

人工智能导论

Introduction to Artificial Intelligence

第七章 大模型推理与规划



填空题 4分

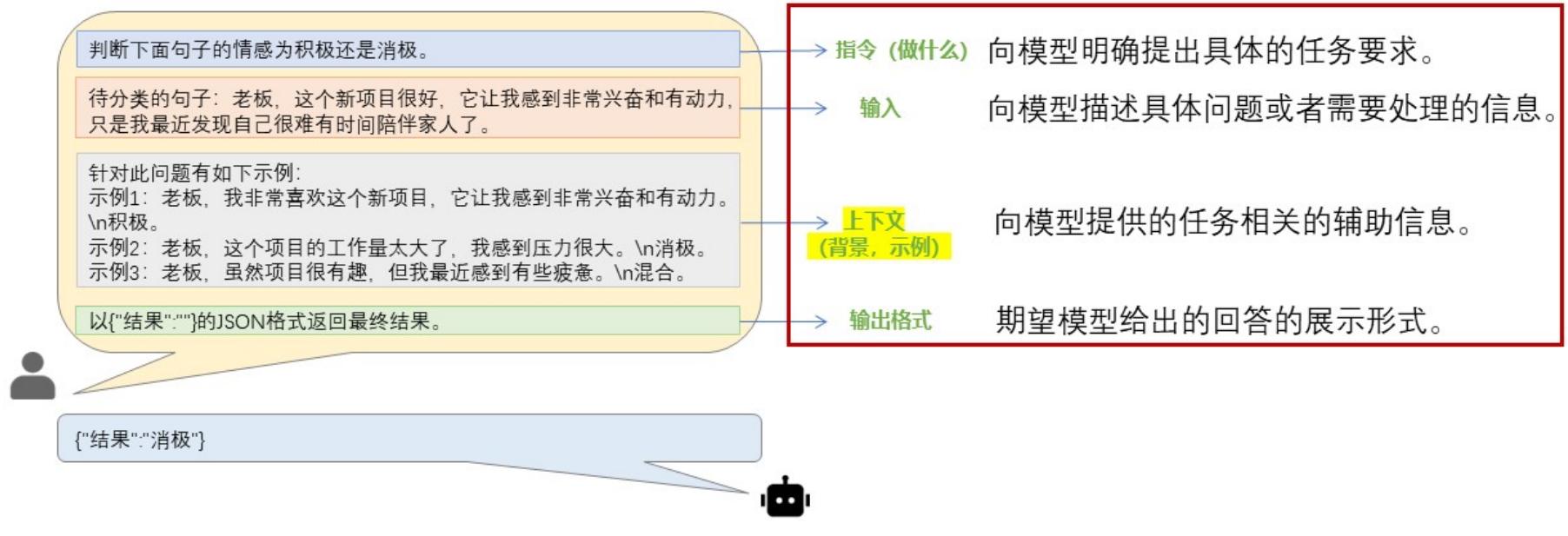
一个标准的提示词通常由四个部分构成。请填空：

- (1) : [填空1] (告诉模型要完成的任务或行为要求)
- (2) : [填空2] (提供模型需要处理的具体内容)
- (3) : [填空3] (为任务提供背景信息、限制条件或示例)
- (4) : [填空4] (要求模型用什么形式来回答)

Prompt工程

➤ 提示词要素

任务说明、问题、上下文和输出格式四个基本元素



单选题 1分

在提示工程中，通过在输入中加入必要的上下文信息，可以显著提升模型回答质量。下面哪一种方式不**属于**常见的上下文增强策略？

- A 在提示中提供任务示例 (In-Context Learning)
- B 使用系统提示 (System Prompt) 设定角色与风格
- C 在提示中加入无关细节增加 token 多样性
- D 提供任务相关的背景知识、约束条件或数据结构

ICL分类

按照**示例数量**的不同，上下文学习可以分为三类：**零样本（Zero-shot）** 上下文学习，**单样本（One-shot）** 上下文学习和**少样本（Few-shot）** 上下文学习。

###判断下面句子的情感为积极还是消极。

###待分类的句子：老板，这个新项目很好，它让我感到非常兴奋和有动力，只是我最近发现自己很难有时间陪伴家人了。



Zero-shot

###判断下面句子的情感为积极还是消极。

###待分类的句子：老板，这个新项目很好，它让我感到非常兴奋和有动力，只是我最近发现自己很难有时间陪伴家人了。

###针对此问题有如下示例：

示例：老板，我非常喜欢这个新项目，它让我感到非常兴奋和有动力。
\\n积极。



One-shot

###判断下面句子的情感为积极还是消极。

###待分类的句子：老板，这个新项目很好，它让我感到非常兴奋和有动力，只是我最近发现自己很难有时间陪伴家人了。

###针对此问题有如下示例：

示例1：老板，我非常喜欢这个新项目，它让我感到非常兴奋和有动力。
\\n积极。

示例2：老板，这个项目的工作量太大了，我感到压力很大。
\\n消极。

示例3：老板，虽然项目很有趣，但我最近感到有些疲惫。
\\n混合。



Few-shot

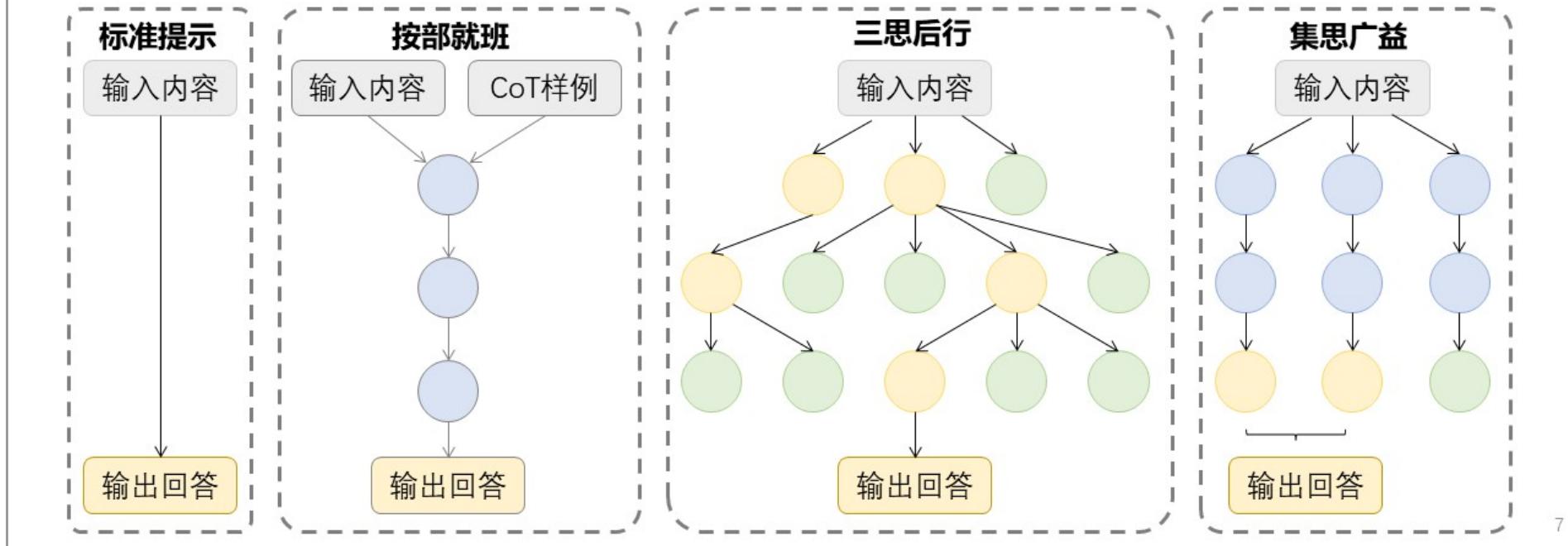
单选题 1分

下列关于思维链提示（CoT）的描述中，哪一项是正确的？

- A CoT要求模型必须输出最终答案，而不能展示中间推理步骤
- B CoT适用于所有任务，包括无需推理的简单分类任务
- C Zero-shot CoT可通过加入一句“Let's think step by step”触发推理能力
- D Few-shot CoT不需要示例也能触发模型推理能力

思维链的分类

在标准Prompt方法上，按照推理方式的不同，可以归纳为三种模式：**按部就班**、**三思后行**和**集思广益**。



智能体 (Agent)



AI Agent vs. Agentic AI

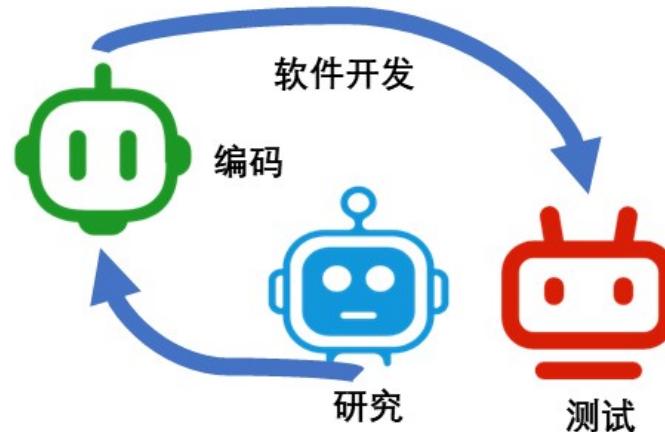
- 单个实体
- 利用工具能力
- 执行但不或短链条推理
- 一个根据指令查询天气的代理



AI Agent

用于自动化执行特定任务的**任务执行者**。聚焦效率与自动化。

- LangChain <https://www.langchain.com/>
- Auto-GPT <https://agpt.co/>



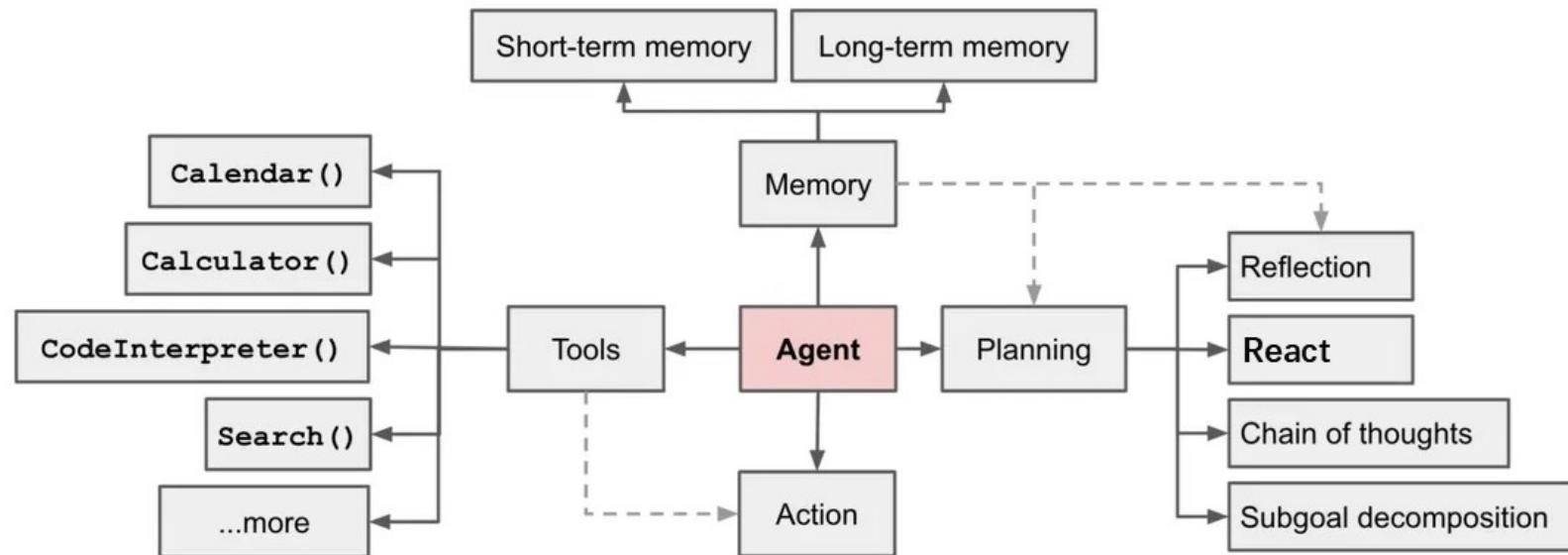
Agentic AI

一个由多个专业代理通过**协作**来完成复杂长期目标的**协同系统**。聚焦协作与负责问题解决。

- AutoGen <https://autogen-five.vercel.app/>
- MetaGPT <https://www.deepwisdom.ai/>

