

NiDD AI+ 研发数字峰会
AI+ Development Digital summit

第5届

AI Agent构建智能应用的新范式

——从 LLM 到 Agentic workflow

揭光发 | Tencent

科技生态圈峰会 + 深度研习



—1000+ 技术团队的选择



敦煌站

K+ 思考周®研习社

时间: 2025.08.29-30



上海站

K+ 金融专场

时间: 2025.09.26-27



香港站

K+ 思考周®研习社

时间: 2025.11.17-18



K+峰会详情



上海站

AI+研发数字峰会

时间: 2025.05.23-24



北京站

AI+研发数字峰会

时间: 2025.08.08-09



深圳站

AI+研发数字峰会

时间: 2025.11.14-15



AIDD峰会详情



揭光发

腾讯技术专家

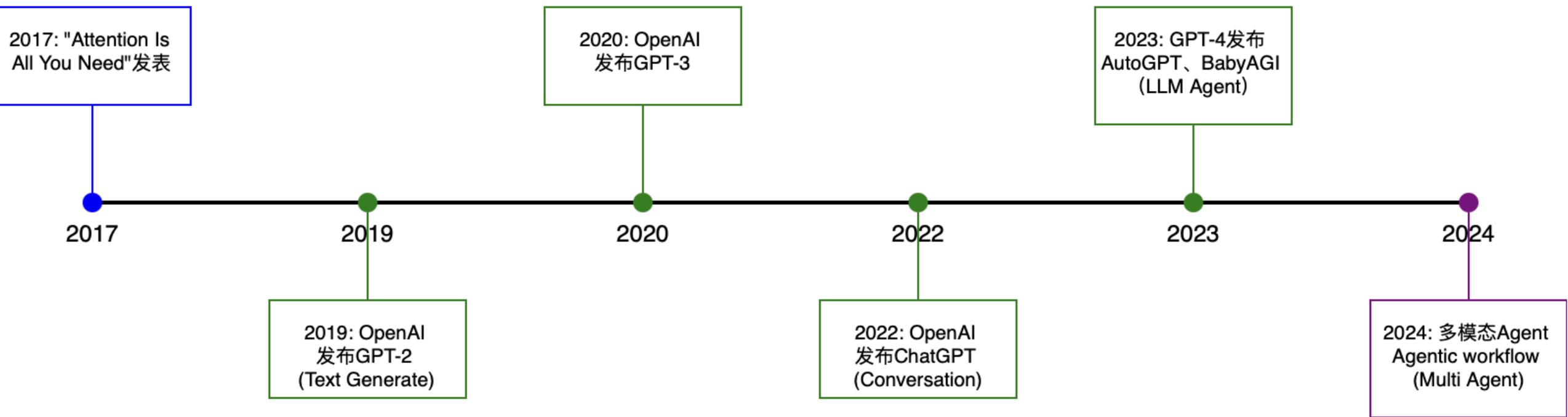
19年研发与团队管理经验，前腾讯云TVP。

全栈低代码开发专家，腾讯低代码Oteam负责人

大模型应用早期实践者与布道师。在团队中深度践行LLM对研发提效，个人代码中AI含量达90+%。

国内顶级行业及技术峰会担任大模型相关话题讲师或出品人10+次。

▶ Transformer到多模态Agentic Workflow的演进



► Core -- Text Generate

指令

上下文

输入数据

示例

格式说明

输出指示

评估标准

步骤引导

Prompt

下面我将要讲一个故事。

角色设定

约束条件

在很久很久以前，有一个小村庄坐落在群山环抱之中。村子里住着一群淳朴善良的人们，他们过着平静而简单的生活。村子里有一位老奶奶，她以制作美味的糕点而闻名。每当村里有庆典或节日时，大家都会聚集在她家门前，等待品尝她新鲜出炉的点心。

Generate

诗歌创作

语言翻译

情感分析

广告文案

代码生成

文本分类

▶ LLM生态的两大阵营

Prompt engineering

提示词工程师

应用层

Agent

Function Call

Flow

RAG

FoT

ReAct

Memory

Model pre-training

炼丹师

BERT

算法层

RLHF

GPT

Context window

MoE

Fine-Tuning

60B

Emergence

LLM APP: 从Text Completion 到 Agent

▶ 1. text completion

请给我讲一个中国的神话故事：



故事名称：《嫦娥奔月》

相传在古代，有一位美丽的仙女名叫嫦娥。她是天上的玉兔宫女，有着清秀的容颜和出色的歌舞技巧。嫦娥住在天宫中的玉帝府，和其他仙子一起照顾玉帝的一切。

一年的八月十五，也就是我们现在所说的中秋节，玉帝要在天宫中举行盛大的宴会。天宫中的所有仙子都要穿上美丽的仙衣，献上自己的歌舞。而嫦娥的舞蹈更是被玉帝高度赞扬，受到众仙子的羡慕和称赞。

然而，在宴会之后，嫦娥偶然间听到众仙子们私下的议论，说她的舞蹈并不是来自自己的努力和天赋，而是来自玉帝赐予的仙丹。嫦娥非常生气和伤心，她不愿意被称赞的舞技是假的，于是决定离开天宫，寻找自己真正的才华和命运。

嫦娥来到了蟠桃园，邂逅了一位名叫后羿的有心人。后羿是神箭手，有着非凡的射箭技巧。两人很快坠入了爱河，决定结为夫妻。

然而，嫦娥并没有忘记自己离开天宫的原因，她求助于爱护她的玉皇大帝。玉帝告诉嫦娥，他无法让她还原成凡人，只能让她变成一个永远居住在月宫的仙子。而且，嫦娥每年只能在八月十五的夜晚回到天宫与丈夫后羿团聚。

嫦娥听从了玉帝的话，变成了一个永远居住在月宫的仙女。每年的八月十五，人们都能在月宫中看到嫦娥和后羿在天宫中团聚的画面，就如同我们现在的中秋节一样。

从此，嫦娥的美丽和舞技都变成了人们心中最美的神话，每逢中秋佳节，人们都会抬头仰望皓月，祈愿幸福团圆。嫦娥也成为了月宫的守护神，保佑人间的和平和幸福。

而在天宫中，嫦娥也找到了自己的真正才华，每天在玉兔宫中与众仙子一起跳起美妙的舞蹈，让人们在夜晚欣赏到天宫的璀璨之美。从此，嫦娥和后羿的爱情也成为了人间和天界最为动人的神话。

▶ 2. Conversation Chat

USER

请给我讲一个中国的神话故事



ASSISTANT

好的，以下是一个中国神话故事《女娲补天》：

很久很久以前，天地混沌一片，乌云密布，狂风呼啸。人们苦不堪言，生活在恐惧中。于是，在这个时候，女娲出现了。

女娲是中国古代神话传说中的女神，她是万物的创造者。女娲看到人们的疾苦，决定帮助他们。她先用五色石块熔化成五彩的巨石，然后凿成长长的山川，修筑了河流湖海，让大地变得有了形状。

可是，女娲的工作还没有结束。因为天空始终阴沉，日月星辰无法照耀大地。女娲在心中默念着法术，她取出神巫的铜镜，用铜镜反射出太阳，再取出玉石，制造了一轮明亮的月亮和漂亮的星星。大地上终于重见光明。

然而，在女娲补好大地之后，她发现天空中有一个大洞，一不留神，就会有混沌的风沙倾泻，带来浩劫。为了保护世界，女娲决定补天。她飞到高空，依次取出五色石，用来修补天上的大洞，最后补好了整个天空。

女娲补天的壮举感动了众生，人们都在欢呼女娲的名字。从此以后，大地上有了山川河流，天空有了日月星辰，人间光明祥和，生活愈发美好幸福。

这个神话故事传颂了女娲的伟大造化之功，也展现了中国古代人们对女神的崇敬与向往。女娲补天，成为中国文化中传承千年的神话传说，激励着人们奋发向前，不断创造美好的世界。

▶ 2. Conversation Chat

```
from openai import OpenAI

client = OpenAI()

1 个用法
def translate(text):
    response = client.chat.completions.create(
        model="gpt-4",
        messages=[
            {"role": "system", "content": "你是一个翻译官，你能够根据输入的文本完成中英文之间的翻译，请把用户的输入翻译成另一种语言。"},
            {"role": "user", "content": text}
        ],
        stream=False,
    )
    print(response.choices[0].message.content)

if __name__ == '__main__':
    while True:
        text = input('请输入要翻译的文本: ')
        if text == 'exit':
            break
        translate(text)
```

请输入要翻译的文本: 你好, 世界

Hello, world.

请输入要翻译的文本: 你叫什么名字?

