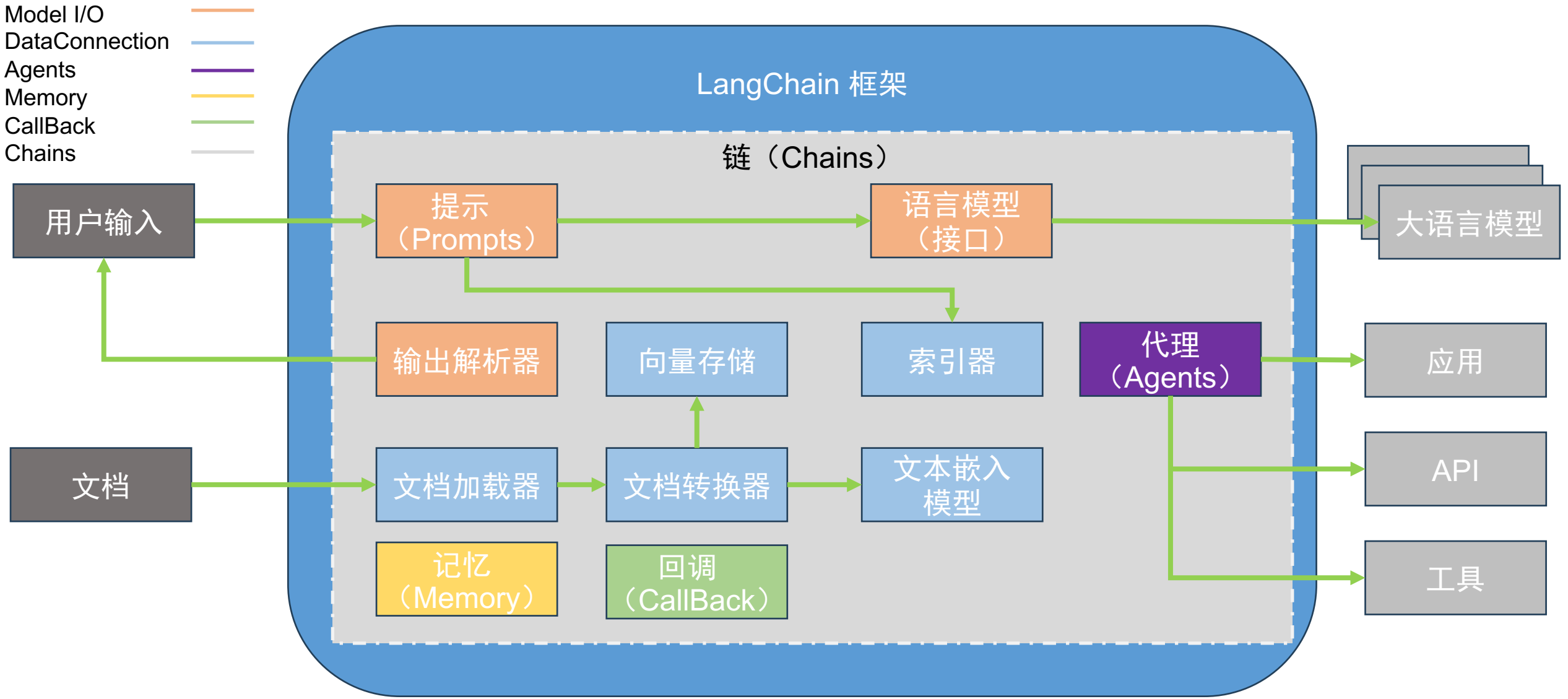
7-1 Agents 组件概述

- 回顾组件地图
- Agents 基本概念

LangChain 6 大组件回顾



Agents的基本概念

基本定义

代理使用来扩展LLM能力的， LLM能力本身有局限性，它是针对大量语料进行训练的结果。

LLM 存在如下短板：

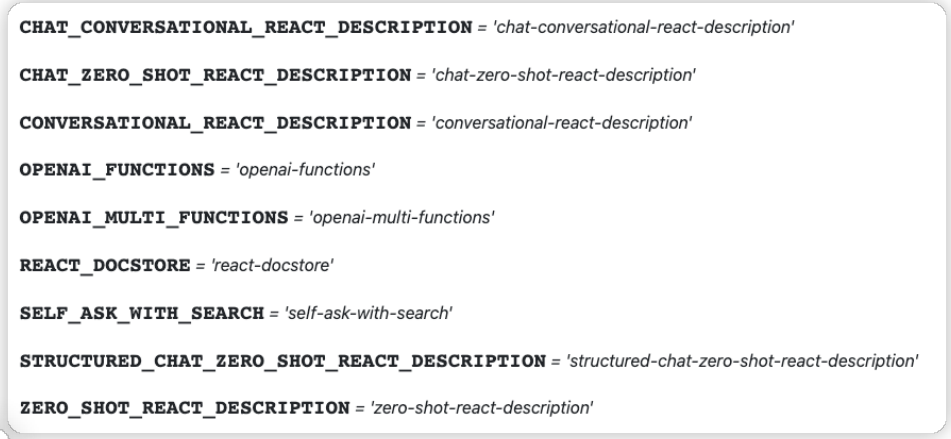