智能体 (Agent)

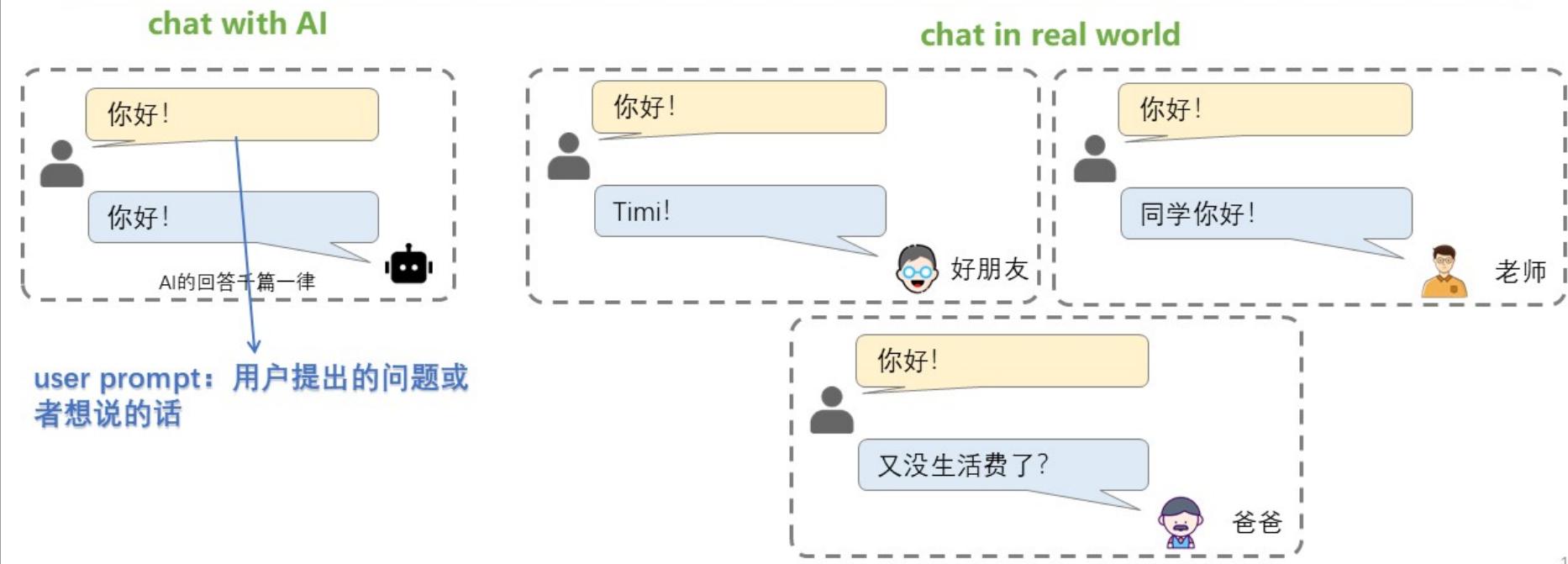
Agent = 大模型(LLM) + 记忆 + 工具 + 规划



LLM+角色

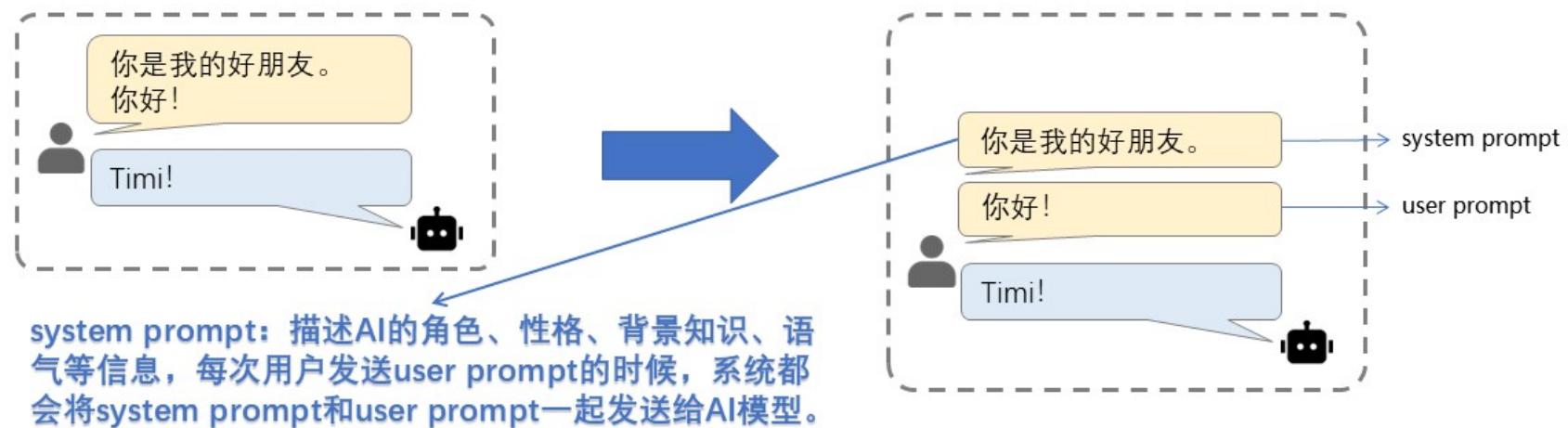
Agent的“思考中枢”

针对一个问题，AI通常只能给出一个**通用的回答**。而在现实世界中，不同的人针对同一问题的回答不同，是因为他们具备不同的人设，而AI不具备人设。



User prompt & System prompt

通过在user prompt中增加人设信息，尝试让AI具有人设。但是每次都带人设信息很麻烦，因此将人设信息单独提出来变成system prompt。



LLM-角色

➤ 初始化:

```
class Agent:  
    def __init__(self, client: OpenAI, model: str = "deepseek-ai/DeepSeek-R1-0528-Qwen3-8  
                tools: List[], verbose : bool = True):  
        self.client = client # <-- 大脑: OpenAI/API客户端  
        self.tools = tools   # <-- 工具: 可调用的函数列表  
        self.model = model  
        self.messages = [  
            {"role": "system", "content": SYSREM_PROMPT}, # <-- 记忆: 初始化系统角色  
        ]  
        self.verbose = verbose
```

➤ SYSTEM_PROMPT——人设:

```
SYSREM_PROMPT = """  
你是一个人工智能助手。你的输出应该与用户的语言保持一致。  
当用户的问题需要调用工具时，你可以从提供的工具列表中调用适当的工具函数。  
"""
```

记忆模块

让Agent能“记住”



➤ 为什么需要记忆模块

记忆系统旨在克服LLM的局限性，赋予智能体以下核心能力：

- **长期保留与高效管理**：存储超出LLM上下文窗口的信息，实现高效检索和过滤，避免信息遗忘；
- **持续知识更新**：通过存储与环境和用户的交互经验，实现自我改进和知识更新；
- **个性化服务**：记录用户偏好和历史互动，提供定制化回应；
- **复杂任务支持**：追踪多Agent任务进展和中间结果，确保连贯完成并优化决策；
- **提升交互质量**：保持上下文连贯性，支持深入推理，并通过反思机制从错误中学习。

<https://aws.amazon.com/cn/blogs/china/agentic-ai-infrastructure-deep-practice-experience-thinking-series-three-best-practices-for-agent-memory-module/>

14

记忆：存储经验和知识，支持长期学习。包含**短期记忆**（如一次任务过程中的多次人类交互）或**长期记忆**（如记录使用者的任务历史、个人信息、兴趣偏好）

➤ 短期记忆

短期记忆是智能体维护当前对话和任务的即时上下文系统。受限于上下文窗口大小，适用于简单对话和单一任务场景。

- 会话缓冲（Context）：使用上下文窗口来处理和记住当前会话中的信息。
- 工作记忆：存储当前任务的临时信息，如中间结果、变量值等。

➤ 长期记忆

智能体用于跨会话、跨任务长期保存知识的记忆形式。依赖于外部存储或知识库。

- 摘要记忆：将长对话内容提炼为关键摘要存储；
- 结构化知识库：使用数据库或知识图谱存储结构化信息；
- 向量化存储：通过向量数据库实现基于语义的记忆检索。

工具使用 (Tool Use)

让Agent能“动手做事”

在生成答案或解决问题过程中，智能体可以调用外部工具（如API、数据库、硬件设备）等，扩展智能体的能力，以完成自身无法直接处理的任务，相当于Agent的“手脚”。



工具注册

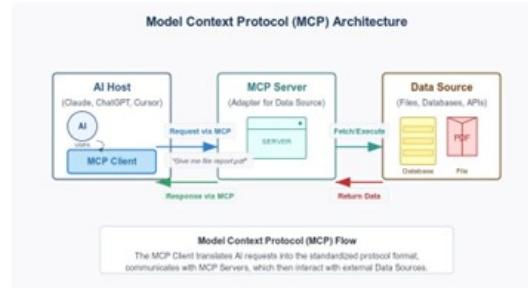
➤ 工具定义：

- 工具名称 (name)
- 功能说明 (description)
- 参数格式 (parameters, JSON schema)

```
name: 'run_shell',
description: 'Write a file to the file system',
parameters: {
  type: 'object',
  properties: {
    command: {
      type: 'string',
      description: 'a shell command to execute'
    }
  },
  required: ['command'],
}
```

➤ 有两种方式：

- **手动注册：**把工具 schema 放进模型输入里 → 简单易用，但扩展性差。
- **MCP 注册：**工具变成独立服务，模型动态获取工具 → 标准化、可复用、可扩展。