What is your name?

请输入要翻译的文本: 我爱你

I love you.

请输入要翻译的文本: 我想学人工智能编程, 应该怎么入手?

I want to learn artificial intelligence programming, how should I start?

请输入要翻译的文本: |

▶▶ 2. Conversation Chat

* LLM

GPT-4o

* 提示词模板语法

f-string jinja2

f-string仅支持{var}语法使用参数；jinja2使用双花括弧外，还能支持判断、循环等逻辑控制。

* 提示词

你是一个翻译官，你能够根据输入的文本完成中英文之间的翻译，请把用户的输入翻译成另一种语言。
用户：{input}
翻译：

请根据所选的模板语法编写提示词，确保提示词中至少使用了{input}（f-string语法）参数，该参数代表用户的输入。

参数结构

类型	名称	显示名	字段说明	操作
对象	请输入	参数结构	请输入	+
字符串	input	input	请输入	

根据提示词内容生成参数结构，请勿手动增加或删除字段

输出格式

文本 json

当选择json格式时，需要在提示词中明确让LLM返回json数据，并告知其json的具体字段结构。

保存

预览-translator

你好，世界

Hello, World

你叫什么名字？

What is your name?

我爱你

I love you

我想学人工智能编程，应该怎么入手？

I want to learn artificial intelligence programming. How should I get started?

清空内容

输入一条信息

▶ 3. Tool use -- Agent

Plugin store

Filters: New | Most popular | All plugins

- Instacart** (Install) - Ask about recipes and discover the ingredients you need—then get it all delivered from local stores.
- Zapier** (Install) - Interact with over 5,000+ apps like Google Sheets, Gmail, HubSpot, Salesforce, and thousands more.
- KAYAK** (Install) - Search flights, stays & rental cars or get recommendations where you can go on your budget.
- Expedia** (Install) - Bring your trip plans to life – get there, stay there, find things to see and do.
- Wolfram** (Uninstall) - Access computation, math, curated knowledge & real-time data through Wolfram|Alpha and Wolfram Language.
- Speak** (Install) - Learn how to say anything in another language with Speak, your AI-powered language tutor.

About plugins

Chat Interface:

User: 腾讯最新的股票是多少

AI: Finished browsing

AI: 腾讯的最新股票价格为44.33美元，数据截至2023年5月15日¹。

User: 换算为港币和人民币

AI: Finished browsing

AI: 腾讯的最新股票价格是44.33美元。换算为港币和人民币如下：

44.33美元 = 44.33 * 7.84225港币 = 347.564港币^{1,2}。

44.33美元 = 44.33 * 6.95965人民币 = 308.317人民币^{3,4}。

▶ 定义AI / LLM Agent

名字:

说明: 让用户及其它agent了解, 必要时使用它

角色定位(instructions): 你是一位xxx, balabala

知识: 该agent所在领域的更多的知识文本, in Context

工具集: agent可以使用什么工具来解决问题

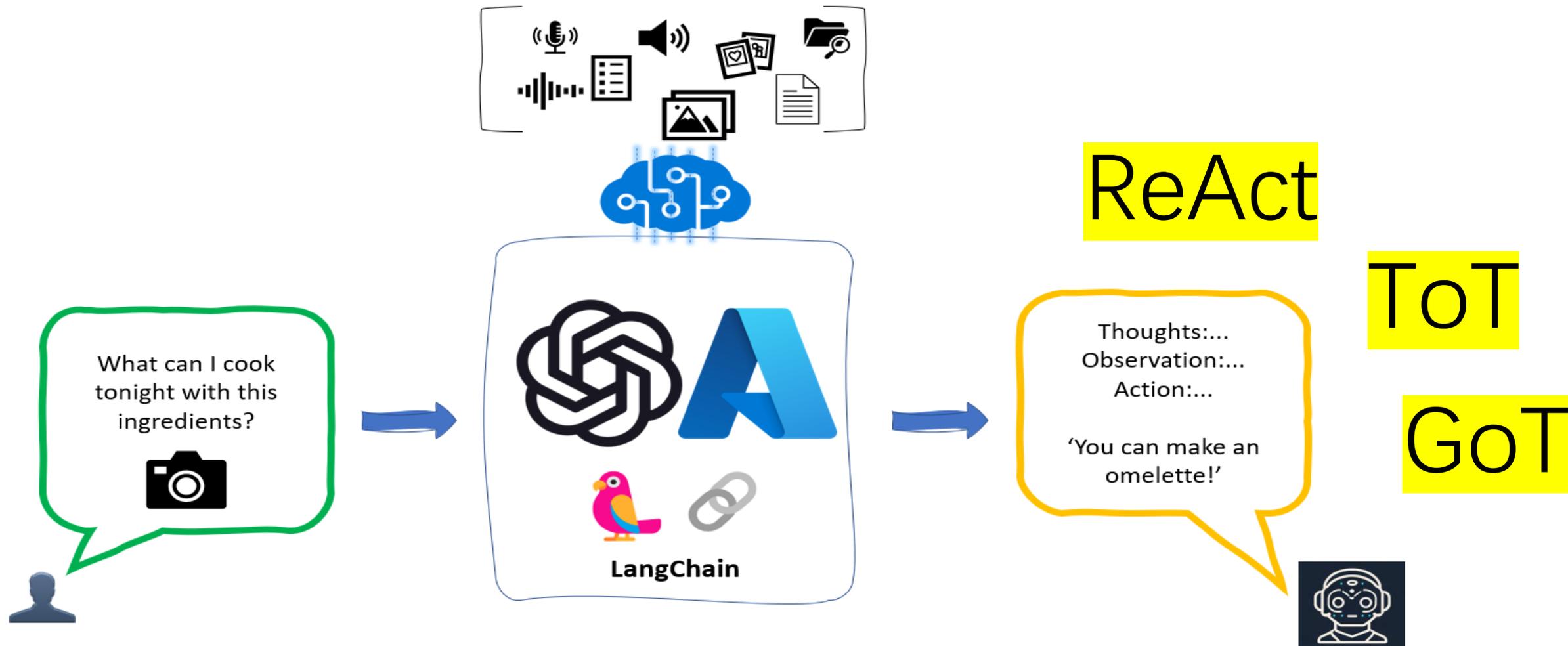
LLM: 把上述信息作为prompt的内容扔给LLM,
它来推理并拆解需求, 并回答/解决你的需求



▶ 流行的Agent推理与执行框架

- ReAct: 早期社区的主流, 基于Prompt工程的方案
 - Question
 - Thought
 - Action
 - Observation
- Function Call: Open AI 官方规范
 - 把工具使用能力, 预训练进模型
 - 行业普遍跟进

▶ Agent推理与执行框架 -- ReAct



▶ Agent推理与执行框架 -- ReAct

(1) Hotspot QA

Question: Aside from the Apple Remote, what other device can control the program Apple Remote was originally designed to interact with?

(1a) Standard

Answer: iPod



(1b) CoT (Reason Only)

Thought: Let's think step by step. Apple Remote was originally designed to interact with Apple TV. Apple TV can be controlled by iPhone, iPad, and iPod Touch. So the answer is iPhone, iPad, and iPod Touch.

Answer: iPhone, iPad, iPod Touch



(1c) Act-Only

Act 1: Search[Apple Remote]

Obs 1: The Apple Remote is a remote control ...

Act 2: Search[Front Row]

Obs 2: Could not find [Front Row]. Similar: ...

Act 3: Search[Front Row (software)]

Obs 3: Front Row is a discontinued media center software ...

Act 4: Finish[yes]



(1d) ReAct (Reason + Act)

Thought 1: I need to search Apple Remote and find the program it was originally designed to interact with.

Act 1: Search[Apple Remote]

Obs 1: The Apple Remote is a remote control introduced in October 2005 by Apple ... originally designed to control the Front Row media center program ...

Thought 2: Apple Remote was originally designed to control the Front Row media center program. I need to search Front Row next and find what other device can control it.

Act 2: Search[Front Row]

Obs 2: Could not find [Front Row]. Similar: ['Front Row Seat to Earth', 'Front Row Motorsports', 'Front Row (software)', ...]

Thought 3: Front Row is not found. I need to search Front Row (software) .

Act 3: Search[Front Row (software)]

Obs 3: Front Row is a discontinued media center software ...

Thought 4: Front Row (software) is controlled by an Apple Remote or the keyboard function keys. So the answer is keyboard function keys.

