

Agent Loop 第一性原理

2026

你叫得出名字的 AI Agent，核心都是 同一行 while

```
while (true) {
```

问模型 · 执行工具 · 回填结果

```
}
```

Agent Loop

不是比喻 —— 字面意义上的一行

B站 / 公众号 / YouTube : 唐国梁Tommy

官网 <https://tgltommy.com>

这行 while 是 最小内核

Agent Loop 内核

```
while (true) {  
...  
}
```

问模型 · 执行工具 · 回填结果

B站 / 公众号 / YouTube : 唐国梁Tommy

官网 <https://tgltommy.com>

真上生产，还要在内核外包一层 状态机

`while(true)` · Agent Loop 内核

↓ 外包一层生产级状态机 · 缺一不可 ↓

✓ 预算

✓ 权限

✓ 验证

✓ 日志

✓ 回滚

✓ 人工审批

别把它当成全部 —— 但也别小看这个内核

B站 / 公众号 / YouTube : 唐国梁Tommy

官网 <https://tgltommy.com>

第一步 · 去神秘化

把 Agent 从神秘感里拉出来



掀开「神秘」那层幕布，底下只是一台你看得懂的机器

核心差别：它能在回答你之前，**自己去做事**



ChatGPT

一问一答

你
帮我看看这个 bug 怎么修？

它
你大概可以这样改……（给你一段文字）

⊖ 到此为止 · 等你下一句



Agent

自己接着做

你
帮我看看这个 bug 怎么修？

↓ 自己决定，不用你插话



查文件

01

↓ 自己决定，不用你插话



跑命令

02

↓ 自己决定，不用你插话



搜网页

03

做完一件 · 接着下一件 ...

while (true)