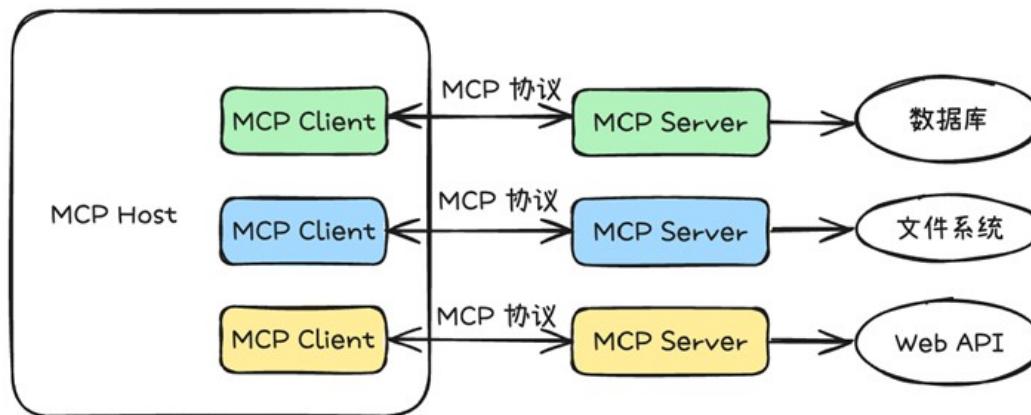


ANTHROPIC

让模型知道工具存在

17

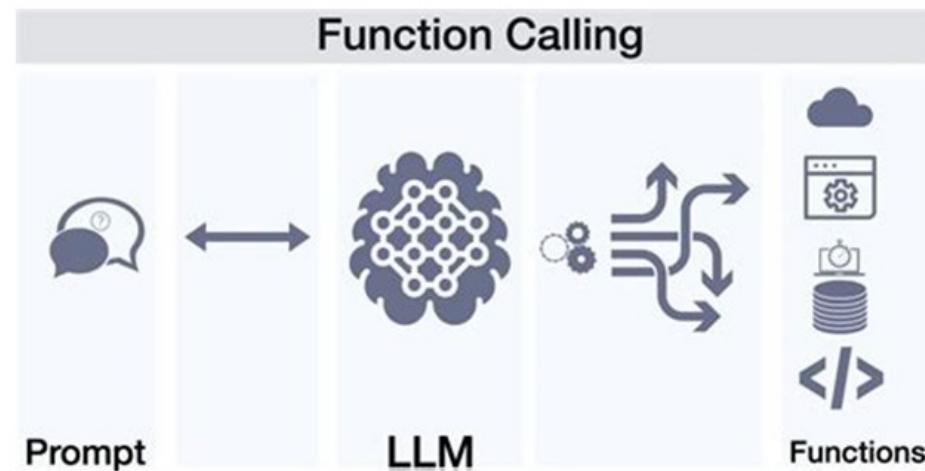
MCP协议



提升了Agent的可扩展性和生态兼容性

- **主机 (Host)**：发起连接的 LLM 应用程序，如 Claude Desktop、IDE 插件等。它负责管理客户端实例和安全策略，是用户与 AI 模型进行交互的平台，同时也承担着集成外部工具、访问多样化数据资源的任务。
- **客户端 (Client)**：是主机内的连接器，与服务器建立 1:1 会话。它负责处理协议协商和消息路由，充当主机与外部资源之间的桥梁，通过标准化的协议接口协调数据传输和指令交互，确保信息的实时性与一致性。
- **服务器 (Server)**：是独立运行的轻量级服务，通过标准化接口提供特定功能，如文件系统访问、数据库查询等。服务器连接外部资源与 AI 模型，向 LLMs 暴露来自不同数据源的内容和信息，还支持创建可复用的提示模板和工作流，帮助开发者设计标准化的交互模式。

工具使用



1. 意图识别：模型分析你的输入，判断是否需要外部工具
2. 工具选择：从工具箱里选择最合适的工具
3. 参数提取：从你的话里提取工具需要的参数
4. 结构化输出：生成符合格式的JSON调用指令
5. 执行与反馈：系统执行工具调用并把结果返回给模型

规划模块

让Agent能“自主思考”

尽管LLM在许多任务上表现出色，但在处理复杂任务时仍存在明显局限性。LLM在处理需要多步骤推理、实时信息获取和动态决策的任务时，常常面临以下挑战：

- **事实幻觉**：模型可能生成看似合理但不准确的信息；
- **缺乏实时信息**：模型训练数据截止后的新信息无法获取；
- **规划能力不足**：面对复杂任务时难以分解和制定有效策略；
- **错误传播**：单个错误推理可能导致整个任务失败；

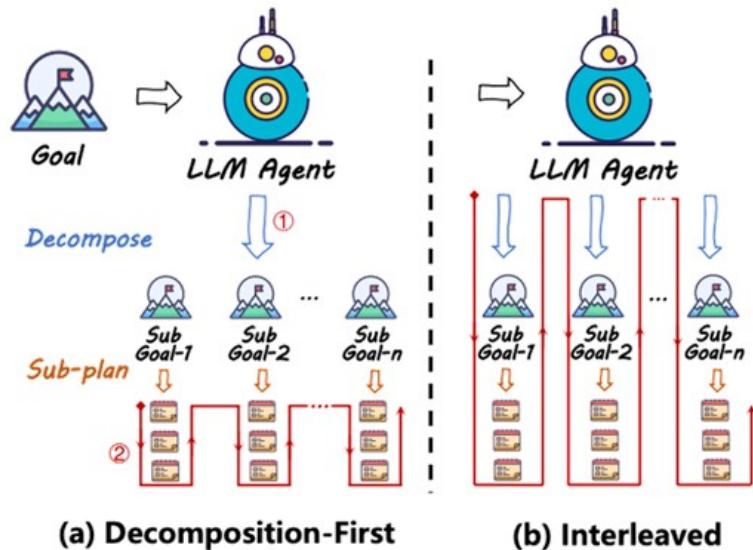


缺乏和外部世界的接触或无法更新自己的知识，会导致事实幻觉和错误传播等问题。

规划模块

让Agent能“自主思考”

任务规划：对复杂任务借助大模型进行分解、规划和调度，并及时观察子任务执行的结果与反馈，对任务及时调整。



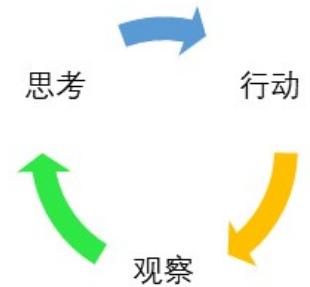
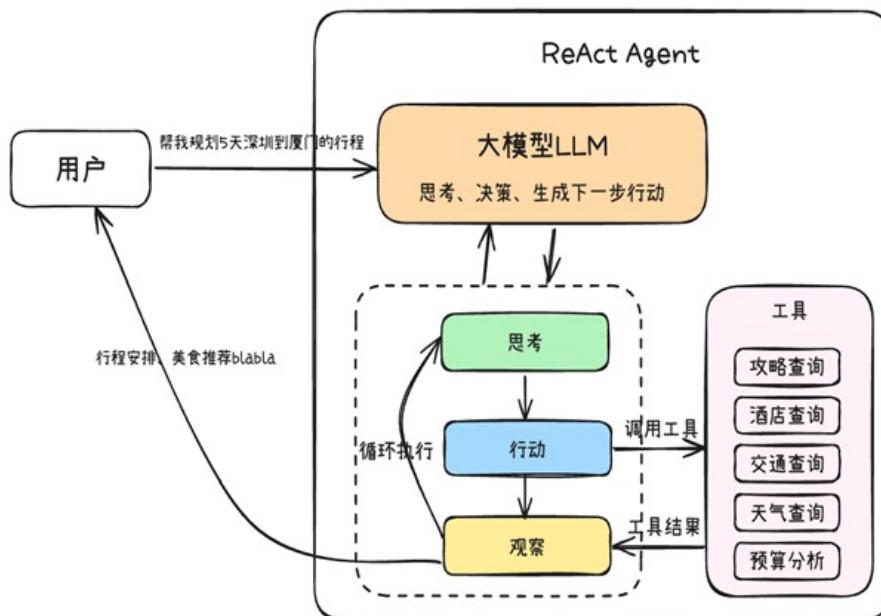
- 任务拆解：复杂任务无法一步到位，规划将复杂目标分解为可执行的子任务
- 方法：CoT

React & Reflexion

ReAct (Reasoning + Acting)

高级规划框架

ReAct 是一个将推理和行为与 LLMs 相结合通用的范例



- **推理**: 模型基于当前任务目标和已有的观察信息，进行逻辑推理和规划。
- **行动**: 模型接收并处理"行动"执行后从外部环境（如工具的返回结果、API的响应、用户的回复）中获得的反馈信息。这些观察结果将作为下一轮"思考"的输入，帮助模型评估
- **观察**: 根据"思考"阶段制定的计划，模型生成并执行一个具体的行动指令

Yao, Shunyu, et al. "ReAct: Synergizing reasoning and acting in language models." *The eleventh international conference on learning representations*. 2022.

22

ReAct (Reasoning + Acting)

高级规划框架

ReAct 是一个将推理和行为与 LLMs 相结合通用的范例



模型生成的 ReAct 轨迹（模拟）—清洗平底锅任务：

- 1 思考 1：平底锅很油腻，我需要有效的清洗方法，可能涉及去油剂。
- 2 行动 1：搜索[清洗非常油腻平底锅的方法]
- 3 观察 1：推荐使用热水的、洗洁精和小苏打；先浸泡，然后擦洗。
- 4 思考 2：我有方法，现在执行步骤。
- 5 行动 2：执行[用热水浸泡锅子；添加洗洁精和小苏打；用海绵擦洗；冲洗干净]
- 6 观察 2：锅子清洁。
- 7 思考 3：如果仍有残留，重复擦洗。
- 8 行动 3：结束[锅子已清洗干净]

Yao, Shunyu et al. "ReAct: Synergizing reasoning and acting in language models." The eleventh international conference on learning representations. 2022.

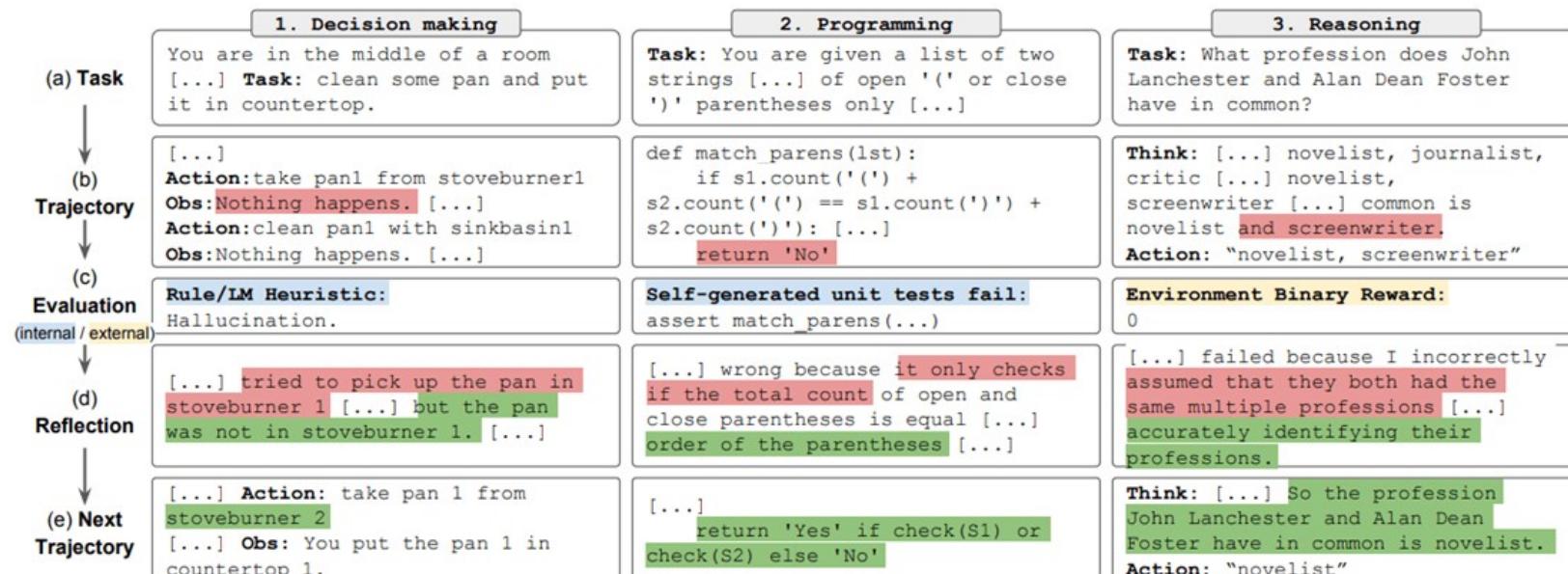
23

Reflexion (ReAct的自我反思)

高级规划框架

Reflexion是一个RL框架，通过语言反馈来强化基于语言的智能体的框架。

通过引入**评估器、自我反思和记忆组件**来拓展 ReAct 框架



Shinn N, Cassano F, Gopinath A, et al. Reflexion: Language agents with verbal reinforcement learning[J]. Advances in Neural Information Processing Systems, 2023, 36: 8634-8652.

