 的词表中



ToolGen: Unified Tool Retrieval and Calling via Generation. ICLR 2025

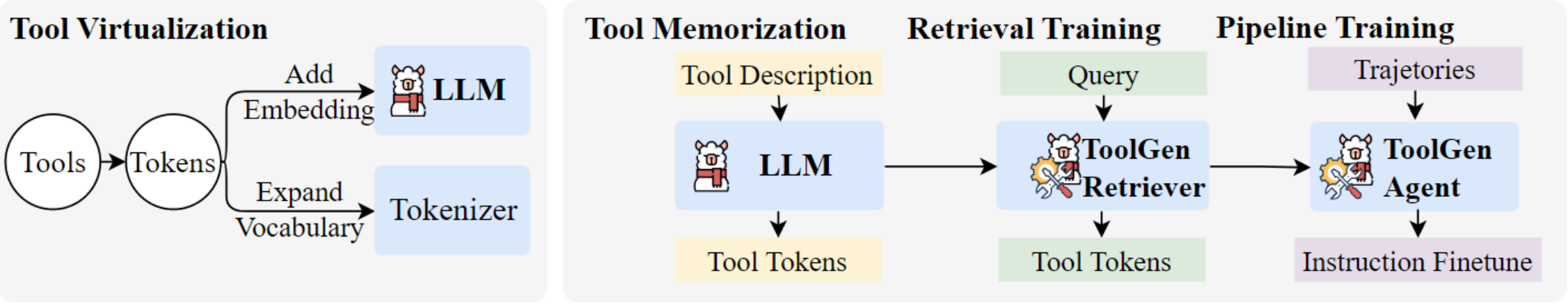
生成式方法： ToolGen

- 三个阶段递进的训练:

① 工具记忆
输入：工具文档/描述
输出：对应的工具 token
目标：让模型把 token 和工具的功能绑定起来

② 检索训练
输入：用户 query
输出：相关的工具 token
目标：学会"看到 query 就生成对的工具 token"

③ Pipline训练
输入：用户query
输出：完整工具使用轨迹
目标：学会端到端完成任务



ToolGen: Unified Tool Retrieval and Calling via Generation. ICLR 2025

核心阶段二：工具调用

- 工具调用的本质：生成结构化文本:

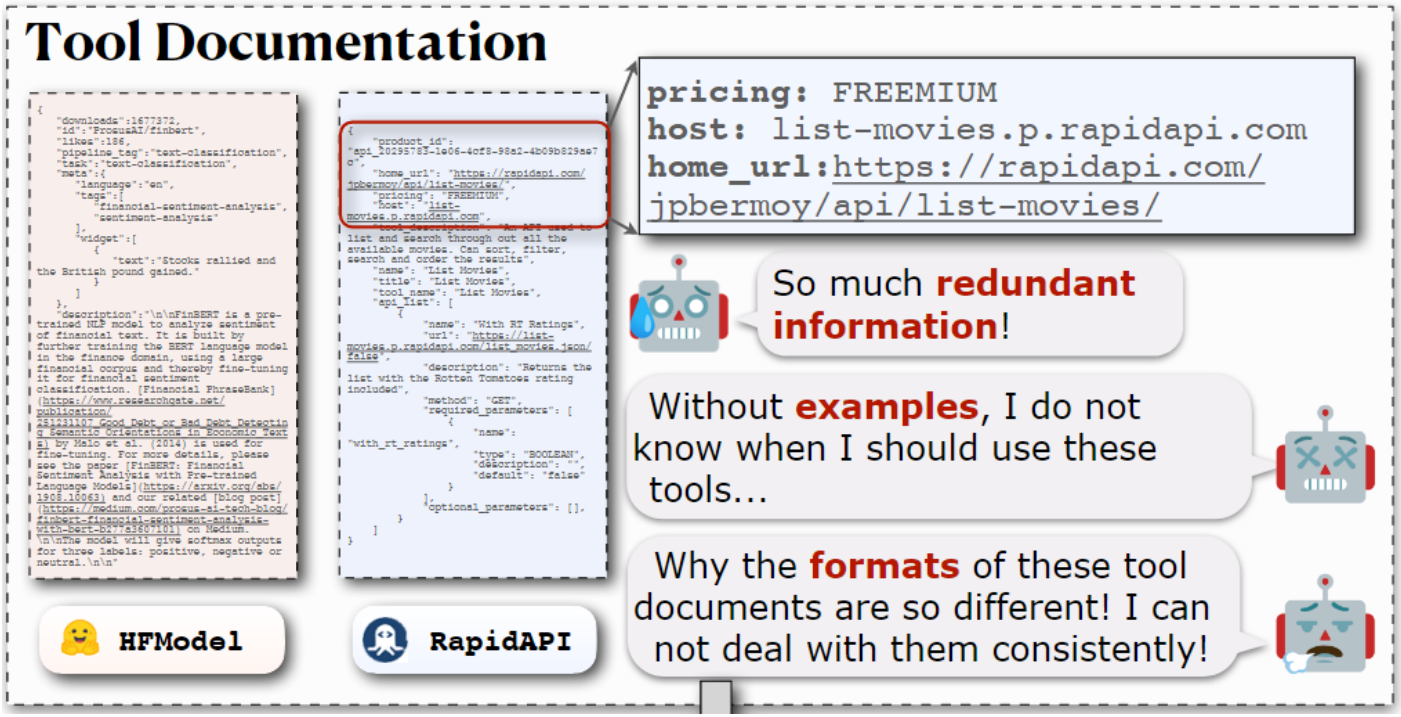
> LLM 需要根据工具描述从用户查询中提取所需参数，并**严格按照规定的输出格式**生成调用请求，不能输出多余文字。