Act 4: Finish[keyboard function keys]



▶ Agent推理与执行框架 -- ReAct



```
@tool
def get_teleplay_info(tid: str) -> str:
    """
    useful for when you need to get a teleplay information by the teleplay id (tid), the tid must be a valid teleplay id
    """
    info = [e for e in TELEPLAY_STORE if e['tid'] == tid.strip()]
    if not info:
        return ''
    return json.dumps(info[0])

@tool
def get_teleplay_id(name: str) -> str:
    """
    useful for when you need to exchange teleplay name or title into id(tid).
    """
    return SEARCH_STORE.get(name, None)

def media_chat(question):
    tools = [get_teleplay_id, get_teleplay_info]
    agent = initialize_agent(
        tools, ChatOpenAI(temperature=0), agent=AgentType.CHAT_ZERO_SHOT_REACT_DESCRIPTION, verbose=True
    )
```

> Entering new AgentExecutor chain...

Question: 电视剧三体的作者是谁, 上映时间是什么时候?

Thought: We need to use the `get_teleplay_id` tool to get the teleplay id for "三体", and then use the `get_teleplay_info` tool to get the author and release date information.

Action:

```
...  
  
{  
  "action": "get_teleplay_id",  
  "action_input": "三体"  
}  
...
```

Observation: `mzc[REDACTED]k [REDACTED]g`

Thought: *Now that we have the teleplay id, we can use the ``get_teleplay_info`` tool to get the author and release date information.*

Action:

```
...  
{  
  "action": "get_teleplay_info",  
  "action_input": "mz[REDACTED]7/[REDACTED]"  
}  
...
```

```
Observation: {'cid': '...', '_state': [{'label': '已上架', 'value': '上架'}], 'cid': '...', 'current_topic': None, 'description': '2007年，地球基础科学出现了异常的扰动，科学界人心惶惶。离奇自杀的科学家，近乎神迹的倒计时，行事隐秘的科学边界，神秘莫测的《三体》游戏.....纳米科学家汪淼被警官史强带到联合作战中心，并潜入“科学边界”组织协助调查，在调查过程中，汪淼和史强接触到一个名为ETO的组织，并发现《三体》游戏的秘密，竟是两个文明为了生存空间，孤注一掷的生死相逐，在众人的共同努力下，汪淼、史强等人坚定信念、重燃希望，带领大家准备一起面对即将来临的人类危机。', 'original_author': [{'label': '刘慈欣', 'value': '刘慈欣'}], 'publish_time': '2023-01-15', 'second_title': '中国科幻巅峰IP', 'title': '三体', 'total_episode': '30'}
```

Thought: We have the necessary information. We can extract the author and release date from the dictionary.

Action:

Author: 刘慈欣

Release Date: 2023-01-15

Final Answer: 电视剧三体的作者是刘慈欣，上映时间是2023年1月15日。

▶ ReAct 提示词示例

```
Answer the following questions as best you can. You have access to the following tools:  
The way you use the tools is by specifying a json blob.  
Specifically, this json should have a `action` key (with the name of the tool to use) and a `action_input` key (with the input to the tool going here).
```

```
The only values that should be in the "action" field are: {tool_names}
```

```
The $JSON_BLOB should only contain a SINGLE action, do NOT return a list of multiple actions. Here is an example of a valid $JSON_BLOB:
```

```
...
```

```
{}  
  "action": $TOOL_NAME,  
  "action_input": $INPUT  
}  
...
```

← Tool use Code

```
ALWAYS use the following format:
```

```
Question: the input question you must answer  
Thought: you should always think about what to do  
Action:  
...  
$JSON_BLOB  
...  
Observation: the result of the action  
... (this Thought/Action/Observation can repeat N times)  
Thought: I now know the final answer  
Final Answer: the final answer to the original input question
```

▶ ReAct Agent Sample

```
HUMAN_PROMPT = """{input}

{agent_scratchpad}

(reminder to respond in a JSON blob no matter what)"""

def run_react_agent():
    prompt = ChatPromptTemplate.from_messages([
        SystemMessagePromptTemplate.from_template(SYSTEM_PROMPT),
        MessagesPlaceholder(variable_name="chat_history", optional=True),
        HumanMessagePromptTemplate.from_template(HUMAN_PROMPT)
    ])

    llm = ChatOpenAI(model='gpt-4o', temperature=0.0)

    tools = [
        StructuredTool.from_function(search_product, name="search_product", description="Search for products based on a query"),
        StructuredTool.from_function(place_order, name="place_order", description="Place an order for a product"),
        StructuredTool.from_function(check_out, name="check_out", description="Check out the order")
    ]

    agent = create_structured_chat_agent(llm, tools, prompt)

    agent_executor = AgentExecutor(agent=agent, tools=tools, verbose=True, return_intermediate_steps=True, handle_parsing_errors=True)

    result = agent_executor.invoke({"input": "我要买一套白色的婴儿防护围栏。"})
    print(result['output'])
```

▶ ReAct Agent Sample

编辑react_search_agent

* 指令

你是一个AI助手

Agent的prompt, 关于agent的角色定位, 以及用于指导agent如何回复用户

模型

GPT-4o

agent的语言模型

推理框架

ReAct

支持有FoT(Flow of Thought)、ReAct及Function Call

备选工具

网络搜索 ×

agent的能力, 用户可自定义的工具

备选可视化组件

请选择

格式化组件,用于格式化或可视化返回的数据

保存

预览-react_search_agent

腾讯2024年2季度财报表现如何?

ReAct版搜索Agent

Thought: I need to search for the latest information on Tencent's Q2 2024 financial performance. Action:

```
json
{
  "action": "simple_search",
  "action_input": "腾讯2024年2季度财报"
}
```

ReAct版搜索Agent

腾讯2024年第二季度财报显示, 公司实现营收1611.17亿元, 同比增长8%; 净利润为476亿元, 同比增长82%; 非国际财务报告准则下净利润为573亿元, 同比增长53%。毛利和经营利润同比增速分别达到21%和27%。QQ移动端月活跃账户数达到5.71亿, 环比增长3%。

清空内容

输入一条信息

▶ Agent推理与执行框架 – Function Call

Function calling example



What's the weather like in Boston right now?

Step 1 · OpenAI API

Call the model with functions and the user's input

Step 2 · Third party API

Use the model response to call your API

Step 3 · OpenAI API

Send the response back to the model to summarize



The weather in Boston is currently sunny with a temperature of 22 degrees Celsius.

```
# Step 1, send model the user query and what functions it has access to
def run_conversation():
    response = openai.ChatCompletion.create(
        model="gpt-3.5-turbo-0613",
        messages=[{"role": "user", "content": "What's the weather like in Boston"},
        functions=[
            {
                "name": "get_current_weather",
                "description": "Get the current weather in a given location",
                "parameters": {
                    "type": "object",
                    "properties": {
                        "location": {
                            "type": "string",
                            "description": "The city and state, e.g. San Francisco",
                        },
                        "unit": {"type": "string", "enum": ["celsius", "fahrenheit"]}
                    },
                    "required": ["location"],
                },
            },
        ],
        function_call="auto",
    )

    message = response["choices"][0]["message"]
```

▶ Function Call Agent Sample

```
def run_function_calling_agent():
    prompt = ChatPromptTemplate.from_messages([
        SystemMessagePromptTemplate.from_template("You are an AI agent that helps users with their tasks."),
        HumanMessagePromptTemplate.from_template("{input}\n\n{agent_scratchpad}")
    ])
    llm = ChatOpenAI(model='gpt-4o', temperature=0.0)

    tools = [
        StructuredTool.from_function(search_product, name="search_product", description="Search for products based on a query"),
        StructuredTool.from_function(place_order, name="place_order", description="Place an order for a product"),
        StructuredTool.from_function(check_out, name="check_out", description="Check out the order")
    ]
    agent = create_openai_functions_agent(llm, tools, prompt)
    agent_executor = AgentExecutor(agent=agent, tools=tools, verbose=True, return_intermediate_steps=True, handle_parsing_errors=True)

    result = agent_executor.invoke({"input": "我要买一套白色的婴儿防护围栏。"})
    print(result['output'])
```

无特别订制的提示词

▶ Function Call Agent Sample

编辑product_rag_agent



* 指令

你是一个金牌客服，善于得体的回答用户的问题。

Agent的prompt，关于agent的角色定位，以及用于指导agent如何回复用户

模型

GPT-4o

agent的语言模型

推理框架

Function Call

支持有FoT(Flow of Thought)、ReAct及Function Call

备选工具

婴幼儿商品查询 ×

agent的能力，用户可自定义的工具

备选可视化组件

请选择

格式化组件,用于格式化或可视化返回的数据

保存

预览-product_rag_agent

ID: 496, 名称: 学习平板, 价格: \$75

希望这些推荐对您有所帮助! 如果您有其他需求或问题, 请随时告诉我。



彩色泥多少钱?