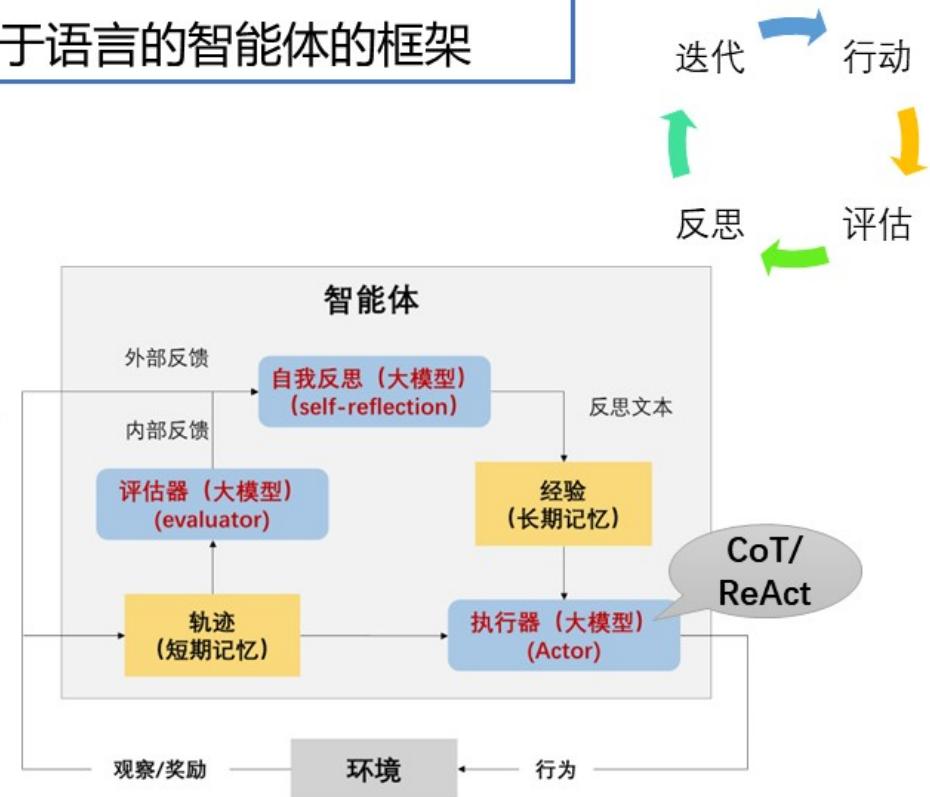
24

Reflexion (自我反思)

高级规划框架

Reflexion是一个通过语言反馈来强化基于语言的智能体的框架

- **Actor**: 根据状态观测量生成文本和动作，在环境中采取行动并接受观察结果，从而形成轨迹。
- **Evaluator**: 评估轨迹，将生成的轨迹作为输入并输出奖励分数。
- **Self-Reflection**: 生成语言强化线索来帮助Actor实现自我完善。自我反思模型利用奖励信号、当前轨迹和其持久记忆生成具体且相关的反思，并存储在记忆组件中。智能体利用这些经验（存储在长期记忆中）来快速改进决策。



Reflexion (自我反思)

高级规划框架

Reflexion 通过行动、评估、反思和迭代来优化响应。假设初始尝试失败，然后通过反思改进。

➤ 初始行动轨迹 (Actor 生成, 基于 ReAct) :

思考 1: 平底锅很油腻, 需要简单清洗。

行动 1: 执行[用冷水冲洗锅子; 擦干]

观察 1: 锅子仍有油渍。

评估 (Evaluator) : 失败 - 锅子未完全清洁。

反思 (Self-Reflection) : 上次尝试使用冷水无效, 因为油渍需要热分解。下次应该搜索有效方法, 包括热水的、洗洁精和小苏打, 并详细列出步骤。存储到记忆: 优先考虑热处理和去油剂。

➤ 迭代行动轨迹 (Actor 使用记忆改进) :

思考 1: 根据记忆, 使用热水和去油剂。

行动 1: 搜索[清洗油腻平底锅的详细步骤]

观察 1: 步骤: 1. 用热水浸泡; 2. 添加洗洁精和小苏打; 3. 用海绵擦洗顽固污渍; 4. 冲洗并检查; 5. 如有残留, 重复。

思考 2: 现在有详细步骤, 可以执行。

行动 2: 执行[上述步骤]

观察 2: 锅子完全清洁。

评估 (Evaluator) : 成功。

Shinn, Noah, et al. "Reflexion: Language agents with verbal reinforcement learning, 2023." URL <https://arxiv.org/abs/2303.11366> 1 (2023).

26

Agent应用实例

- 教育：个性化学习平台/智能辅导和答疑
- 旅游：虚拟旅游助手
- 医疗：智能问诊和咨询服务
- 金融：智能风控
- 制造：语音助手与操作指导
-



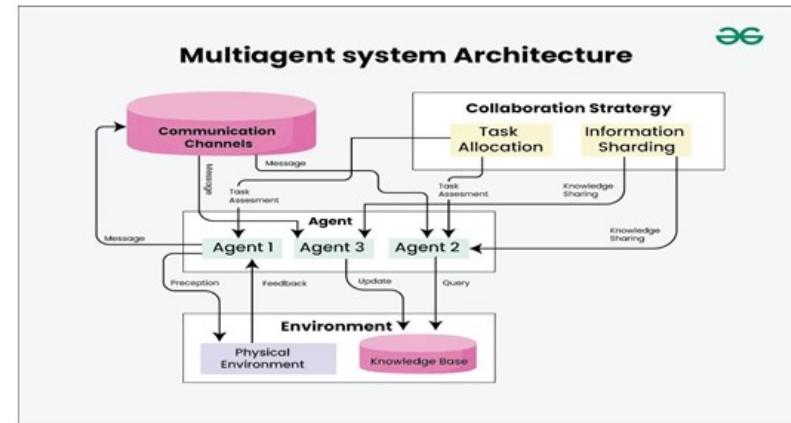
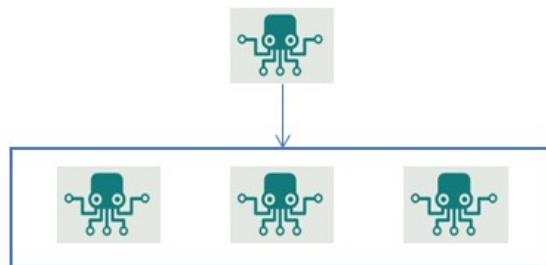
主观题 1分

Agent 的个性化人设、风格和行为规则，主要是通过 [填空1] 组件来设定的；当 Agent 需要在当前任务的多次交互中，快速处理并回忆最近的对话信息时，会利用 [填空2] 机制（如上下文窗口或缓存）来实现这一目标；而当 Agent 需要执行网络搜索、代码计算或文件操作等自身无法直接处理的任务时，它会调用 [填空3]；最后，为了实现任务拆解、逻辑推理和自我修正，Agent 则需要依赖 [填空4] 来实现“自主思考”。

- A. 短期记忆
- B. 规划模块
- C. System Prompt
- D. 外部工具

从 Single-Agent到Multi-Agent

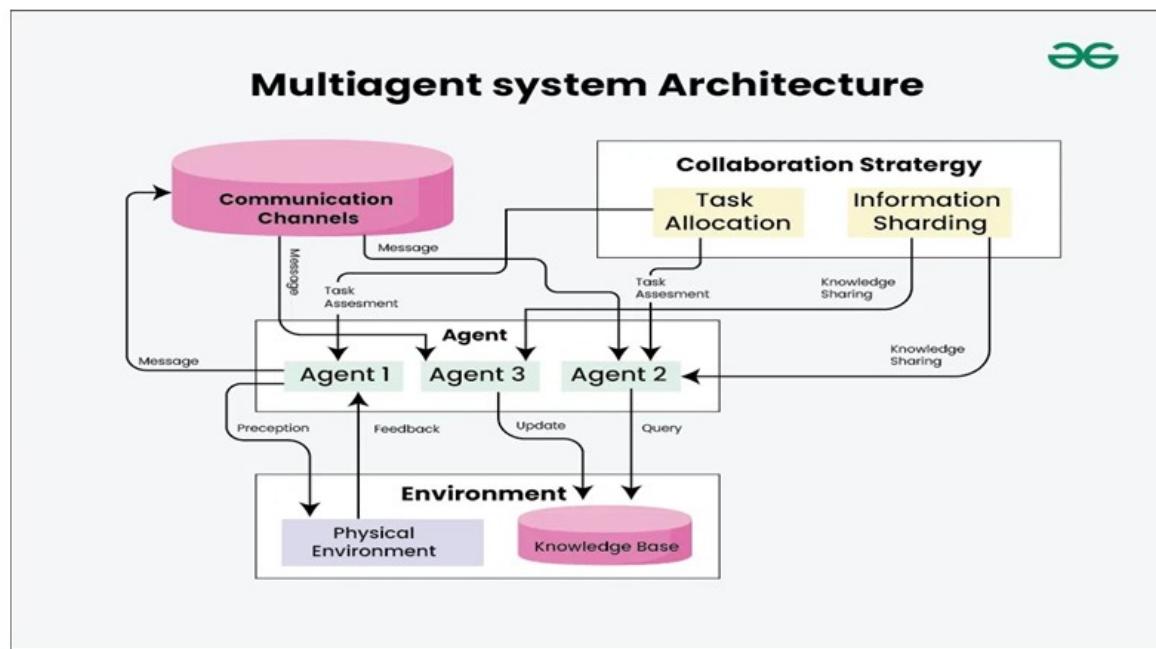
- 当工具过多的时候，单智能体无法正确决定下一步应该调用哪一个工具
- 随着互动轮次的增加，上下文的数量增加，超过ilm的窗口，导致无法正常运行
- 在实际应用场景中，需要不同角色的Agent，将所有的身份信息都放在同一个prompt中可能效果不太好



29

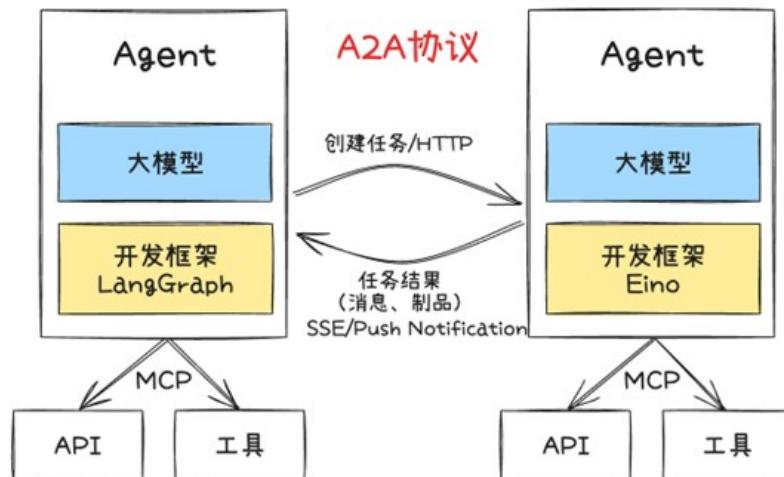
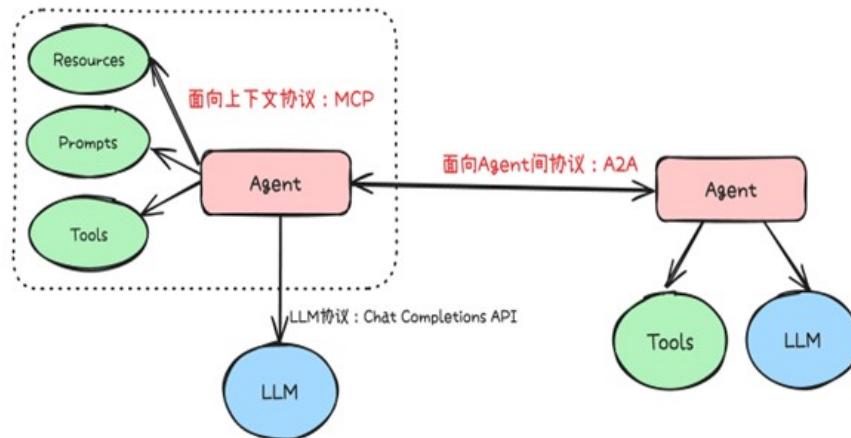
Multi-Agent