> 更清晰准确的工具描述

注意：清晰的工具描述是工具调用阶段的重要基础，但其他阶段也同样受益（如任务规划，工具选择）

EASYTOOOL

- 现实世界**工具描述文档**存在以下三个问题:

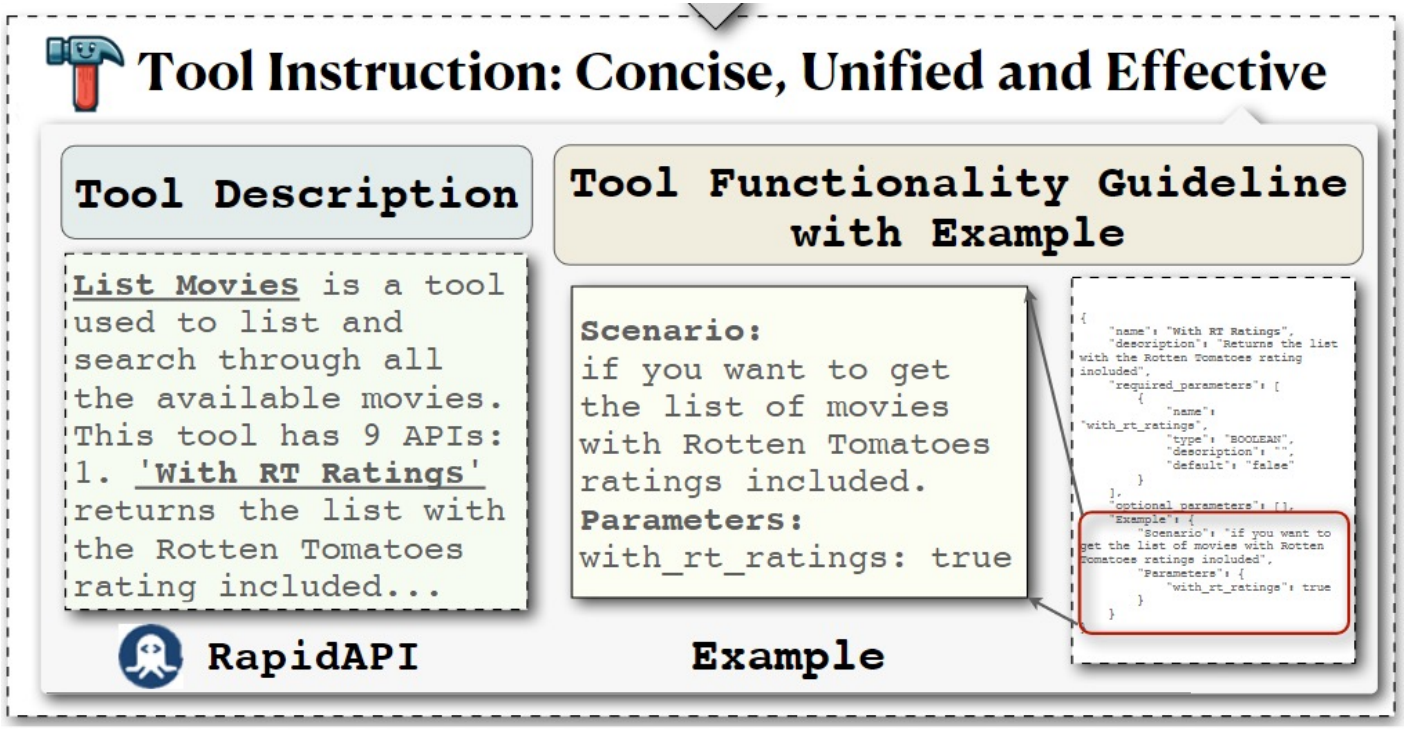


- **冗余性**：文档里包含大量无关信息（URL、ID、定价等）
- **不完整性**：许多文档缺乏使用场景和示例，LLM 不知道何时调用、如何传参，导致参数错误和执行失败
- **不一致性**：来自不同来源的工具文档格式差异巨大，LLM 难以以统一方式理解

EASYTOOL: Enhancing LLM-based Agents with Concise Tool Instruction. NAACL 2025

EASYTOOOL

- 将工具文档转化为统一、简洁的“工具指令”