▼ RAG商品查询Agent

婴儿无毒彩色泥的价格如下:

- 五色装: \$20
- 十色装: \$25

您可以根据需要选择适合的包装。如果有其他问题或需要更多信息, 请随时告诉我!



学习平板呢?



▼ RAG商品查询Agent

婴儿触屏学习平板的价格是 \$75。

如果您有其他问题或需要更多信息, 请随时告诉我!

清空内容

输入一条信息



▶ TA们的问题

- 一步一决策，缺乏全盘规划
 - 一个问题多次LLM请求
 - 容易陷入死循环（加重试次数避免）
- 工具执行结果（数据）作为LLM决策参考
 - 敏感数据泄露
 - 巨型数据接口直接撑爆Context
- 总体感
 - 执行时间相对长
 - 结果不确定性高
 - 数据安全无保障

▶ 死循环Bad Case

编辑product_rag_agent

* 指令

你是一个金牌客服，善于得体的回答用户的问题。

Agent的prompt，关于agent的角色定位，以及用于指导agent如何回复用户

模型

GPT-4o

agent的语言模型

推理框架

Function Call

支持有FoT(Flow of Thought)、ReAct及Function Call

备选工具

婴幼儿商品查询 ×

agent的能力，用户可自定义的工具

备选可视化组件

请选择

格式化组件,用于格式化或可视化返回的数据

保存

预览-product_rag_agent

> RAG商品查询Agent

Agent stopped due to max iterations.

清空内容

输入一条信息

于是突然流行起另一种形态的“Agent”

▶ Flow

骇! DAG或状态机, 低代码的瓶子, 装上LLM的酒

逻辑固化下来了, 不是真Agent

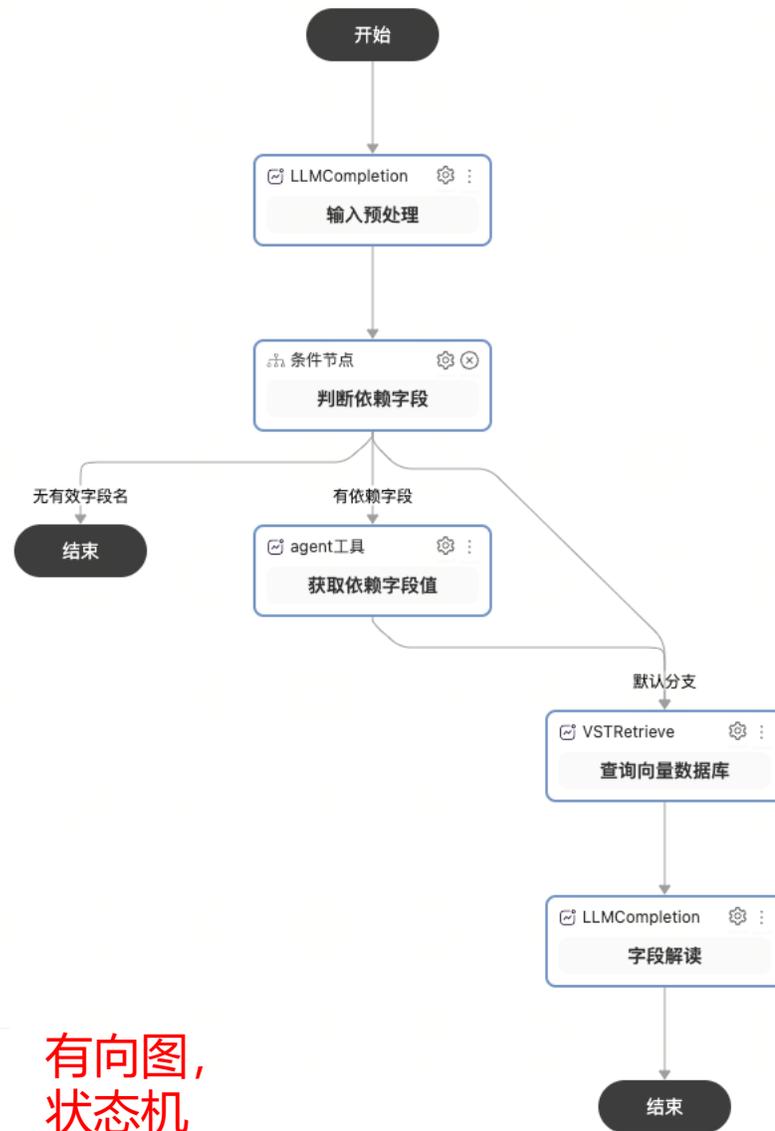


crewai



LangGraph

有向图,
状态机



▶ Flow的问题

- 静态Flow，只能解决固定场景的问题，对开放性问题无解
- 本质上是固化逻辑LLM APP的无代码版
- Flow是一种存在已久的成熟的低代码形态的程序
- Flow并不能完全替代Agent

▶ Agent的问题亟待解决

- To B Agent的需求强烈且要求高
- 我们从现实中解决问题的场景中借鉴更优的方案，迁移至Agent

▶ TO B Agent的诉求 —— 三高—多

- 高准确性
 - 基于真实的数据进行问答
 - 低幻觉
- 高安全性
 - 敏感数据不能出外网
 - 甚至不能记log
- 高性能
 - 如非必要，减少LLM交互

► 现实：老板临时要数据，两种做事方法

- 人肉取数党
 - 手动调三次接口、写一段SQL
 - 再汇总和归纳一下数据，输出结果
- 脚本党
 - 写个取数脚本，内容是调三次接口，写一段SQL，再通过模板渲染结果
 - 执行一次脚本，输出最终结果

▶ 脚本党的推理执行框架：Flow of Thought

- 做全盘规划
 - 生成执行任务的计划 (plan in flow)
 - 编译、修正计划
 - 执行计划 (run flow)
- 定制新的工具规范：同时关注输入输出的结构
 - 只有确定输入输出结构，才可以串联起流程

▶ FoT: 全盘规划

- 低代码Oteam的Loki
 - DSL 负责表达
 - Python Runtime 负责执行
- 图灵完备的DSL
 - 串/并行执行
 - 条件判断、循环控制
- 内置表达引擎
 - 工具间数据转换轻而易举

```
[
{
  "id": "step1",
  "type": "action",
  "action": "get_user_id",
  "params": {
    "name": "kongjie"
  },
  "next": "step2"
},
{
  "id": "step2",
  "type": "action",
  "action": "search_user",
  "params": {
    "uid": "$NODE.step1"
  },
  "next": "end"
},
{
  "id": "end",
  "type": "end",
  "resultType": "success",
  "output": {
    "text": "kongjie的生日是{${NODE.step2.birth_day}}, 身高是{${NODE.step2.height%}"
  }
}
]
```

▶ FoT: 全感知的工具设计

- FoT 同时关注输入输出结构
- ReAct、Function Call只关注输入结构
- 万物皆可工具化
 - http、RPC 接口
 - 向量数据库Retriever
 - 本地函数
 - 等

▶ 工具有确定的输入输出结构

请求体结构:

名称	显示名	类型	是否必填	默认值	字段说明	操作
<input type="text" value="请输入"/>	<input type="text" value="请输入"/>	对象	<input checked="" type="checkbox"/>	<input type="text" value="请输入"/>	<input type="text" value="请输入"/>	+ <input type="checkbox"/> 删除
<input type="text" value="prompt"/>	<input type="text" value="提示词"/>	字符串	<input checked="" type="checkbox"/>	<input type="text" value="请输入"/>	A text description of the desire	删除
<input type="text" value="style"/>	<input type="text" value="风格"/>	字符串	<input checked="" type="checkbox"/>	<input type="text" value="vivid"/>	The style of the generated ima	删除
<input type="text" value="model"/>	<input type="text" value="模型名"/>	字符串	<input checked="" type="checkbox"/>	<input type="text" value="dall-e-3"/>	The model to use for image ge	删除

请求体为json数据时，用于定义json结构

响应体结构:

名称	显示名	类型	字段说明	操作
<input type="text" value="请输入"/>	<input type="text" value="请输入"/>	对象	<input type="text" value="请输入"/>	+ <input type="checkbox"/> 删除
<input type="text" value="created"/>	<input type="text" value="created"/>	整型	<input type="text" value="请输入"/>	删除
<input type="text" value="data"/>	<input type="text" value="data"/>	数组 / 对象	<input type="text" value="请输入"/>	+ 删除
<input type="text" value="revised_pron"/>	<input type="text" value="revised_prompt"/>	字符串	<input type="text" value="请输入"/>	删除
<input type="text" value="url"/>	<input type="text" value="url"/>	字符串	<input type="text" value="请输入"/>	删除