Agent 循环的节奏

>> 模型决定

直接答 / 调工具

不要工具

✓
最终答案

循环结束

要工具

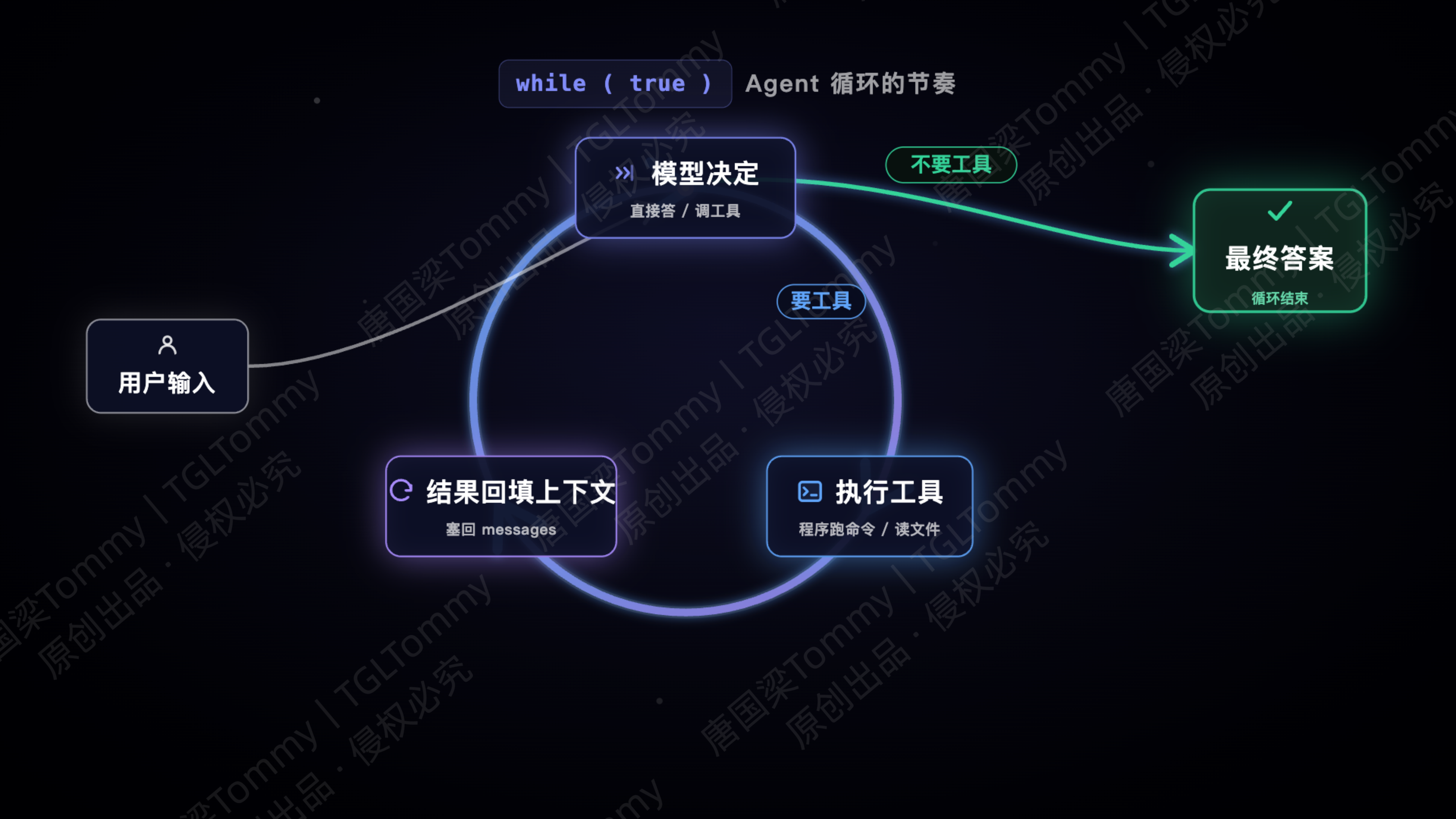
📄 执行工具

程序跑命令 / 读文件

🔄 结果回填上下文

塞回 messages

👤
用户输入



两家权威，说的是同一个词

🧠 OpenAI · Codex 工程团队

“每个 AI Agent 的核心，都是这个循环。”

英雄所见略同

= 同一个循环

🌟 Anthropic · Claude 文档

评估指令



调用工具



接收结果



重复

...直到任务完成。

模型只决定，程序去执行 和 回填

🔧 模型

只输出一个意图，碰不到真实世界

意图便签

```
tool_call: {  
  name: "run_cmd",  
  args: { ... }  
}
```

tool_call

玻璃墙

回填结果

🔄 程序 · 循环代码

真正去做事，再把结果回填

\$ _ 跑命令

📄 读文件

地基： 决定（模型） ⊕ 执行 + 回填（程序）

Agent Loop 最干净的伪代码

LlamaIndex · ~12 行

agent_loop.js

```
1 let messages = [{ role: "user", content: getUserInput() }];
2
3 while (true) {
4   const response = await llm(messages, tools); // 1. 模型推理
5   messages.push(response); // 2. 回复入历史
6
7   if (response.tool_calls) { // 3. 模型请求了工具?
8     for (const toolCall of response.tool_calls) {
9       const result = await callTool(toolCall); // 4. 执行工具
10      messages.push({ role: "tool", content: result, tool_call_id: toolCall.id });
11    }
12    // 5. 带着工具结果回到 while 顶部, 再问一次模型
13  } else {
14    break; // 6. 没有工具调用 → 终止
15  }
16 }
```