Multi-Agent System (MAS) 是由多个智能体组成的系统，这些智能体在同意环境中相互作用，能够自主作出决策。



Multi-Agent

A2A协议旨在为AI智能体之间的互操作性和协作提供标准化的通信方式



- **能力发现**: 智能体通过JSON格式的"Agent Card"宣传自身能力，便于其他Agent发现和调用最合适智能体。
- **任务和状态管理**: 以任务为导向，协议定义了任务对象及其生命周期，支持任务的分发、进度同步和结果（工件）输出，适配长短任务场景。

- **协作与消息交换**: 智能体之间可发送消息，交流上下文、回复、工件或用户指令，实现多智能体间的高效协作。
- **用户体验协商**: 每条消息可包含不同类型的内容片段（如图片、表单、视频等），支持客户端和远程Agent协商所需的内容格式和UI能力。

人工智能导论

Introduction to Artificial Intelligence

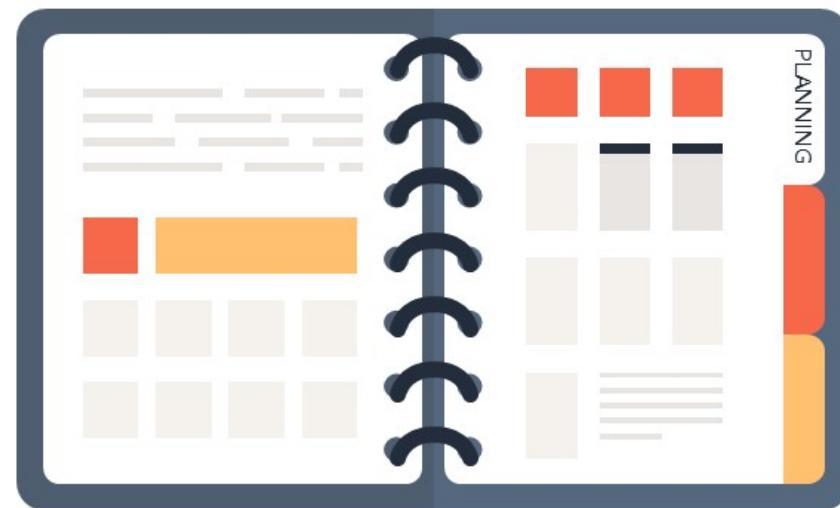
第八章 信息检索



32

Lecture Plan

- 信息检索定义与分类
- 基于关键词匹配的检索
- 基于向量语义的检索
- 大模型时代的信息检索



信息检索定义

➤ 信息检索

信息检索 (Information Retrieval, IR) 是计算机科学的核心领域之一，致力于**从大规模的非结构化数据集合中精准高效地查找与用户需求相关的信息。**



信息检索分类

1. 稀疏检索：基于**关键词匹配**的检索

词汇匹配：看查询词是否精确出现在数据集合/文档中

2. 稠密检索：基于**语义向量**的检索

语义匹配：将文字转化成语义空间中的点，计算点与点的距离

稀疏向量检索

sparse repr: $[0\dots 1 \dots 1 \dots 0..1] \in \mathbb{R}^{d_1}$

- 匹配能力有限，只能捕捉**字面匹配**
- 不可学习

稠密向量检索

dense repr: $[1.03, -5.72, 6.42, .., 9.91] \in \mathbb{R}^{d_2}$

- 能够建模**语义匹配**
- 可学习的

信息检索分类

1. 稀疏检索：基于**关键词匹配**的检索

词汇匹配：看查询词是否精确出现在数据集合/文档中

2. 稠密检索：基于**语义向量**的检索

语义匹配：将文字转化成语义空间中的点，计算点与点的距离

传统检索

智能检索

为什么会长从传统检索逐渐演变到智能检索？

稀疏向量检索

sparse repr: $[0 \dots 1 \dots 1 \dots 0..1] \in \mathbb{R}^{d_1}$

- 匹配能力有限，只能捕捉**字面匹配**
- 不可学习

稠密向量检索

dense repr: $[1.03, -5.72, 6.42, \dots, 9.91] \in \mathbb{R}^{d_2}$

- 能够建模**语义匹配**
- 可学习的

从传统检索到向量检索的演变

➤ 传统数据检索主要针对**传统数据库**

特点→结构化存储

- 严格的关系模型

sid	sname	department	birthday
S-001	张三	计算机系	2020-04-17
S-002	王五	工商管理系	2020-04-17
S-003	汤姆	音乐与舞蹈系	2020-04-16

- 字段类型明确

类型	大小	用途
CHAR	0-255字节	定长字符串
VARCHAR	0-65535字节	变长字符串
TINYTEXT	0-255字节	短文本字符串
TEXT	0-65 535字节	长文本数据
LONGTEXT	0-4 294 967 295字节	极大文本数据
TINYBLOB	0-255字节	二进制字符串

- 依赖SQL进行结构化查询

```
SELECT <目标列名序列> FROM <表名> [<JOIN <表名> ON <连接条件>]
[<WHERE <行选择条件>]
[<GROUP BY <分组依据列>]
[<HAVING <组选择条件>]
[<ORDER BY <排列依据列>]
```



从传统检索到向量检索的演变

➤ 传统检索→精确匹配

苹果

小明买了一个苹果和一个苹果手机。

小明最喜欢苹果了。

小明最喜欢苹果手机了。

小明买了一个蘋果

数据集合

小明买了一个苹果和一个苹果手机。

小明最喜欢苹果了。

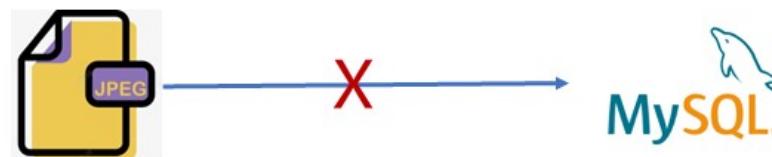
小明最喜欢苹果手机了。

检索结果

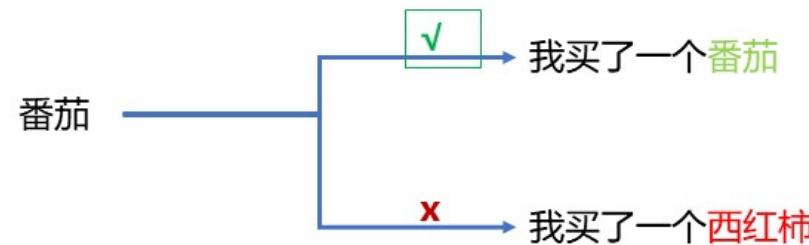
从传统检索到向量检索的演变

➤ 传统检索的不足

- 难以处理高维向量数据(图片、音频等)



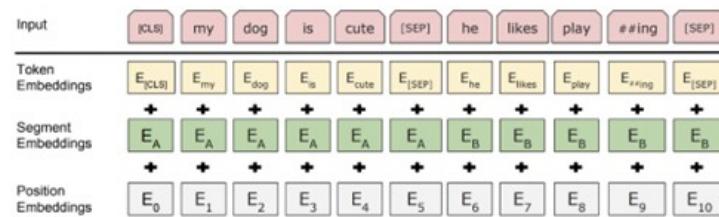
- 传统关键词匹配无法满足语义搜索



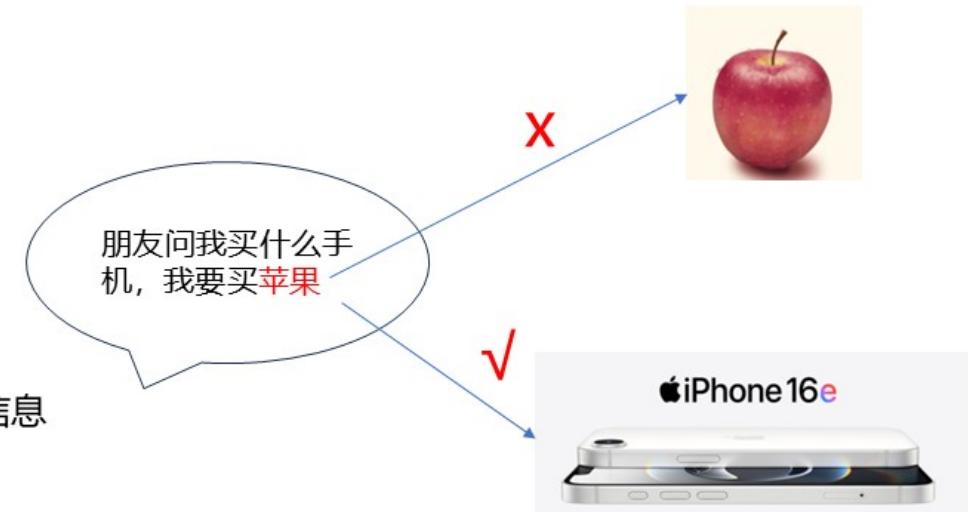
从传统检索到向量检索的演变

➤ 向量检索→语义建模

通过向量化技术，将非结构化数据转化为高维向量，实现从“关键词匹配”到“语义相似性匹配”的跃迁。



例如，BERT实现上下文编码，更好地获取语义信息



从传统检索到向量检索的演变

➤ 向量检索→语义建模

结合文本、图像、音频的多模态融合检索

文搜图



一张在草地上奔跑的狗



图搜文



适合阅读的
下午茶搭配

基于关键词匹配的检索

➤ 布尔检索

利用逻辑算符（AND、OR、NOT）组合关键词，精确筛选数据库中的文献或数据

示例：

- 查询“人工智能OR机器学习”：返回包含任一关键词的结果
- 查询“人工智能AND应用”：仅返回同时包含两词的结果

运算符	功能	示例
AND	交集	“信息AND检索” → 返回同时含“信息”和“检索”的文档
OR	并集	“猫OR狗” → 返回含“猫”或“狗”或两者的文档
NOT	排除	“苹果NOT水果” → 返回含“苹果”但排除与“水果”相关的文档

基于关键词匹配的检索

➤ 布尔检索

关联矩阵索引：对于规模较小的的文档集，可以对文档集构建词项-文档(term-doc)的关联矩阵。

词项	文档				
	动物世界	水果百科	手机推荐	在家上网	智能家居
猫	1	0	0	1	0
狗	1	0	0	0	0
苹果	0	1	1	0	0
香蕉	0	1	0	0	0
手机	0	0	1	1	1
电脑	0	0	0	1	1

基于关键词匹配的检索

➤ 布尔检索

关联矩阵索引：对于规模较小的的文档集，可以对文档集构建词项-文档(term-doc)的关联矩阵。

示例：

布尔表达式：猫 AND NOT 狗 AND 手机

对应向量：猫：1，狗：0，手机：1

检索结果

	动物世界	水果百科	手机推荐	在家上网	智能家居
猫	1	0	0	1	0
狗	1	0	0	0	0
苹果	0	1	1	0	0
香蕉	0	1	0	0	0
手机	0	0	1	1	1
电脑	0	0	0	1	1

基于关键词匹配的检索

➤ TF-IDF (词频-逆向文档频率)

评估一个字词对于一个文件集中的其中一份文件的**重要性**

字词重要性: 与在文件中出现的次数成正比，在语料库中出现的频率成反比 → 出现次数越少区分能力越强

“某个词在某篇文章中的TF-IDF值越大，这个词对这篇文章的重要性越高”