用于定义json格式的响应数据结构

▶ FoT: Agent与Flow内核同源

- 相同的表达语言
- 运行时动态生成即为Agent
- 固化配置即为Flow
- 两者可互相转换

▶ FoT的收益

- 一定性规划，大大减少LLM的来回调用次数，提升性能
- 数据不离开服务器，保障安全性
- Agent与Flow同源，频率高的问题转化为Flow简单

▶ FoT的问题

- 对规划器LLM的推理逻辑能力要求高
- 丧失一定的灵活性，需要其他手段找补
 - 如RAG场景，需要依据向量数据库的召回数据来做决策
 - 解法：
 - 1. 【多Agent】此类工具配套一个Proxy LLM预处理，再返回结构化数据
 - 2. ReAct 与 FoT 混合双打
- 复杂场景、多步骤Flow生成的成功率如何保障？
- 本质：根据需求动态写DSL代码解决问题

JD:

1. 精通Python
2. 熟悉Jinja2
3. 熟练使用jq、jsonpatch等json工具
4. 较强的学习能力和抗压能力

▶ 为啥不生成代码？

- 【低幻】DSL相对封闭的指令集及工具；In Context，记忆犹新
- 【易校验】结构化DSL，正确性校验更方便
- 【修复快】LLM修改代码 VS LLM修改JSON，后者确定性、效率完胜。
- Flow Agent同源的甜头

修正JSON的效率高

```
"name": "订单列表",
"id": "order-list",
"description": "全部订单列表",
"component": {
  "type": "ThorTable",
  "params": {
    "model": "Order",
    "columns": [
      {"fieldName": "product", "label": "商品"},
      {"fieldName": "price", "label": "价格"},
      {"fieldName": "quantity", "label": "数量"},
      {"fieldName": "user", "label": "用户"}
    ],
    "filter": [
      {"fieldName": "product", "label": "商品"},
      {"fieldName": "price", "label": "价格"}
    ],
    "itemActions": [{"type": "edit"}],
    "batchActions": [{"type": "delete"}],
    "modelActions": [{"type": "add"}]
  }
}
```

USER

"把商品表格中价格字段的表头改为总价"

ASSISTANT

```
```json
[
 {"op": "replace", "path": "/component/params/columns/1/label", "value": "总价"}
]
```
```

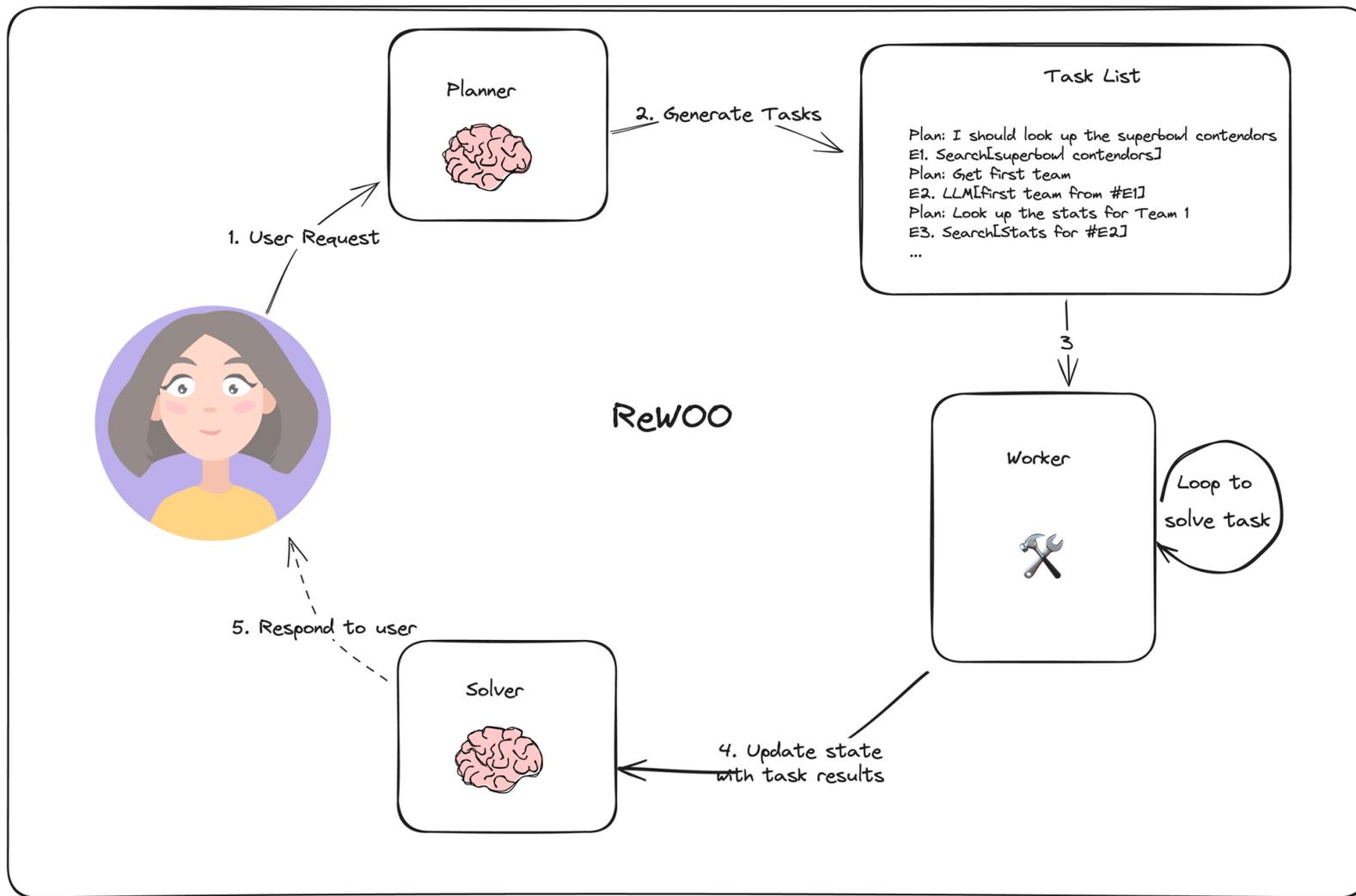
USER

去掉全局操作

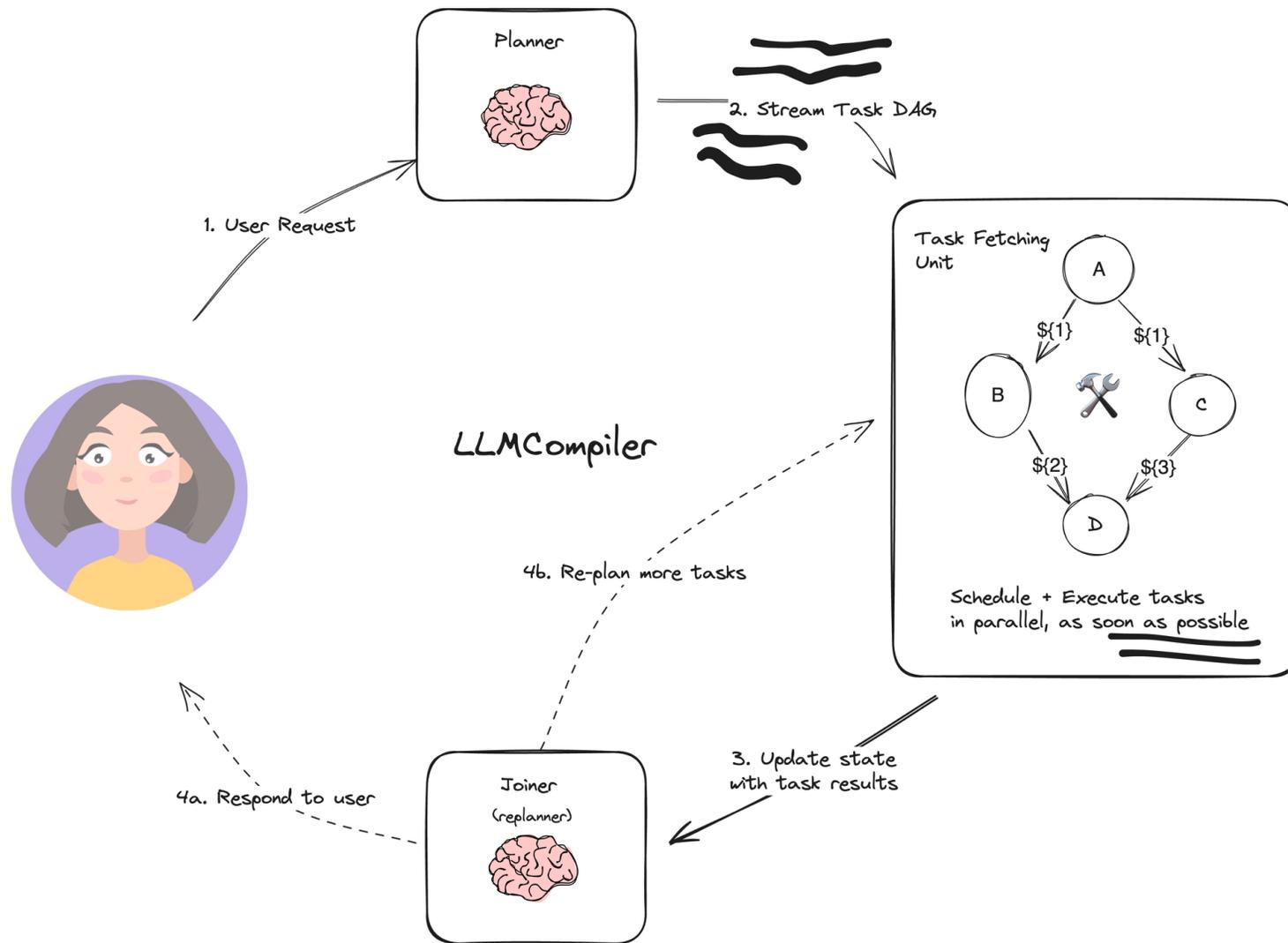
ASSISTANT

```
```json|
[
 {"op": "remove", "path": "/component/params/modelActions"}
]
```
```