➢ **阶段一：工具描述生成**

使用 ChatGPT，通过一个精心设计的 prompt，将原始文档转换成简洁的**功能描述**，仅保留核心功能

➢ **阶段二：工具功能指引构建**

还需要解决参数错误问题。让 ChatGPT 从文档中抽取参数，并为每个功能生成包含使用**场景和参数的结构化示例**

EASYTOOL: Enhancing LLM-based Agents with Concise Tool Instruction. NAACL 2025

DRAFT

- 现有工具文档基本都是为人类开发者编写的，未针对 LLM 的理解特性优化，存在表达方式不匹配的问题

Tool Name: GET_person_person_id
Description: …Added the [translations] (#endpoint:CSaMjCxXAtGpxNGfS) …

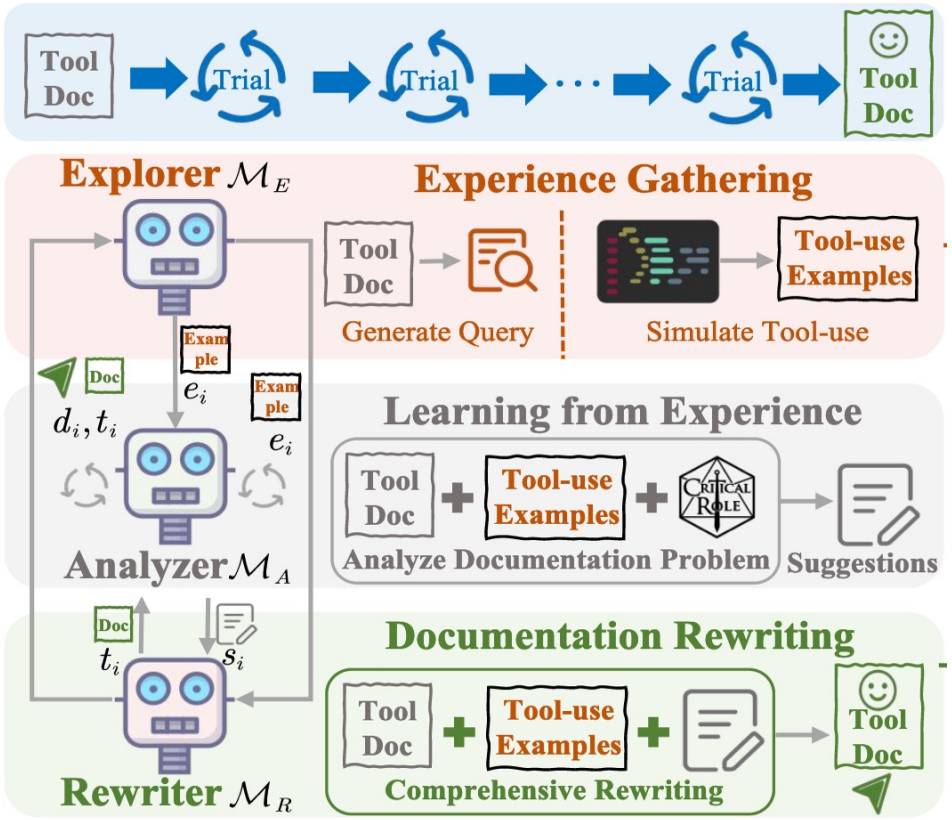
Query: Can you give me details about the person with ID 12345?
Tool Parameters: {"ID":"12345"}
Observation: { "biography": "", "birthday": "1919-11-29", "deathday": "1996-01-01", "gender": 2, "homepage": null …}

Rewriting: This API retrieves detailed information about a person using their unique ID. The response includes … The API supports the 'appe nd_to_response' parameter… Example usage: …
Direction for Next Exploration: …

From Exploration to Mastery: Enabling LLMs to Master Tools via Self-Driven Interactions. ICLR 2025

DRAFT

- 一个基于试错、迭代优化工具文档的自驱动框架



Step 1 经验生成：模拟潜在的工具使用场景，构建探索性实例，并记录工具执行结果

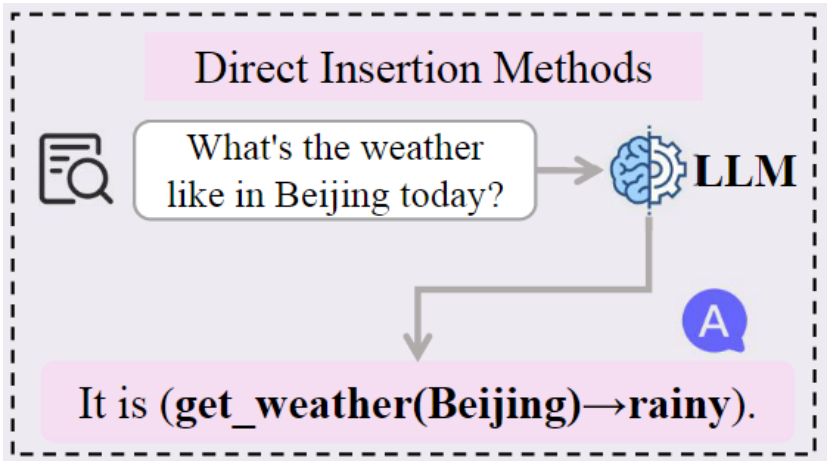
Step 2 从经验中学习：识别对当前文档进行分析，结合 explorer 所获得的结果与反馈，提出文档修改建议