基于关键词匹配的检索

➤ TF-IDF (词频-逆向文档频率)

词频 (TF) : 表示词在文本中出现的次数

$$TF(w, d) = \frac{w \text{ 在文档 } d \text{ 中出现的次数}}{\text{文档 } d \text{ 的总词数}}$$

词语: 苹果

文章: 苹果, 学名 *Malus pumila Mill.*, 别称西洋苹果、柰, 属于蔷薇科苹果属的植物。苹果是全球最广泛种植和销售的水果之一, 具有悠久的栽培历史和广泛的分布范围。苹果的原始种群主要起源于中亚的天山山脉附近, 尤其是现代哈萨克斯坦的阿拉木图地区, 提供了所有现代苹果品种的基因库。苹果通过早期的贸易路线, 如丝绸之路, 从中亚向外扩散到全球各地。

基于关键词匹配的检索

➤ TF-IDF (词频-逆向文档频率)

逆向文档频率 (IDF) : 词在整个语料中是否稀有

包含词条t的文档越少, IDF越大, 说明词条具有很好的类别区分能力

$$IDF(w) = \log \left(\frac{\text{语料库中文档的总数}}{\text{包含词} w \text{的文档数} + 1} \right)$$

TF-IDF: 过滤掉常见的词语, 保留重要的词语

$$TF - IDF = TF_w \times IDF_w$$

基于关键词匹配的检索

TF-IDF算法有没有什么缺陷呢？

较长文档在匹配时因为包含更多词语而占不公平优势（TF值会更大）



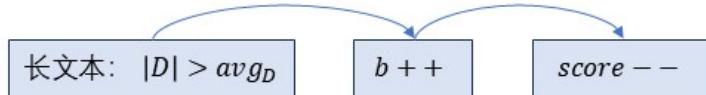
BM25的提出

- 对TF-IDF算法的优化
- 引入文档长度的归一化处理，以解决不同长度文档的偏差问题
- 提供可调参数，对文档长度进行打压

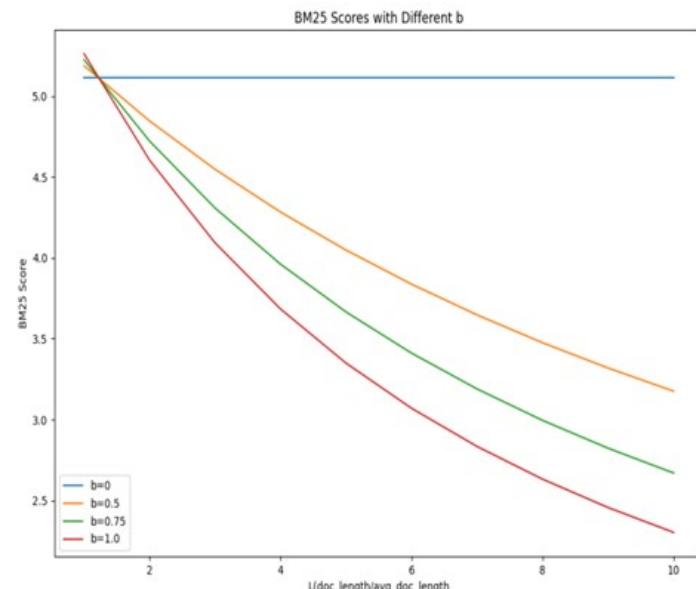
基于关键词匹配的检索

➤ BM25

$$Score(D, Q) = \sum_{q \in Q} IDF(q) \cdot \frac{TF(q, D) \cdot (k_1 + 1)}{TF(q, D) + k_1 \cdot (1 - b + b \cdot \frac{|D|}{avg_D})}$$



- Q : 查询词集合
- D : 文档
- q : 查询中的某个词
- $TF(q, D)$: 查询词 q 在文档 D 中的词频
- $|D|$: 文档 D 的长度
- avg_D : 所有文档的平均长度
- $IDF(q)$: 词 q 的逆文档频率
- k_1 : 控制词频饱和的参数, 常取值1.2或2
- b : 文档长度归一化参数, 范围0.5~1



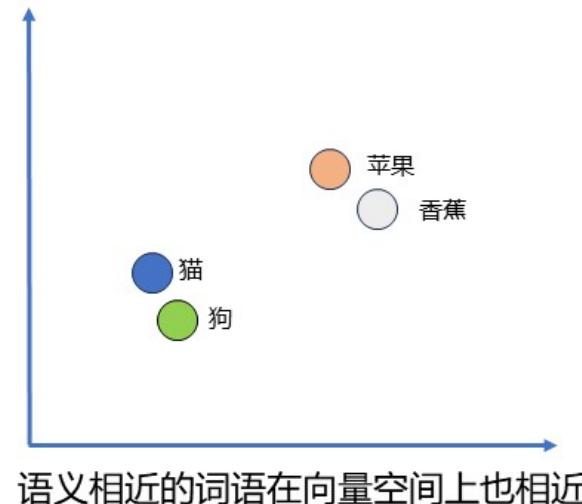
参数 b 越大, 对长文本打压越大

基于向量语义的检索

➤ 词向量

词向量 (Word Vector) , 又称为词嵌入 (Word Embedding) , 将词表中的单词映射为语义空间的实数向量。

- 将文本转化为计算机可处理的形式
 - 文本数据→数值矩阵
- 捕捉语义和上下文信息
- 提升NLP任务的效率和效果
 - 低维向量, 计算资源少



词向量模型是向量语义检索的基础!

基于向量语义的检索

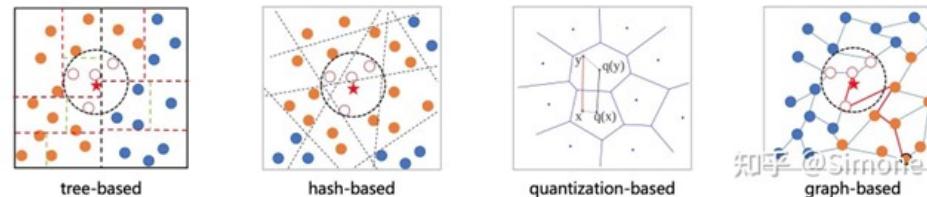
有了词向量模型后，完成将文本转化为词向量

基于向量语义的检索算法先将文本表示为向量后，在向量语义空间中检索与查询向量最相似的向量。

经典的**近似最近邻搜索 (Approximate Nearest Neighbor Search, ANN)** 用于在高维向量空间中快速检索与查询向量最相似的向量。

中心思想：对样本进行分簇，建立索引，提高搜索效率

- 基于树的算法
- 基于哈希的算法
- 基于量化的算法
- 基于图的算法



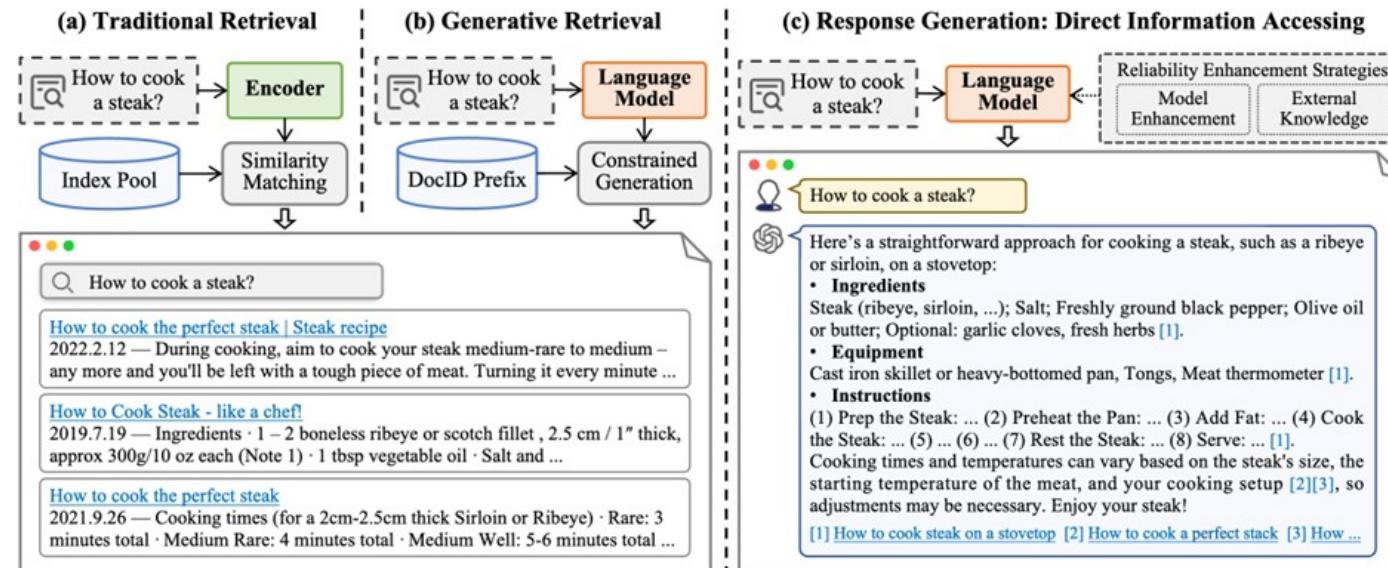
当前向量检索算法主要关注向量空间搜索的效率优化，而检索的质量

则主要依赖于文本或词向量模型的表示能力

大模型时代的信息检索

➤ 从匹配到生成

从直接根据相似度进行检索，到使用大模型优化查询进行检索，再到用大模型生成最终的回复，得到直接的、用户友好的答案。



大模型时代的信息检索

➤ 从匹配到生成

(a) 传统检索

流程:

编码: 将查询转为向量；

索引池: 存放海量的网页/文档向量表示；

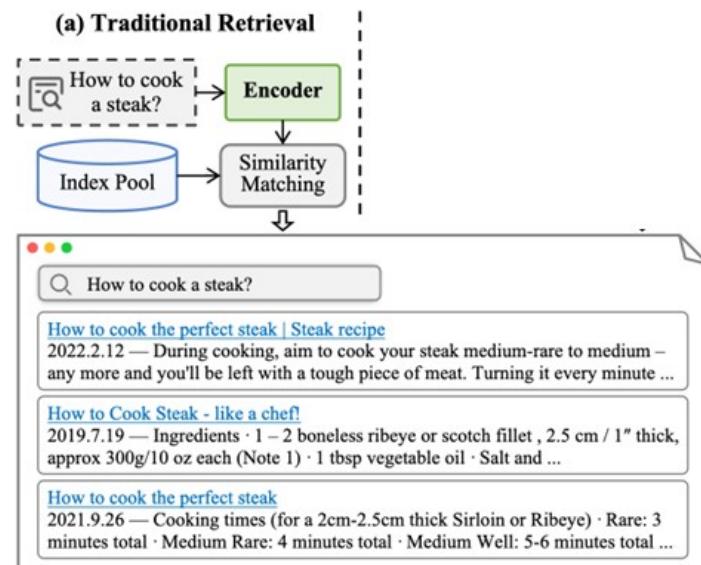
相似度匹配: 计算用户的查询与索引池中的所有文档向量相似度分数；

输出: 返回一个按相似度排序的文档列表。

特点:

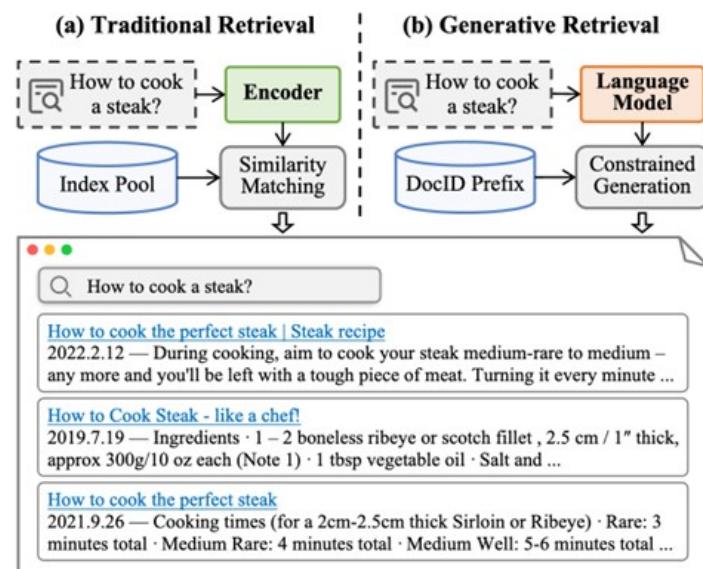
用户负担重: 最终的“缝纫”（整合信息、得出结论）工作需要用户自己完成。

结果是“链接列表”: 输出的是一系列的文档引用，而不是直接的答案。



大模型时代的信息检索

➤ 从匹配到生成



(b) 生成式检索—将检索问题转化为了一个生成问题。

流程:

语言模型: 用户查询直接输入给语言模型;
受限生成: 语言模型直接“生成”出相关文档的唯一标识符(DocID); **输出**: 返回一个相关文档的列表。

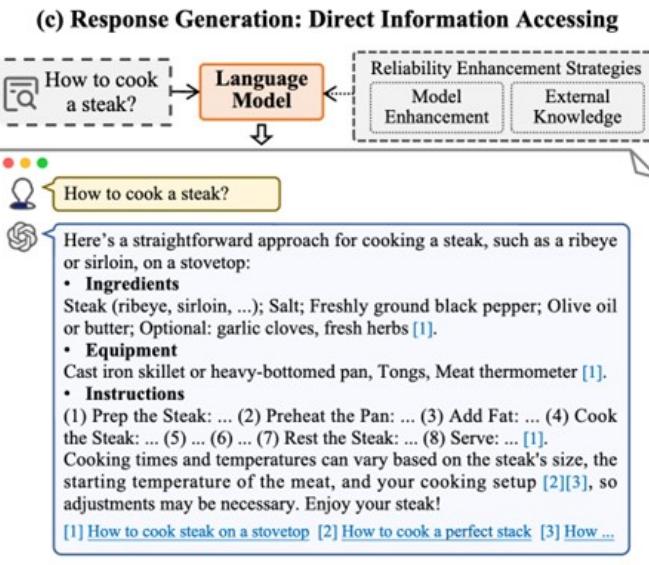
特点:

端到端: 检索过程可以被整合到语言模型的训练中, 实现端到端的优化。

结果仍是“文档列表”: 这一步仍然只是完成了“检索”, 没有生成最终答案。

大模型时代的信息检索

➤ 从匹配到生成



(c) 响应生成：直接信息访问 —— 完整RAG系统的最终形态

流程：

语言模型：用户输入查询后，语言模型直接介入。

可靠性增强策略：语言模型会主动调用“增强策略”来获取外部信息。这可以包括：

模型增强：调用内部更专业的模块。

外部知识：从外部知识库中获取相关的上下文信息。

生成答案：生成一个结构化的、直接的、用户友好的答案。

输出：

直接答案：以易于阅读的格式（如列表、步骤）呈现。

引用溯源：答案中的关键信息点会附上引用角标（如[1], [2]），并链接到原始的来源文档。

特点：

用户负担轻：用户可以直接获得答案，无需再阅读大量原始文档。

可靠与可信：通过引用溯源，解决了大模型的“幻觉”问题，让答案变得可验证。

大模型时代的信息检索

➤ 大模型存在的问题

- **模型幻觉问题**: 生成模型可能在缺乏足够依据时虚构不存在的事实或数据
- **知识滞后性**: 模型自身的知识完全源于训练数据，无法回答训练后的新事件或领域专业知识
- **私有数据限制**: 企业本地知识不公开，通用大模型不具备企业私有化知识

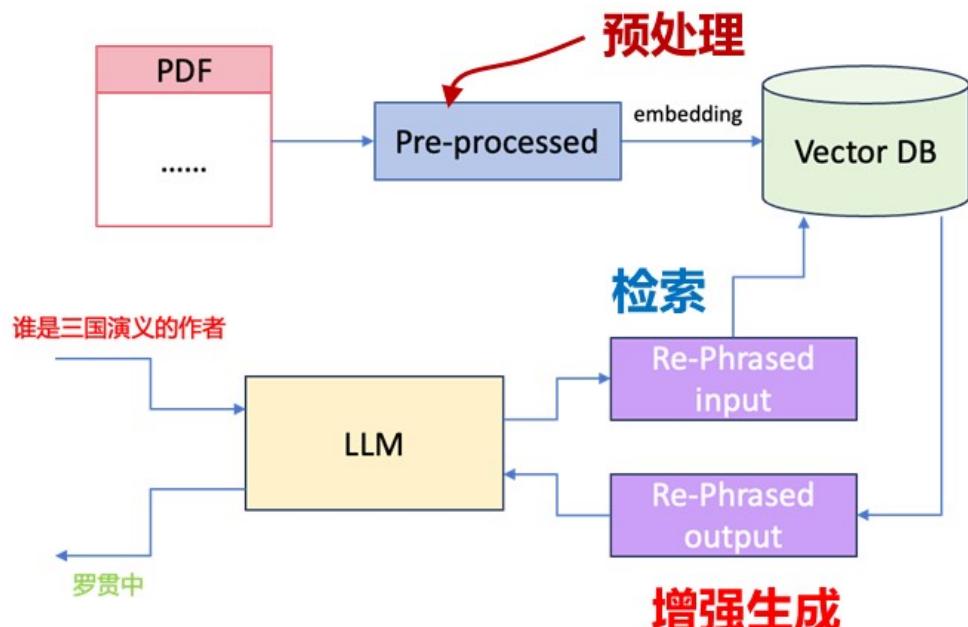
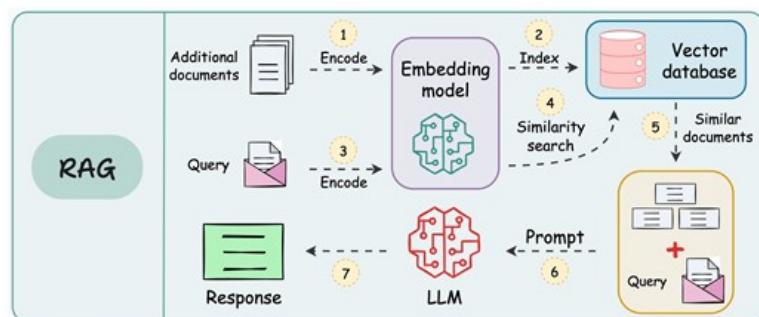


大模型存在幻觉问题

大模型时代的信息检索

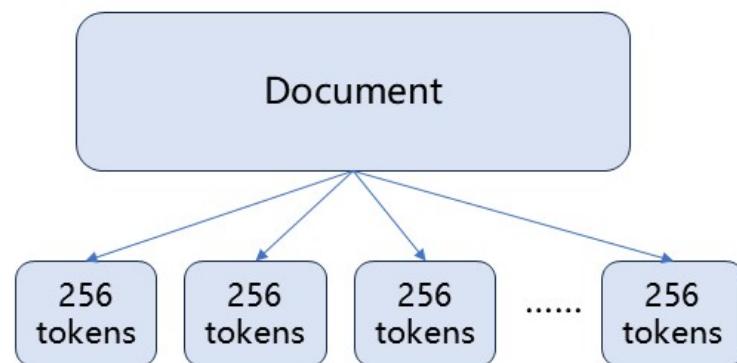
➤ 检索增强生成 (Retrieval Augmented Generation)

检索增强生成技术是结合**信息检索**和**文本生成**的技术框架，旨在通过动态访问外部知识库来增强生成模型的能力



大模型时代的信息检索

➤ 检索增强生成 – 分块

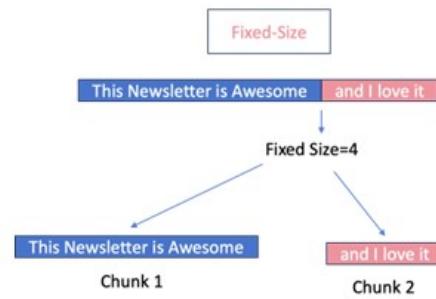


➤ 为什么进行分块?

分块，相当于我们为一本书精心划分章节、段落，并为每个段落提炼一个核心主题。现在，当你想查找那个信息时，你可以迅速定位到相关章节，再找到具体的小节，最后在那几个相关的段落里找到精确的答案。

大模型时代的信息检索

➤ 检索增强生成 – 分块方式



固定大小分块

将文本按照统一长度标准进行分割



基于语义分块

根据句子边界对文本进行分割，如基于句点符号

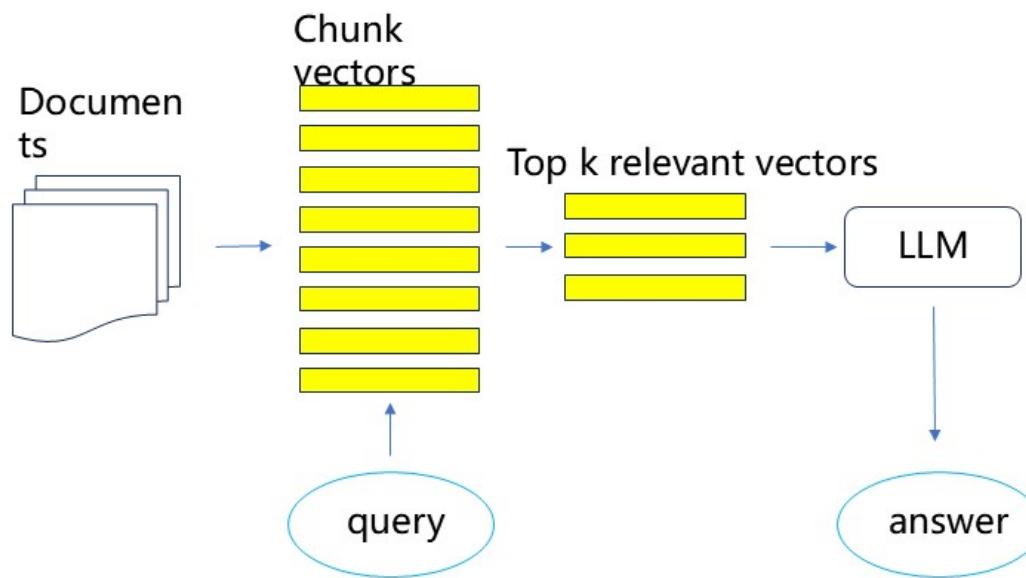


滑动窗口分块

在文本上移动固定大小的窗口来创建具有重叠内容的文本块

大模型时代的信息检索

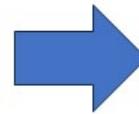
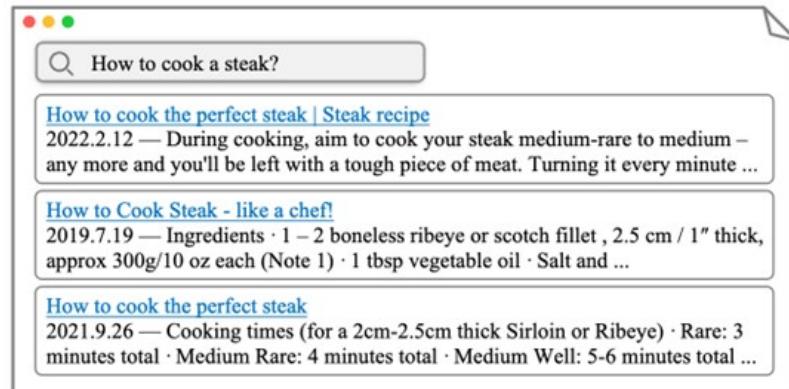
➤ 检索增强生成 – 检索



1. 将查询转化为向量
2. 根据索引遍历查询向量和所有块向量之间的距离
3. 对结果进行排序，选择前k个值返回

大模型时代的信息检索

➤ 检索增强生成 – 增强生成



增强生成

How to cook a steak?