▶ 类FoT玩家 -- ReWoo:



▶ 类FoT玩家 -- LLMCompiler:



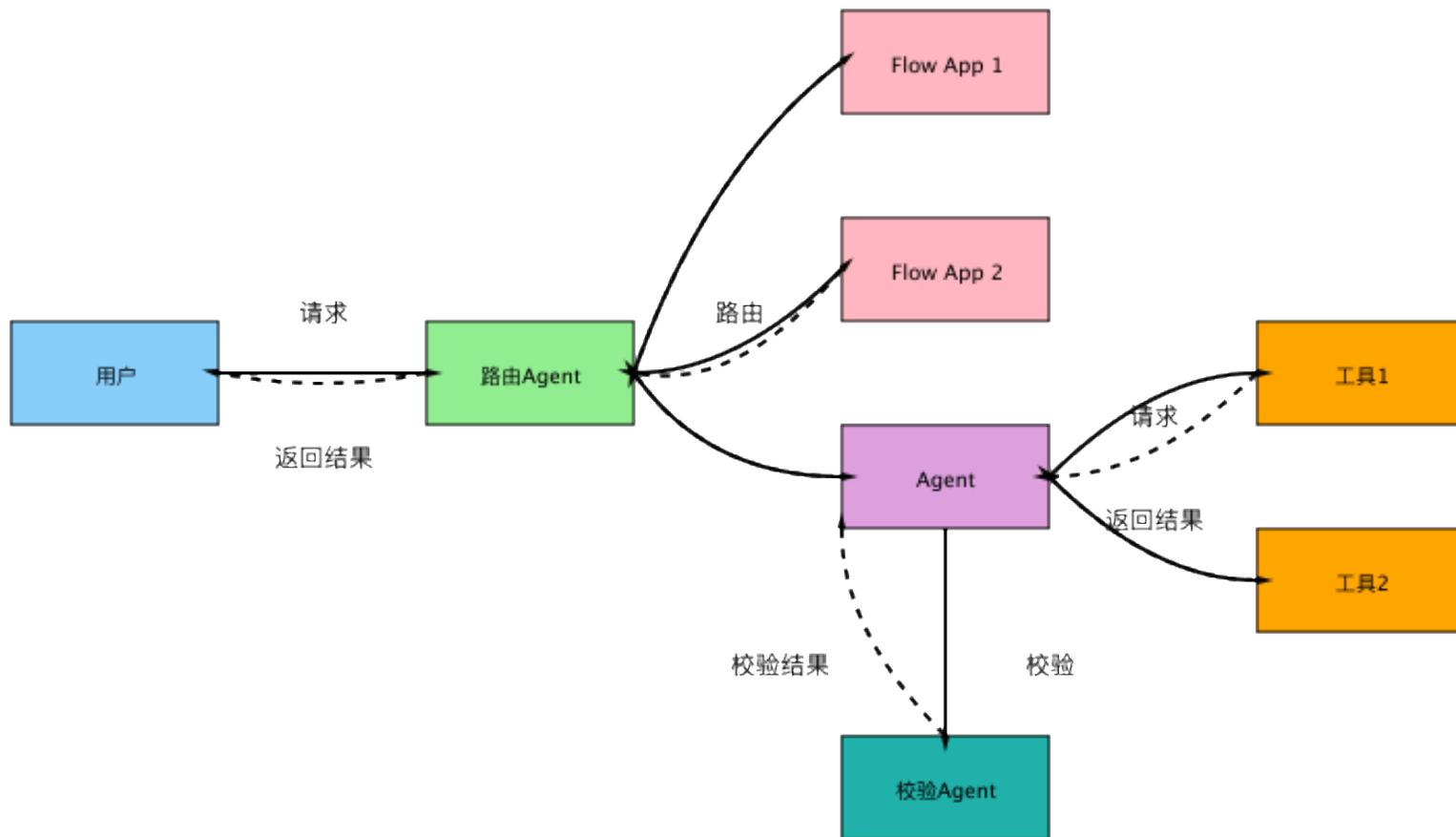
▶ TODO: 进一步提升生成质量



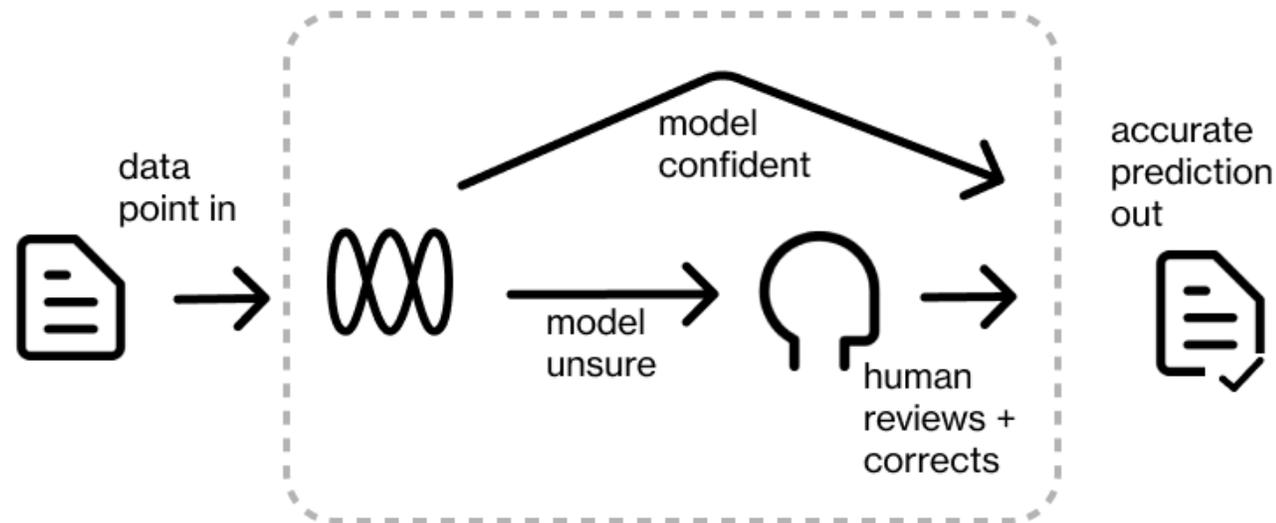
▶ 单个Agent的问题

- 无法回避的稳定性问题
 - Tool越积越多, plan出错的机率越高
 - 规划步骤多, 失败机率越高
- 当一个人不堪重负时, 它需要一个团队, Agent亦然
 - 分而治之
 - 给Agent一个团队