Step 3:文档重写：综合这些信息，对工具文档进行优化，同时进一步指导 explorer 的后续探索过程。

From Exploration to Mastery: Enabling LLMs to Master Tools via Self-Driven Interactions. ICLR 2025

核心阶段三：响应生成

- **直接插入方法**：早期方法直接将工具输出插入到 LLM 生成的文本中，仅适合输出格式简单、固定的工具



实现简单直接，延迟低。

工具输出不可预测，可能影响用户体验。
且复杂多模态输出难以直接嵌入

核心阶段三：响应生成

- **信息整合方法**：将工具输出作为上下文输入 LLM，由模型整合工具信息与自身知识，生成更连贯、更完整的响应

工具输出可能很长，如何适配 LLM 有限的上下文长度？

Schema 简化：RestGPT 利用预创建的 schema 简化冗长的 API 返回结果，只保留关键字段信息

截断策略：ToolLLaMA 进一步采取截断策略，只保留前1024 tokens，简单但可能丢失关键信息

训练压缩器：ReCOMP 针对RAG（检索增强生成），训练模型将长文本浓缩为简洁摘要，训练目标直接对齐下游 LLM 的答题效果

开放挑战

- 工具学习的发展还面临一系列更深层的挑战：

1. 工具不可解任务

存在一些工具不可解的任务，比如提供的**工具和任务均不相关、或用户请求中缺少调用工具的必要参数值信息**，LLM需要明确指出无法调用工具或与用户进一步交流。一种解决方案是在LLM预定义的动作空间中加入“缺少工具”和“与用户交流”两种额外的特殊动作。

2. 工具过度使用问题

LLM 在面对工具时**常常对本可直接作答的简单任务也不必要地调用工具**，反而工具选择或参数错误引入额外噪声，并增加成本与延迟。WTU-Eval 等基准的评测结果表明，当前主流 LLM 普遍难以识别自身能力边界，无法准确判断何时该用、何时不该用工具。构造专门的微调数据增强模型的工具调用决策能力，是一种可行的解决思路。

Reducing Tool Hallucination via Reliability Alignment
WTU-EVAL: A Whether-or-Not Tool Usage Evaluation Benchmark for Large Language Models

开放挑战

3. 工具使用与通用能力的权衡

现有工作多通过**微调增强 LLM 的工具使用能力，造成灾难性遗忘**。一种解决思路是借助路由机制，对工具相关输入专门训练 LoRA 学习工具知识，以兼顾通用能力。