1. 缺乏获取当前信息的能力：预训练的语言模型无法跟上动态信息的变化，例如汇率、当前的COVID病例数、股票价格，甚至是当前的日期。

2. 缺乏获取专有信息源的能力：这些模型无法获取到专有信息，例如公司的客户名单或在线游戏的状态。

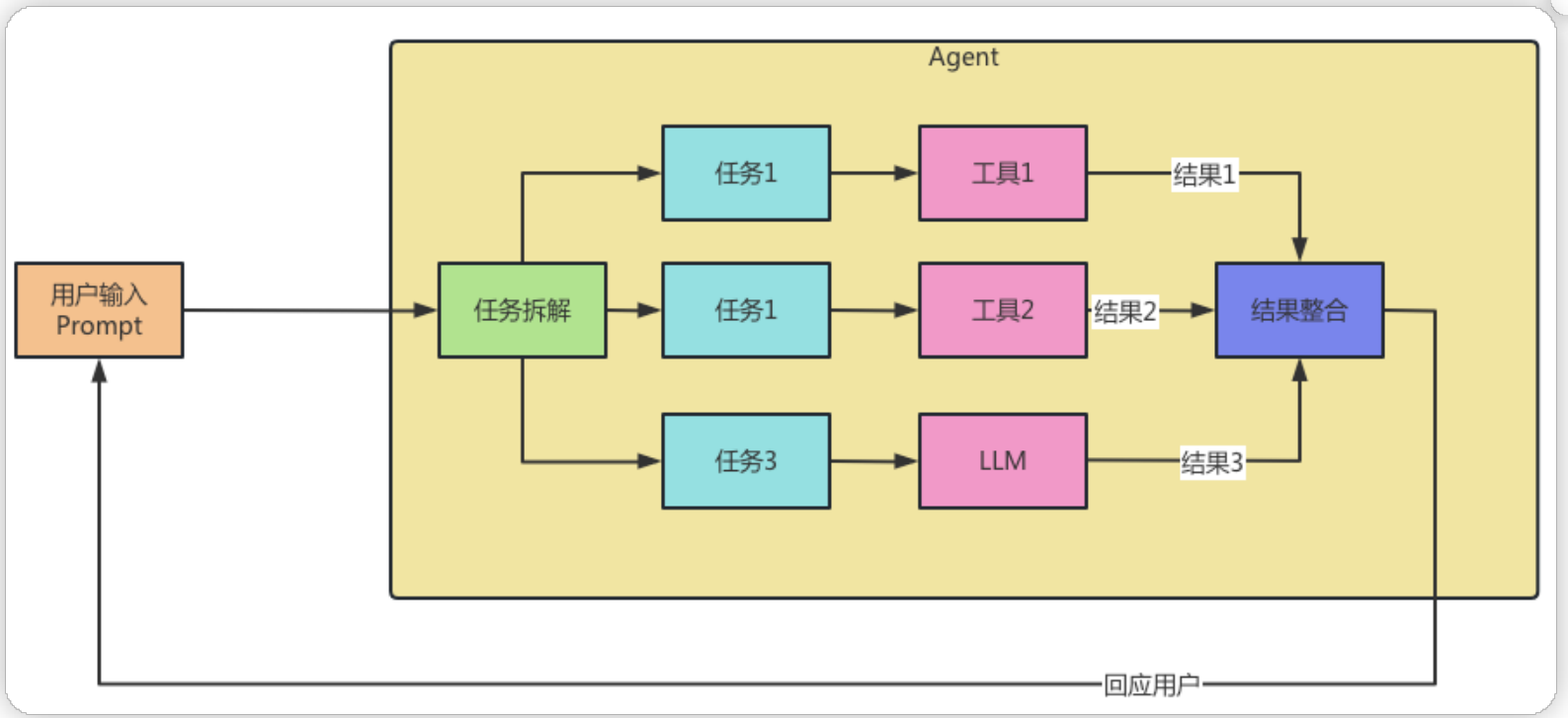
3. 缺乏推理能力：某些推理超出了神经方法的范围，需要专门的推理过程。例如，虽然语言模型在两位数的加法上表现良好，但在更复杂的计算上却遇到困难。