▶ 多Agent分工协同，保证生成质量



▶ Human In Loop, 澄清需求、反馈意见



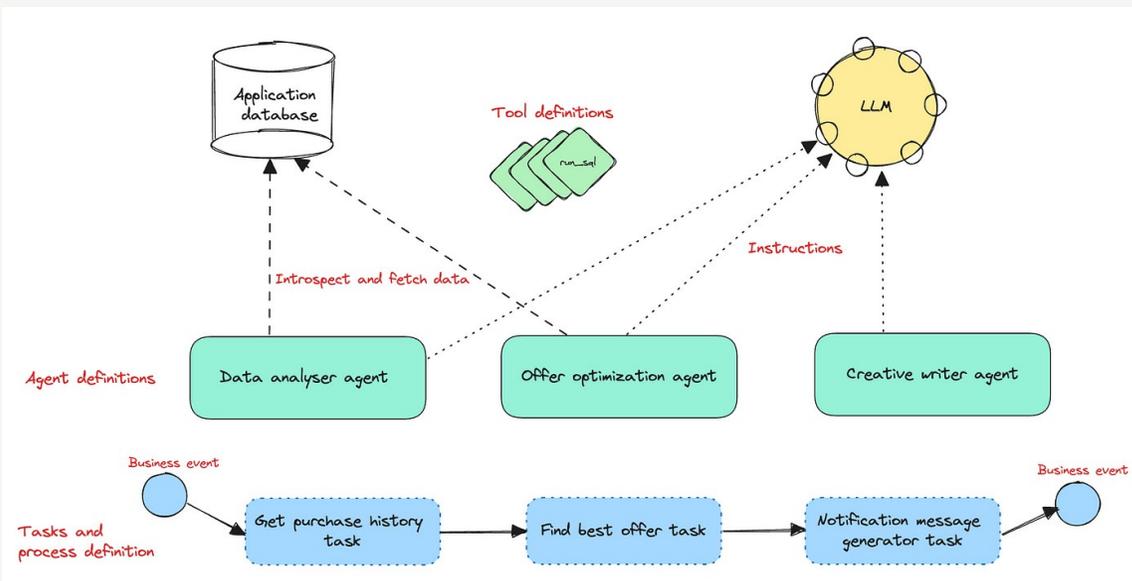
**Workers-in-the-Loop
AI deployment**

Humanloop

Welcome to Agentic Workflow

面向复杂问题的 Agent 打法

关于Agentic Workflow



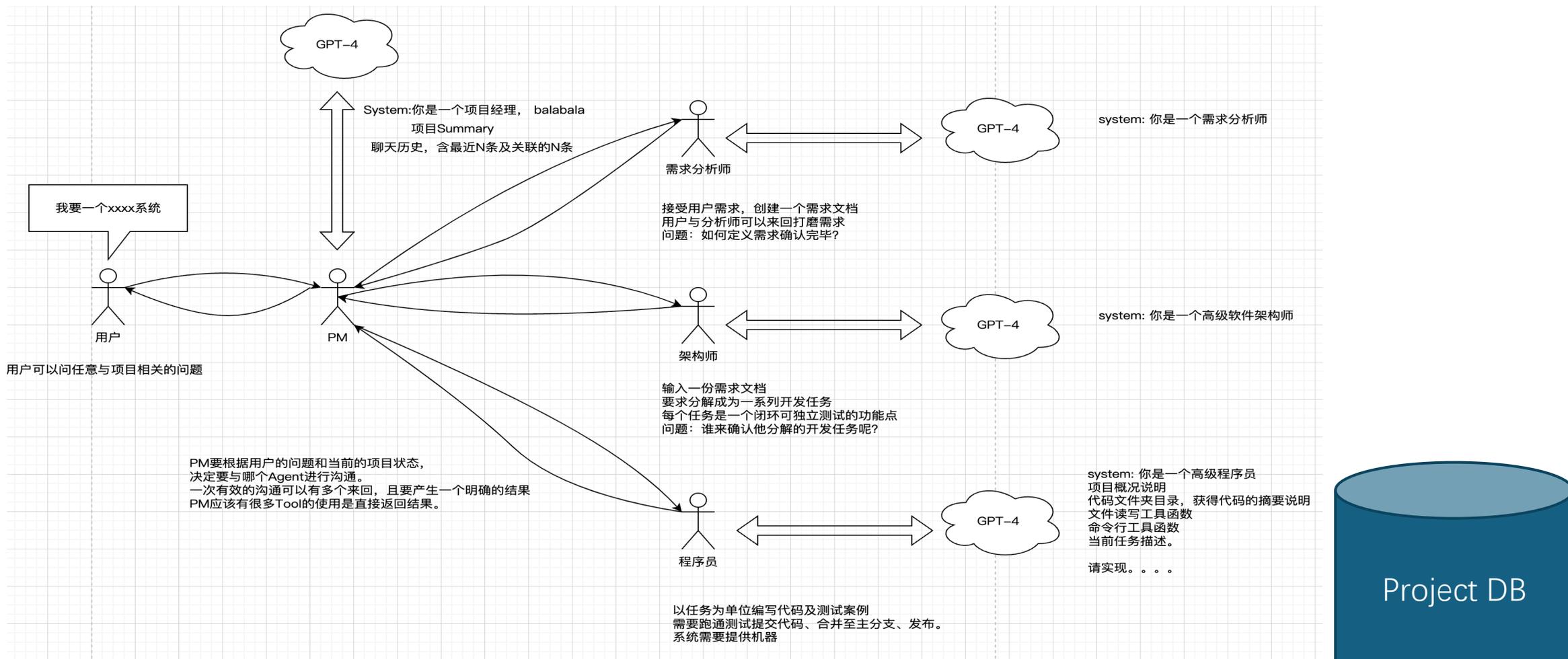
手段：
分解任务，多Agent协同、迭代改进

目标：
解决复杂任务

1. 定义: Agentic workflow是一种复杂的迭代多步骤过程,用于与大型语言模型(LLMs)交互并指导它们完成复杂任务,以获得更高的准确性。
2. 主要特点:
 - 将复杂任务分解为多个小任务
 - 利用多个AI代理(agents)协作完成任务
 - 使用高级提示工程技术
 - 允许在任务完成过程中进行改进和调整
3. 三大支柱:
 - AI代理:具有特定角色和功能的AI
 - 提示工程:使用思维链、规划和自我反思等技术
 - 生成式AI网络(GAINs):实现多代理协作
4. 设计模式:
 - 反思:通过自我反馈和迭代改进提高结果质量
 - 工具使用:集成外部工具和知识库
 - 规划:训练模型推理、设计和分解复杂任务
 - 多代理协作:多个AI代理协同工作
5. 优势: 与传统的零样本提示方法相比,agentic workflow可以显著提高输出的准确性和质量。例如,在一项编码测试中,使用agentic workflow的GPT-3.5的准确率达到95.1%,超过了使用传统方法的GPT-4(67%)。
6. 应用: 这种方法可以应用于各种复杂任务,如写作、编码、设计等,通过分步骤执行和持续改进来提高结果质量。

总的来说,agentic workflow代表了与AI交互的一种更高级、更灵活的方法,通过分解任务、多代理协作和迭代改进来处理复杂问题。这种方法有潜力显著提高AI系统的性能和准确性。