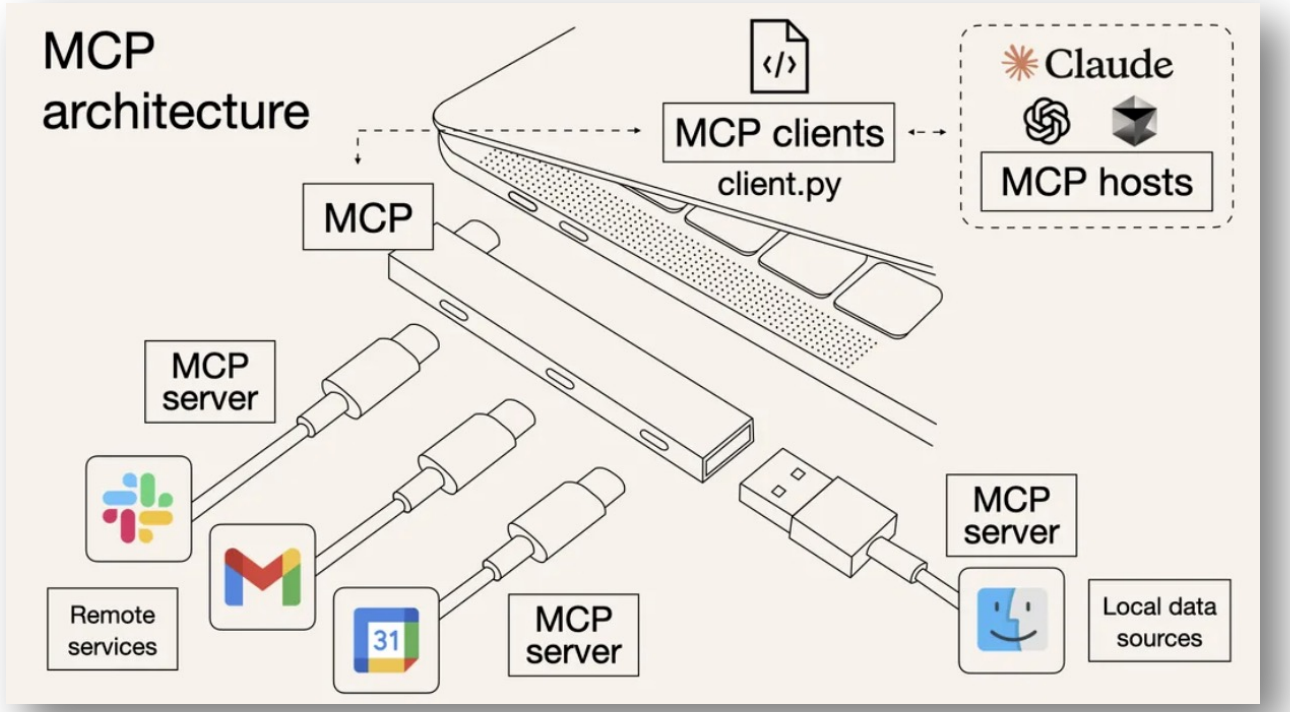
4. 工具使用引入的安全挑战

工具调用让 LLM 能够读取外部内容（如网页、邮件、API 响应），但这也打开了新的攻击面，比如，**攻击者可将恶意指令藏在外部内容中，劫持 agent 执行未授权操作或窃取隐私（即，间接提示注入）**。如何让 agent 区分用户的真实指令与外部内容中夹带的恶意指令，仍是一个开放问题。

CITI: Enhancing Tool Utilizing Ability in Large Language Models without Sacrificing General Performance
InjecAgent: Benchmarking Indirect Prompt Injections in Tool-Integrated Large Language Model Agents

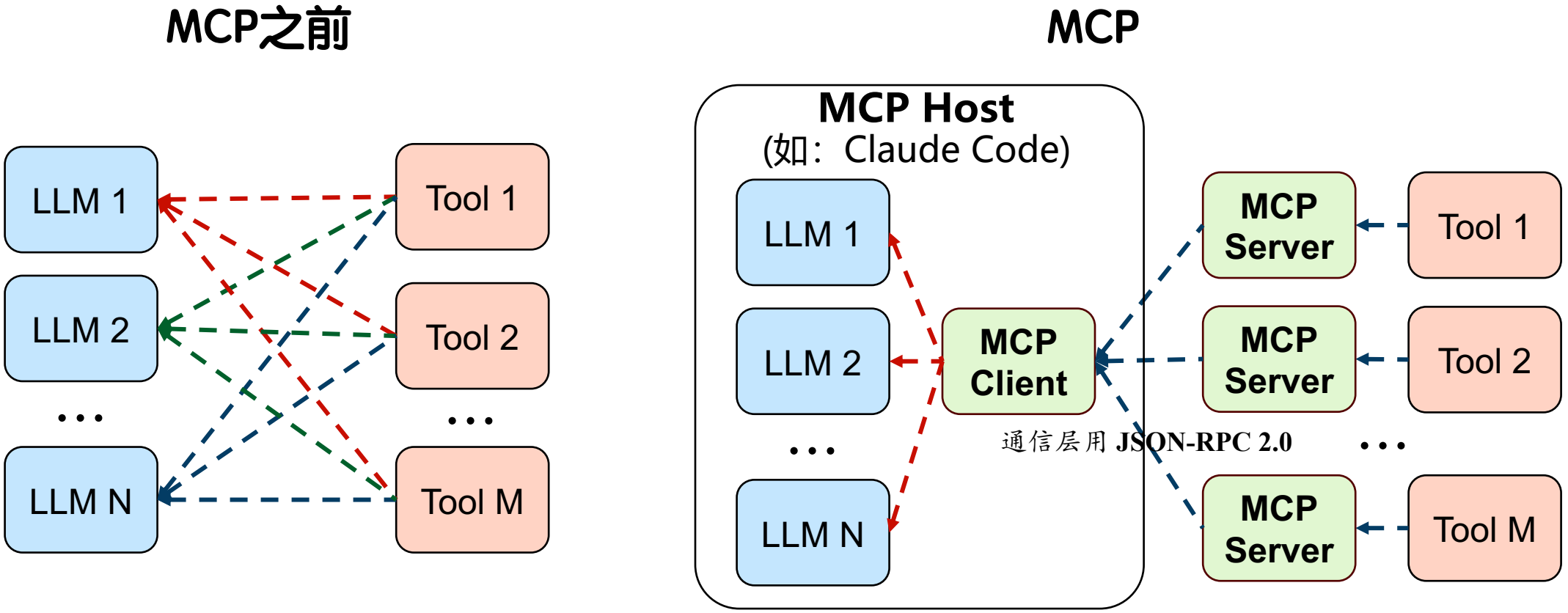
MCP

MCP（Model Context Protocol，模型上下文协议）是 Anthropic 于 2024 年 11 月推出的**开放标准**，用于标准化 LLM 应用与外部工具、数据源之间的连接方式