Here's a straightforward approach for cooking a steak, such as a ribeye or sirloin, on a stovetop:

- Ingredients**
Steak (ribeye, sirloin, ...); Salt; Freshly ground black pepper; Olive oil or butter; Optional: garlic cloves, fresh herbs [1].
- Equipment**
Cast iron skillet or heavy-bottomed pan, Tongs, Meat thermometer [1].
- Instructions**
(1) Prep the Steak: ... (2) Preheat the Pan: ... (3) Add Fat: ... (4) Cook the Steak: ... (5) ... (6) ... (7) Rest the Steak: ... (8) Serve: ... [1].
Cooking times and temperatures can vary based on the steak's size, the starting temperature of the meat, and your cooking setup [2][3], so adjustments may be necessary. Enjoy your steak!

[1] [How to cook steak on a stovetop](#) [2] [How to cook a perfect stack](#) [3] [How ...](#)

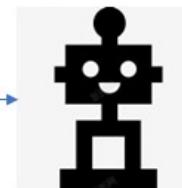
大模型时代的信息检索

➤ 检索增强生成 – 技巧

对用户输入的原始查询进行改写、扩展或分解，生成更适合检索系统理解的查询形式。

周树人是哪里人？

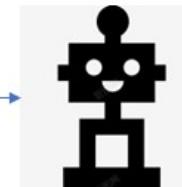
含义模糊，信息不足



？

鲁迅是周树人的笔名，
周树人是哪个省份的人？

指令清晰，信息充足



！

大模型时代的信息检索：智能体驱动的信息检索

智能体正在彻底改变信息检索范式，从传统的被动式关键词匹配转变为端到端解决复杂信息需求的主动式任务解决系统。

⚡ 主动式检索

- ✓ 智能体能够主动理解用户需求和意图，甚至预测用户的潜在需求
- ✓ 对用户指令进行推理分析，将复杂任务自主规划分解并执行
- ✓ 探索环境和调整策略，通过观察-思考-行动进行自主修正与优化

从被动的检索工具到
主动预判和规划的智能
检索助理

💬 对话交互形式

- ✓ 从一问一答的方式转化为具有长期记忆的多轮对话交互
- ✓ 智能体运用对话能力进行迭代式澄清，与用户协同修正完成检索
- ✓ 适应不同用户群体，能够理解口语化、模糊甚至有误的指令

从孤立单轮的查询-应答
模式到连贯有记忆的协
作式对话检索

✍ 生成式回答

- ✓ 深度融合视频、文字等多种模态信息，生成图表等全新形式洞察
- ✓ 避免传统检索的答案列表，直接生成直接生成可落地的解决方案
- ✓ 动态生成个性化、可解释的回答内容，增强用户信任与理解

从零散匹配的信息列表
到个性化多维度整合的
解决方案

大模型时代的信息检索：智能体驱动的信息检索

我想在八月初带家人去成都玩四五天，有什么建议吗？我们家有老人和小孩。给我一份旅行计划的PPT

天工

进行意图识别和需求分析

你的问题已成功接收，天工即将梳理思路，为你呈现全面且精准的答案

您好！我很乐意为您准备一份成都家庭旅行计划的PPT。首先，我需要了解一下您的需求和相关资料。

MCP 工具 | 意图识别

我需要进一步了解您对成都旅行计划的具体需求，以便为您提供最适合的PPT。

好的，我将为您创建一份成都家庭旅行计划的PPT。首先，我会制定一个详细的计划，确保涵盖您带老人和小孩出行的所有重要方面。

待办清单

自动执行

- 整理适合老人和小孩的休息和活动安排建议
- 生成PPT大纲
- 确定PPT的整体结构与各章节重点
- 规划行程安排的展示方式

将复杂需求分解成可执行的子任务

主动探索与任务规划

调用网页搜索浏览工具，与环境进行交互

为用户规划八月初成都四到五天的家庭行程，成员包括老人和小孩，核心是“合理”、“舒缓”和“适合全家”。首先搜索成都家庭游特别是带老人和孩子行程安排，了解经典路线和注意事项。Action 1:

MCP 工具 | 搜索网页 成都四到五天家庭游行程攻略 老人 小孩

MCP 工具 | 搜索网页 成都旅游景点推荐 宽窄巷子 锦里 青城山 都江堰

MCP 工具 | 搜索网页 成都亲子游行程安排

我将深入研究成都各景点，特别是宽窄巷子、锦里、都江堰和青城山，以规划适合老人小孩的舒缓行程。

MCP 工具 | 浏览 <https://www.tibetcn.com/wenda/377.html>

MCP 工具 | 浏览 <https://you.ctrip.com/travels/chengdu/04/3978108.html>

我已收集交通、住宿、餐饮等信息，需整合攻略并细化交通、住宿、餐饮的具体操作细节。

MCP 工具 | 浏览 <https://www.chengduhotels.net/zh/theme/family-hotels-1/>

MCP 工具 | 浏览 <https://www.hotels-of-chengdu.com/zh/theme/family-hotels-1/4/>

MCP 工具 | 浏览 https://cn.tripadvisor.com/Restaurants-g297463-zfp5-Chengdu_Sichuan.html

我需重新搜索具体酒店和餐厅，特别是家庭友好型商家，因之前浏览结果不理想。

MCP 工具 | 搜索网页 成都 春熙路 亲子酒店 推荐

MCP 工具 | 搜索网页 成都 宽窄巷子 家庭友好餐厅

MCP 工具 | 搜索网页 成都 地铁 使用方法 旅游

我已从第六轮搜索中获取成都交通、住宿、餐饮信息，但需进一步完善攻略细节。

观察环境，根据反馈进行分析思考，决定下一步动作

观察-思考-行动
动态调整策略

大模型时代的检索：智能体驱动的信息检索

PPT大纲

- 标题 • P1 成都家庭旅行计划 - 八月初五日游
这是为有老人和小孩的家庭设计的成都旅行计划封面页，包含旅行主题和时间。
- P2 行程概览
五天成都家庭旅行的整体安排，包括每日景点分布和路线规划。
智能体主动询问修改意见
五天成都家庭旅行的整体安排，包括每日景点分布和路线规划。
提出进一步需求，协作式迭代执行
我还想去成都周边转转，再多玩个四天吧
通用场景 联网
- 终止任务

我要修改 继续执行

对话交互协作

生成式回复

成都八月气候与出行准备

八月气候特点

八月的成都天气温和，气候湿润宜人。整体而言是“不冷不热”的时候。

气温
● 平均气温: 22°C-30°C
● 气温范围: 18°C-34°C
● 湿度: 可达80%
● 多云天气概率: 40%-60%

湿度与降水
● 湿度: 60%-90%
● 降水概率: 30%-50%
● 多雨天气概率: 40%-60%

气候提示
建议多带些防晒霜，适时防晒，避免中暑。建议穿着轻便，透气的衣物，避免过热。同时要注意补水，补充水分。

出行

出行概览

五天成都家庭旅行安排

第一天：初遇蓉城
上午：抵达成都，入住酒店
下午：品尝美食（重庆火锅，盆地小吃）
晚：体验成都地道火锅餐

第二天：萌趣熊猫日
上午：成都大熊猫繁育研究基地
下午：青城山（道教、佛教道教文化）
晚：龙门古镇夜游

第三天：世界遗产之旅
上午：都江堰景区（千年水利工程）
下午：武侯祠（三国、诸葛亮文化）
晚：锦里夜市

第四天：老成都生活
上午：宽窄巷子（体验老成都生活）
下午：四川传统武术馆（高跷表演）
晚上：白鹿音乐剧

第五天：自由探索与返程
上午：自由探索（购物、休闲选择其他景点）
下午：往返机场，返回

老人和孩子特别注意事项

老人和小孩出行，需特别关注他们的身体状况和需求。请根据以下建议进行准备。

行程节奏
建议行程过于紧凑，每日景点数量控制在3个以内
保证充足的休息时间，尤其是在午间
行程节奏 leisure，适应不同人的步伐

健康与药品
备好感冒药、腹泻药、晕车药、止痛片、创可贴、蚊虫驱避用品
老人携带自身疾病的常用药
如计划前往高海拔地区，需准备便携氧气袋和相关药物

安全保障
时刻关注老人和孩子的身体状况，避免过量劳累
老人和小孩的身体，需要着重保护，防止走失
老人和孩子体力不支，需提供适当保暖和轻便

深度理解用户需求，生成个性化方案

融合多种外部来源知识

The diagram illustrates a workflow. It starts with a 'PPT大纲' (Outline) slide showing a travel plan for Chengdu. A 'Smart Agent interaction' leads to a modified travel guide titled '成都八月气候与出行准备' (Chengdu August Climate and Travel Preparation). This guide includes weather details, a 5-day itinerary (Day 1: Initial meeting in Chengdu; Day 2: Panda-themed day; Day 3: World Heritage tour; Day 4: Old Chengdu life; Day 5: Free exploration and return), specific travel tips for seniors and children, and a section on health and safety. Arrows indicate the flow from the initial outline to the modified travel guide, highlighting '提出进一步需求, 协作式迭代执行' (Propose further needs, collaborative iterative execution) and '深度理解用户需求, 生成个性化方案' (Deeply understand user needs, generate personalized solutions).

实验作业2： COT实验作业

➤ 作业要求

作业提交截止日期：截止时间：2026-01-04 23:59

提交形式：一个压缩包，命名方式：姓名-学号-COT实验，
包含：

cot_experiment.py：完整的可运行Python代码文件。

实验报告.pdf：详细的实验报告。

第二次作业-cot（下午）

提交地址：<https://send2me.cn/WQWsX4o5/QVGpC4aqzF0yhA>

第二次作业-cot（晚上）

提交地址：<https://send2me.cn/A7R-21nX/QrmX4vf7b5PUkg>

66

实验作业2：CoT实验作业

➤ 作业任务说明

- **核心任务：**编写一个Python程序，调用一个大模型API（如OpenAI GPT-3.5/4，或国内的通义千问、文心一言等），完成一个对比实验。
- **实验内容：**
 - 标准提问（Zero-Shot）：**将一个复杂的数学或逻辑推理问题直接抛给大模型，要求它给出答案。
 - CoT提问（Zero-Shot-CoT）：**在同样的问题后面，追加一句引导词（如：“让我们一步一步地思考。”），引导大模型展示其推理步骤，最后给出答案。
- **对比分析：**你的程序需要分别调用两种提问方式，并记录下模型的完整回答。你需要通过分析输出结果，判断CoT是否以及如何帮助模型得出了更准确的答案。
- **技术要求：**使用openai库或任何你所选大模型提供的官方Python SDK。程序结构清晰，包含必要的注释。能够处理至少3个不同的测试问题，以证明你的发现不是偶然。

67

实验作业3：动手实现 Agent 的工具调用

➤ 作业目标

- 配置环境：成功配置并运行提供的 Agent 代码。
- 工具调用：验证 Agent 在用户提问时能够正确地识别并调用工具函数。
- 理解流程：加深对 Agent 初始化、工具定义和工具调用循环（Function Calling）的理解。

➤ 作业要求

- 截止时间：2026-01-04 23:59
- **下午提交地址：**<https://send2me.cn/OTKEESb1/Rvumi6M0Ur50ww>
- **晚上提交地址：**https://send2me.cn/Wky-_j3_/SiWPYJ9kkT4y2w
- 提交压缩包
 - **实验代码**
 - **实验报告：**实验过程与代码运行成功截图

68