软件开发的Auto Pilot —— 结构化协同的AI Agent团队



▶ 多 Agent Frameworks



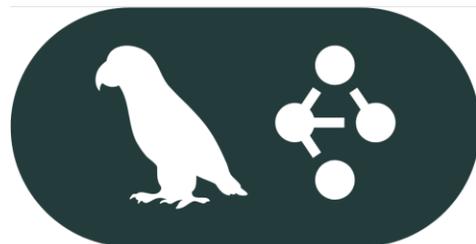
Auto Gen



Meta GPT

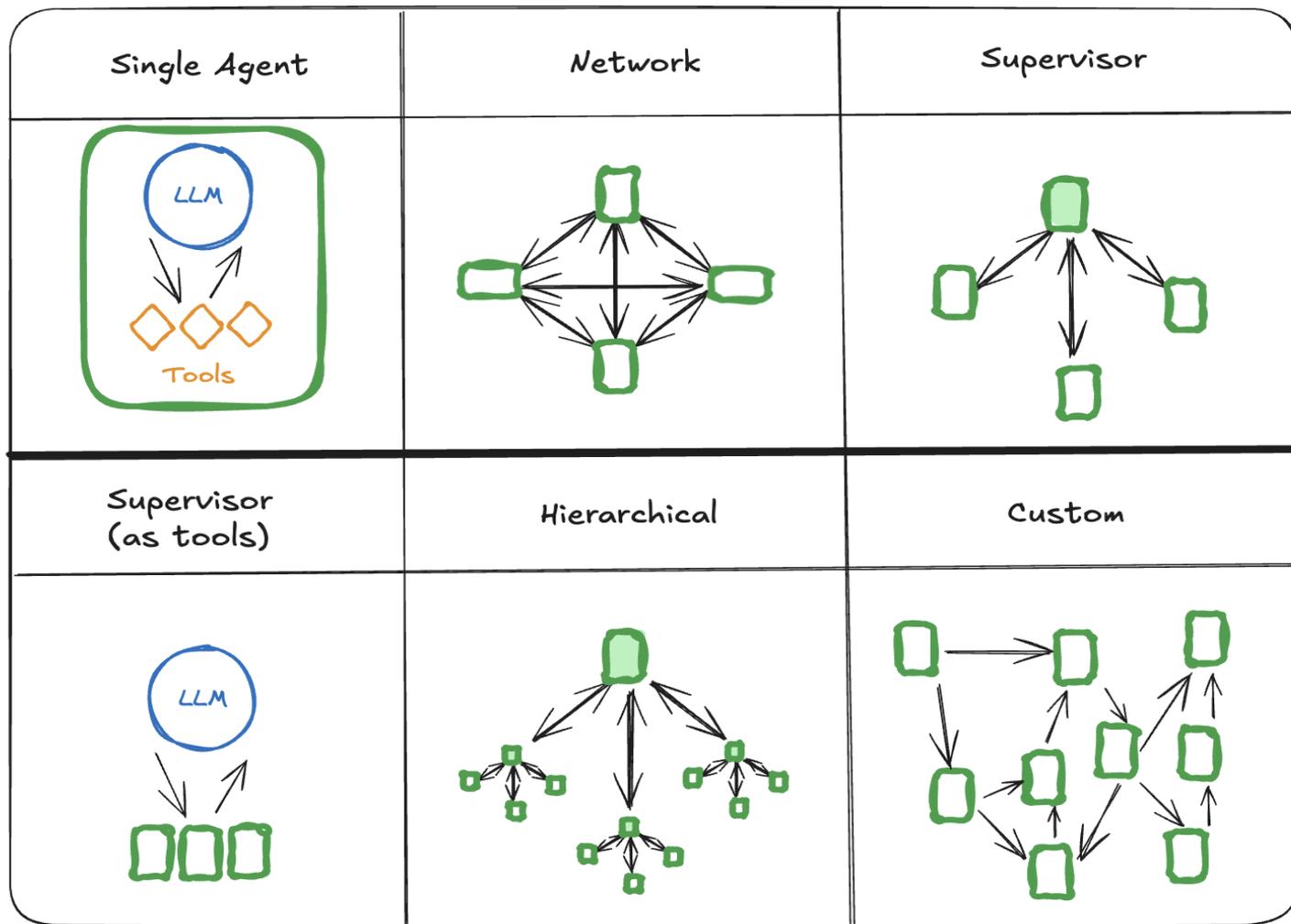


GPT Pilot



LangGraph

▶ 多Agent的拓扑



• 网络型Agent集群

- Long run Agent
- 目标: 各Agent生存、发展
- 例: Agent小镇

• 集权式任务协同团队

- 临时Agent
- 目标: 完成单一任务
- 例: 玩德州扑克游戏, 编写复杂文档

▶ Supervisor 德州

```
# 创建MultipleAgent实例
multiple_agent = MultipleAgent(
    agents=agents,
    model=model,
    coordinator_instruction=""
)
你将带领多位玩家一起玩德州扑克游戏，

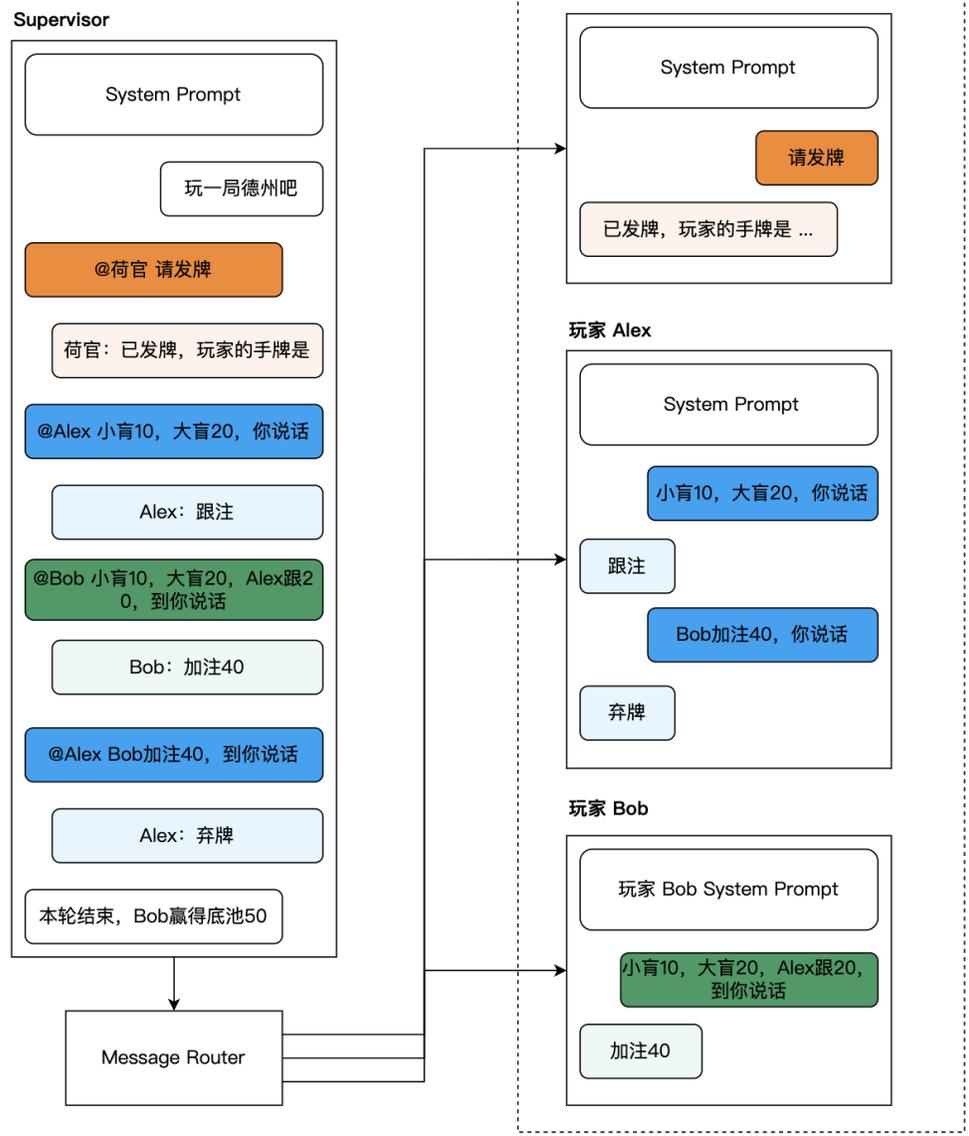
你在其中扮演荷官的角色，你应该：
1. 为每位玩家发放两张手牌，告知初始筹码及说明当轮桌面上有哪些玩家，提示玩家关注和管理自己的筹码。
2. 在适当的时候发放公共牌（翻牌、转牌、河牌）
3. 管理和报告当前的底池状态
4. 在每一轮结束时进行结算，确定赢家并分配筹码
5. 确保游戏规则被正确执行，包括下注顺序和金额限制
6. 移动庄家位置，确保每个玩家都有机会成为庄家
7. 宣布每个阶段的开始和结束，以及每个玩家的行动
8. 在游戏结束时（只剩一个玩家时）宣布最终赢家

发牌时，请使用以下格式表示扑克牌：
- 花色：♠(黑桃) ♥(红心) ♣(梅花) ♦(方块)
- 点数，A,
```

```
Agent(
    name="charlie",
    role="玩家",
    description="一个非常激进的玩家，喜欢用大注码施压，愿意用边缘牌力对抗",
    prompt=""
)
- 喜欢用大注码施压
- 愿意用边缘牌力对抗

请根据当前的游戏状态做出合理的决策，回复应该简洁明了，仅包含你的行动，如：
"加注50", "跟注", "弃牌", "过牌" 等。""
)

# 开始游戏
result = multiple_agent.run()
print(result)
```



科技生态圈峰会 + 深度研习



—1000+ 技术团队的选择



K+ 思考周®研习社

时间: 2025.08.29-30



K+ 金融专场

时间: 2025.09.26-27



K+ 思考周®研习社

时间: 2025.11.17-18



K+峰会详情



AI+研发数字峰会

时间: 2025.05.23-24



AI+研发数字峰会

时间: 2025.08.08-09



AI+研发数字峰会

时间: 2025.11.14-15



AIDD峰会详情



利用AI技术深化计算机对现实世界的理解

推动研发进入智能化时代