MCP

- MCP 的核心是提供了一个统一封装的接口



Skills

- Agent Skills 是工具之上的“技能手册”。它定义了特定任务的工作流，告诉智能体任务要按什么步骤做，以及要怎么做

> 本质上，Skill是一个包含 SKILL.md 文件的文件夹。该文件包含 metadata（包含 name, description 两个字段）以及指示智能体如何执行特定任务的 instructions。Skill 还可以捆绑脚本和其他必要资源文件。

```
my-skill/
├── SKILL.md          # Required: metadata + instructions
├── scripts/          # Optional: executable code
├── references/       # Optional: documentation
├── assets/           # Optional: templates, resources
└── ...               # Any additional files or directories
```

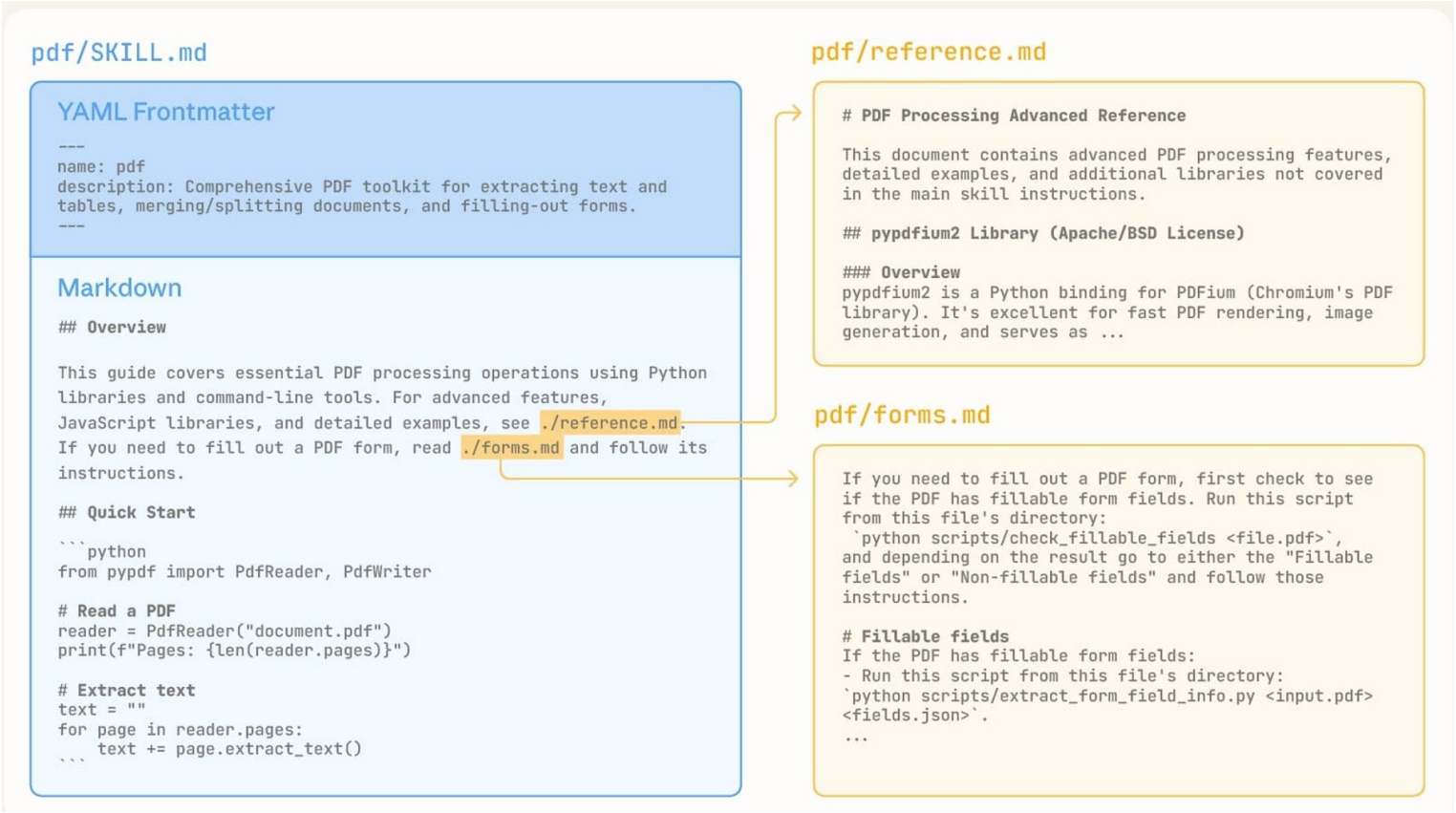
Skills

❑ 核心设计原则：渐进式披露 (Progressive Disclosure)