Agents的基本概念

- 代理（Agent）：将请求拆解成任务，调用工具完成回应。LangChain提供了不同的代理类型。
- 工具（Tools）：工具是代理调用的函数。LangChain提供了一套广泛的工具供你开始使用，同时也使自定义你的工具（包括自定义描述）变得简单。



```
CHAT_CONVERSATIONAL_REACT_DESCRIPTION = 'chat-conversational-react-description'
CHAT_ZERO_SHOT_REACT_DESCRIPTION = 'chat-zero-shot-react-description'
CONVERSATIONAL_REACT_DESCRIPTION = 'conversational-react-description'
OPENAI_FUNCTIONS = 'openai-functions'
OPENAI_MULTI_FUNCTIONS = 'openai-multi-functions'
REACT_DOCSTORE = 'react-docstore'
SELF_ASK_WITH_SEARCH = 'self-ask-with-search'
STRUCTURED_CHAT_ZERO_SHOT_REACT_DESCRIPTION = 'structured-chat-zero-shot-react-description'
ZERO_SHOT_REACT_DESCRIPTION = 'zero-shot-react-description'
```



- 工具箱（Toolkits）：工具的合集。
- 代理执行器（AgentExecutor）：代理执行器是代理的运行时环境。它实际调用代理并执行其选择的操作。代理执行器处理各种复杂情况，例如代理选择了不存在的工具、工具出错。

7-2 Agents 类型分类

- Conversational
    - 使用专门为对话优化的代理。其他代理通常优化为使用工具来找出最佳回应 使用一种特殊类型的代理（conversational-react-description），该代理预计会配合一个记忆组件来使用。
- OpenAI Multi Functions Agent
    - OpenAI的多功能代理（OpenAI Multi Functions Agent）来处理用户的提问
- Self ask with search
    - 用自问自答的方式拆解问题

7-3 Tool 应用

- Multi-Input Tools（自定义工具）

自定义一个乘法工具，然后在agent中调用。通过这个例子，大家可以定义自己的工具，可以是读取文件，也可以调用其他的API或者接口

Tool Input Schema

- 定义校验工具输入的一个模式或规范。

安装了一个Python包tldextract，然后使用它来验证URL的域名是否在许可的域名列表中。它使用ToolInputSchema类来定义并验证工具输入的格式。这个类有一个url字段，并有一个根验证器（@root_validator）用于检查URL的域名是否在_APPROVED_DOMAINS列表中。如果不在，它将抛出一个错误。

- Human-in-the-loop Tool Validation(人为参与验证)

使用ShellTool：首先，该文本介绍了ShellTool的使用方法。ShellTool在这里被用来执行echo Hello World!命令，并输出Hello World!。

添加人工审批：然后，文本介绍了如何为工具添加人工审批的方法。通过为工具添加默认的HumanApprovalCallbackHandler，可以确保用户必须手动批准输入到工具的每个输入，然后才会实际执行命令。在这里，它被用来在执行ls /usr命令之前进行人工审批。