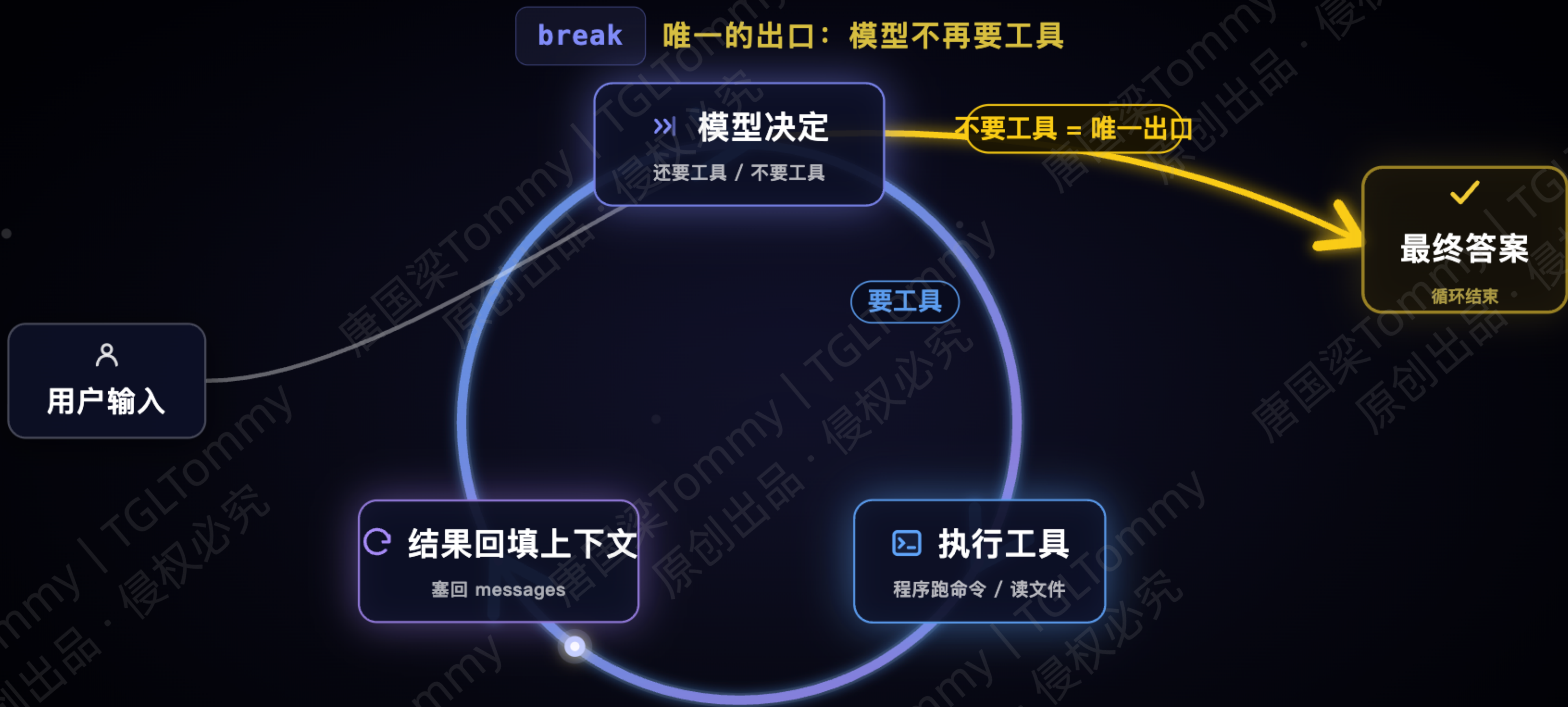
循环体里就三件事

1 问模型
调一次 llm

2 回复入历史
push 进 messages

3 查工具调用
有→执行 / 无→break

break 无工具 → 跳出



break

自然出口之外：生产还有外部约束

>> 模型决定

还要工具 / 不要工具

不要工具 = 唯一出口

③ 预算

要工具

✓
最终答案

循环结束



用户输入

🔄 结果回填上下文

塞回 messages

🛡️ 护栏

📄 执行工具

程序跑命令 / 读文件

🔄 轮数

从外面叫停 · 第三集专讲



平时聊 ChatGPT 脑子里的画面

屏幕对面坐着一个聪明的协作者



我跟它对话

把这个画面换掉：模型是被反复调用的一个函数



```
while (true) { ... }  
  
output = llm(messages, tools)
```