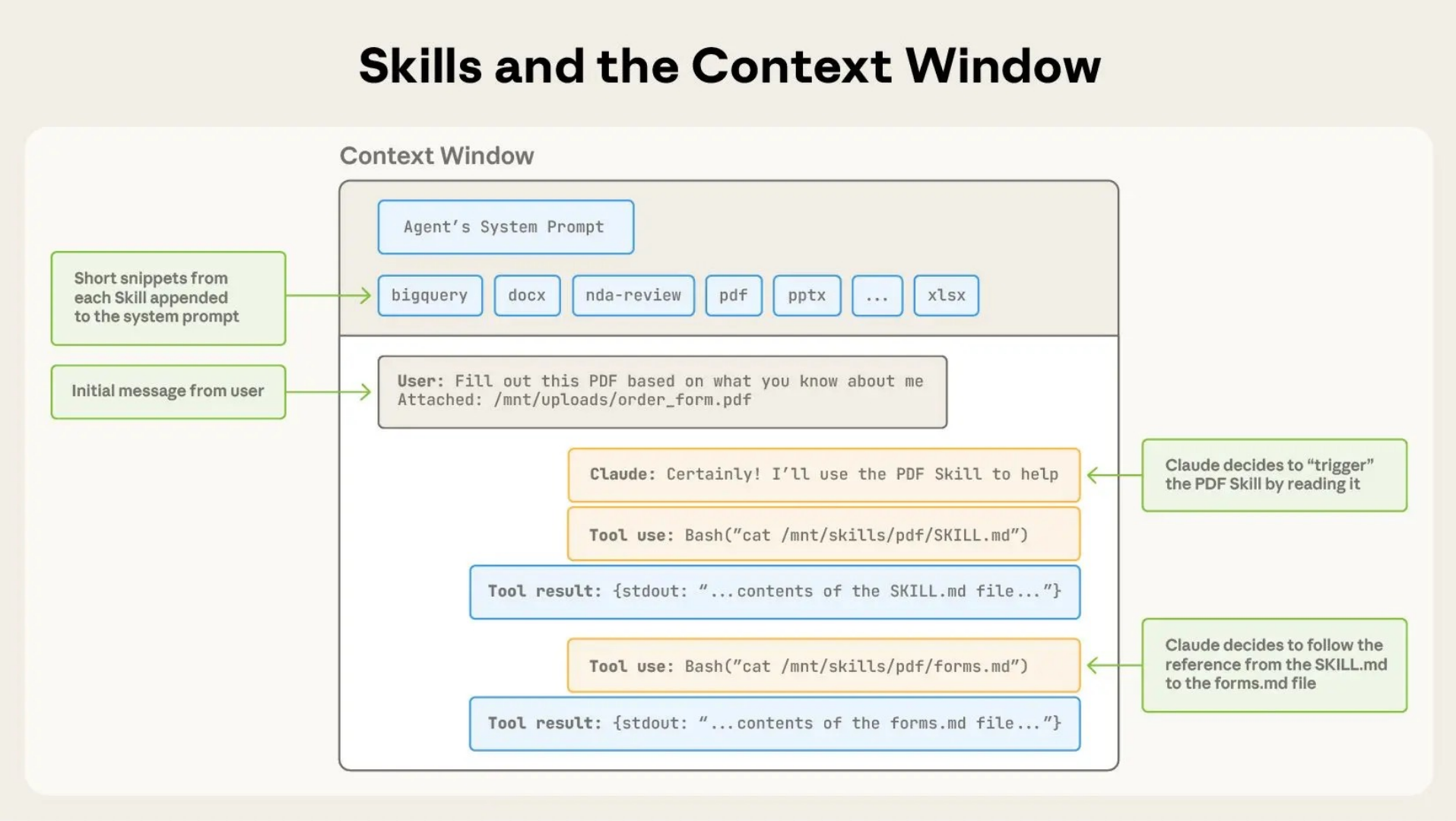
第一层： Agent 启动时，将 SKILL.md 的 name 和 description 加入系统提示，开销小

第二层： 用户提问时，Agent 根据任务匹配 Skill，并加载完整 SKILL.md 的 instructions

第三层（或更高）： 复杂或长的 instructions 可拆成附加文件，仅在需要时才被智能体读取



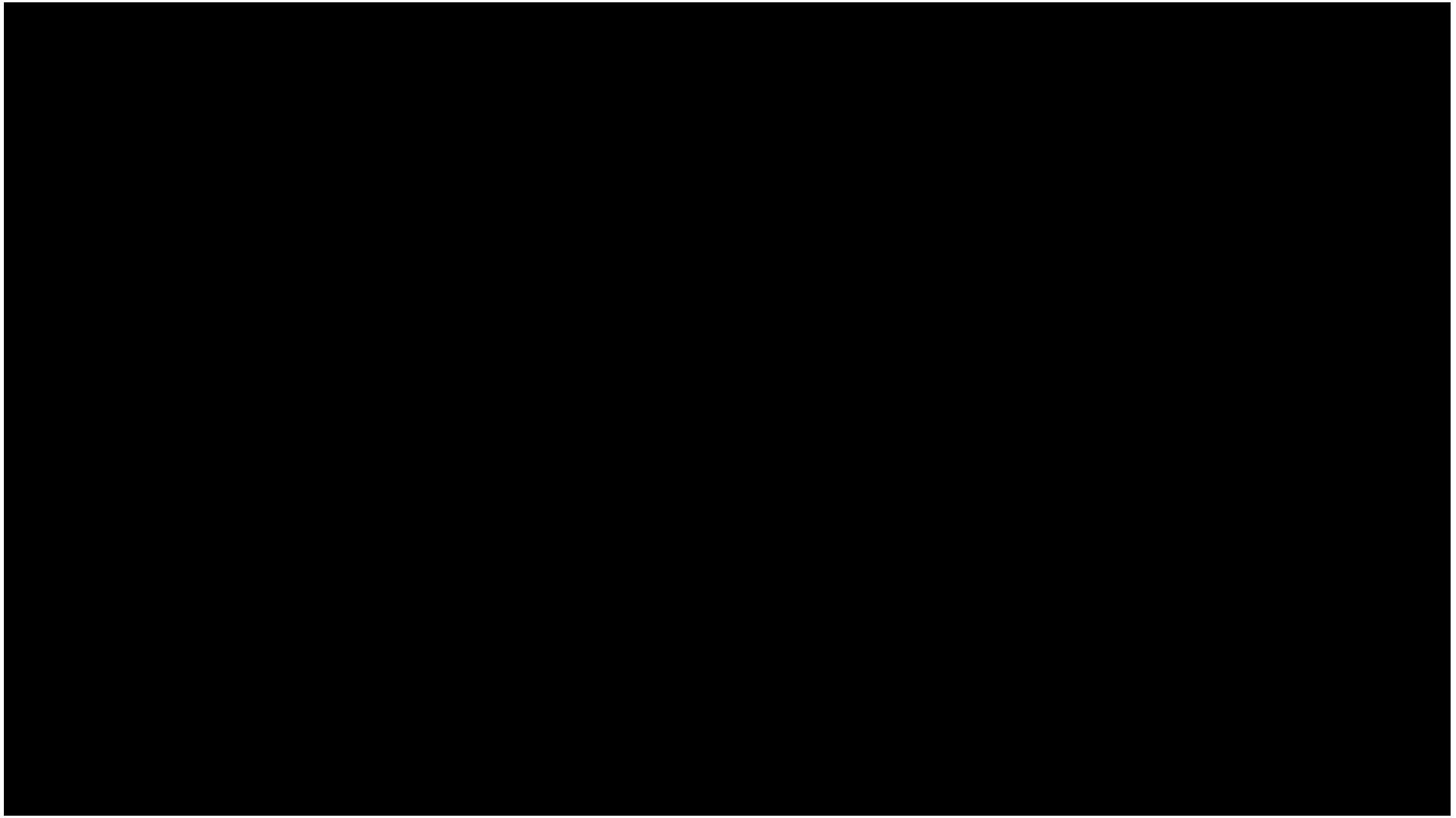
Skills


Skills and the Context Window
Context Window
Agent's System Prompt
bigquery
docx
nda-review
pdf
pptx
...
xlsx
Short snippets from each Skill appended to the system prompt
Initial message from user
User: Fill out this PDF based on what you know about me
Attached: /mnt/uploads/order_form.pdf
Claude: Certainly! I'll use the PDF Skill to help
Tool use: Bash("cat /mnt/skills/pdf/SKILL.md")
Tool result: {stdout: "...contents of the SKILL.md file..."}
Tool use: Bash("cat /mnt/skills/pdf/forms.md")
Tool result: {stdout: "...contents of the forms.md file..."}
Claude decides to "trigger" the PDF Skill by reading it
Claude decides to follow the reference from the SKILL.md to the forms.md file

本节复习

- 智能体及其四个核心模块
  - 感知模块：多模态输入
  - 规划模块：任务分解
  - 记忆模块：Token、 Parametric、Latent
  - 工具调用模块：工具选择、工具调用

黑客帝国三部曲（Matrix）



参考文献

- A Survey on Large Language Model based Autonomous Agents. 2023
- WebArena: A Realistic Web Environment for Building Autonomous Agents. 2023
- ReAct: Synergizing Reasoning and Acting in Language Models. ICLR 2023
- Memory in the Age of AI Agents: A Survey. 2025
- ToolGen: Unified Tool Retrieval and Calling via Generation. ICLR 2025
- MemGen: Weaving Generative Latent Memory for Self-Evolving Agent. ICLR 2026
- https://openai.github.io/openai-agents-python/agents/

致 谢

- 胡玥、曹亚男、方芳：国科大《自然语言处理基础》
- 曹亚男、任昱冰：国科大《深度学习与自然语言处理概述》


THANKS