纯函数：输入上下文，输出下一步动作

messages

tools

没有记忆

没有连续意识

每轮纯函数调用

上一轮发生了什么，全靠 **messages** 替它记着

函数没有记忆 —— 记账的是这条列表

```
messages = [
```

U

user

用户输入

A

assistant

模型决定

T

tool

工具结果

A

assistant

模型决定

```
]
```

喂入



```
output = llm(messages, tools)
```

函数 · 无记忆

模型成了一个函数后，你优化的对象变了

旧焦点
🔍 这句 prompt 怎么写

```
prompt = "你是一个专家，请一步步....."
```

停在「怎么跟它对话」的层面

转向

新焦点
A 这个循环怎么设计

```
while(true) {  
    决定 → 执行 → 回填  
}
```

看见后面真正的工程问题

模型是函数，循环是你写的程序

这正是后面几集的思想起点



模型是函数，循环是你写的程序

Firecrawl

把循环总结成四步

act · observe · reason · repeat



工具 = 循环伸出去的「手」

模型只输出文字和决定，碰不到真实世界



让它真的去做事 ——

📄 读文件

📄 跑命令

🌐 搜网页

工具是循环的手，模型靠它 够到外面的世界

思想源头 · ReAct

祖先 · 2022



这套循环不是这两年才发明的

Reasoning + Acting 推理加行动

普林斯顿 + 谷歌 · 2022 年提出

让模型交错产出三种东西 —— 思考 · 行动 · 观察



1

思考 Thought

纯内部推理 · 不改外部世界



2

行动 Action

一次具体的工具调用



3

观察 Observation

结果追加进上下文 · 供下一轮



观察追加进上下文，供下一轮 —— 交错往复

这不就是今天的循环吗？



这正是今天 **Agent Loop** 的概念骨架

现代实现里 —— 穿上了工程的外衣



思考 Thought



藏进模型的推理过程



行动 Action



标准化的结构化工具调用



观察 Observation



一条工具结果消息

骨架没变 · 只是穿上了工程的外衣

ReAct 之前 · 两条原本分开走的老路

路 ①

只推理

代表 · 思维链 Chain-of-Thought

闭门造车 · 缺事实接地



+ 过程清楚 · 可解释

⚠ 一本正经胡说 = 幻觉

路 ②

只行动

+ 跟环境交互

+ 能查东西

⚠ 缺高层规划



走两步就迷路

ReAct · 把两者绑进 同一个循环

只推理
CoT



只行动
交互 · 查询

推理为行动指方向 ⊕ 行动为推理供事实

论文证明 · 推理 + 行动 的范式

多跳问答

Multi-hop QA

事实核查

Fact Verification



系统性地更强

强于单走任一条路

ReAct · 至今没根治的毛病

推理 → 行动 → 观察 (结构刚性)



❖ 结构太刚性

❖ 反复重复同样思考与行动

❖ 原地打转 · 停不下来

论文的解法 · 步数上限

到了上限 → 强制停



迭代步数预算

1

2

3

4

5

6

7

5 / 7 → 强制停

2022 · 步数上限

跑 5 / 7 步强停（兜底土办法）

演化



max_turns · 迭代预算

生产级 Agent 必备护栏（停下来 = 第三集）

Anthropic 的区分很干净 —— Agent 不是 工作流 (Workflow)

Workflow

预先定义好的代码路径



第一步 谁决定下一步：你（步骤人写死） 第二步 第三步

区别

VS

= 谁决定
下一步

Agent

每轮动态决定下一步



谁决定下一步：模型（每轮现选）

Agent 更强 也 更危险，根子都在 自主

自主 (autonomy) = 同一枚硬币的两面

✓ 更强

能处理你没预料到的情况

遇到意外路况能自己绕路、自己补救



⚠ 更危险

可能跑偏

没人写死的路径 · 越跑越久越贵

成本失控

Anthropic 建议：能用简单方案就别上 Agent —— 本质是用更高延迟和成本换性能

ReAct · 2022

用循环生成数据 → 反过来 微调一个更小的模型

思考

Thought

行动

Action

观察

Observation

一条条「思考+行动+观察」的轨迹

循环 → 熔炉 → 锻造小模型

微调 → 更小的模型

≈3,000 条

模型自跑的「正确轨迹」

同一个循环 · 两个身份

用循环生成数据 · 再拿数据训练模型

2022 ReAct 就有原型 · 第五集展开



推理时

产品

推理循环跑出答案



数据引擎

采样轨迹



训练时

数据引擎

同一循环 = 训练采样轨迹

今天强化学习训练全面爆发的那套， 根子就在这儿

`while ( true )`

最底层都是同一个循环

1 问模型

模型 = `llm(messages, tools)`

直到模型不再要工具，循环结束

3 回填结果

2 执行工具

⚠ 难的从来不是让它跑起来

最重要的心智转变



❌ 不是聊天对面的**协作**者

✓ 是被程序反复调用的**函数**
`llm(messages, tools)`

🕒 祖先 · 2022

🔄 老毛病 · 停不下来

NEXT EP

下一集

上下文 context —— Agent 真正的战场

📈 跑得越久越贵

📦 上下文窗口被一点点占满

版权声明

唐国梁 Tommy 原创出品 | 熵智未来（深圳）科技有限公司。保留所有权利。

本课程及配套资料，包括但不限于课程视频、PDF 文档、课件、代码、图表、示例项目、知识库文档及其他学习材料，仅供购买本课程的用户本人学习使用。除法律法规另有规定或获得权利人书面授权外，任何个人或机构不得对本课程及配套资料进行录屏、翻录、对外复制、传播、上传、分发、转售、改编、翻译、节选或汇编；不得将其用于商业培训、企业内训、付费产品、商业化知识库建设、数据集构建、AI 模型训练、微调或其他商业用途。

本课程及配套资料中涉及的第三方开源软件、模型、代码、数据集、商标、图片及其他素材，其相关权利归原权利人所有，使用时应遵守对应许可证、授权条款或使用规则。如发现侵犯本课程及配套资料相关权益的行为，权利人有权依法要求停止侵害、删除下架、赔偿损失，并追究相应责任。

侵权举报 / 授权合作: legal@tgltommyai.com