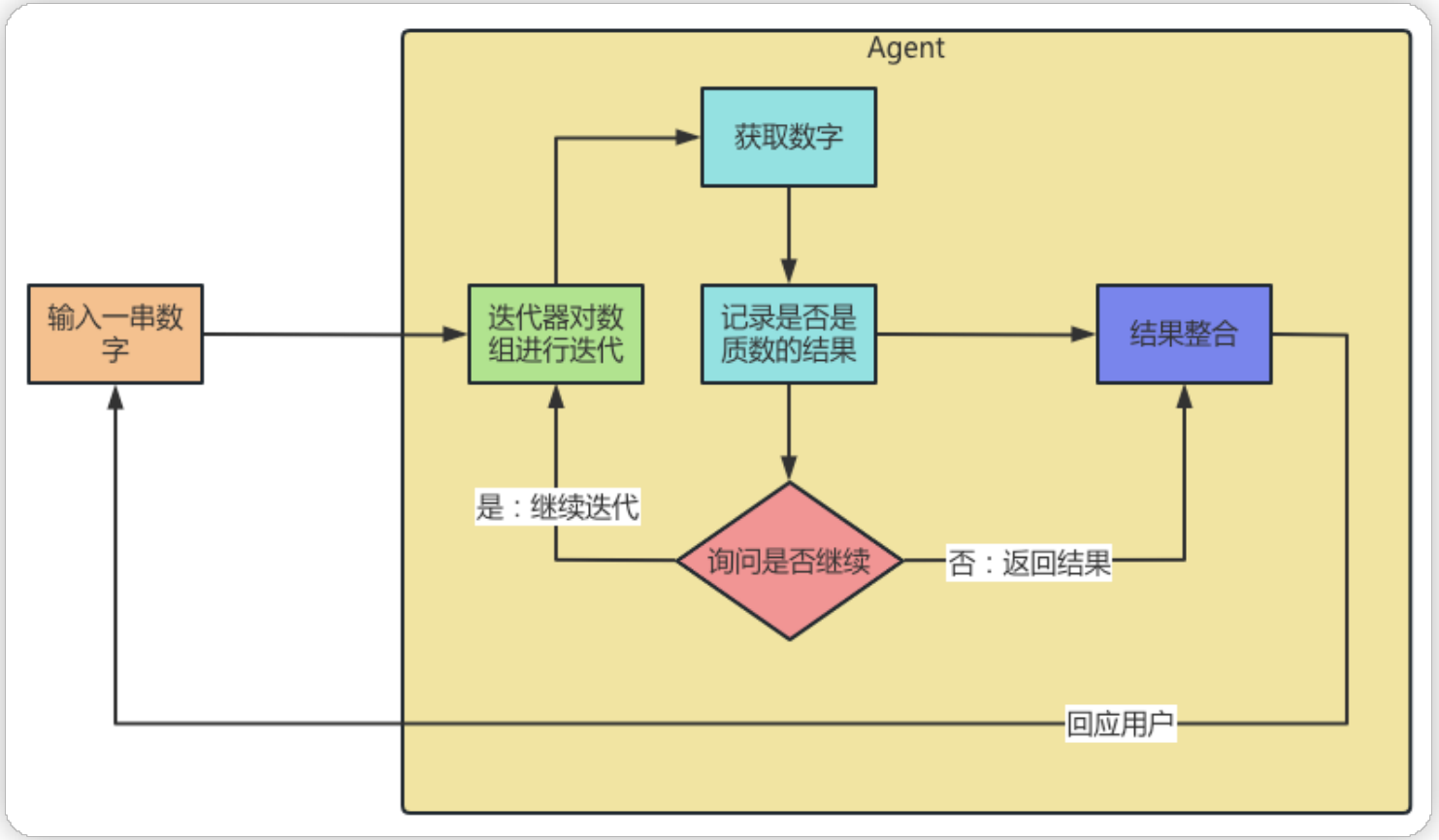
7-4 Agent 常规应用

- Agent 迭代器

表示将问题的处理过程分成多个步骤，每次循环处理一个步骤。

- Agent 与向量数据库协同

结合代理和向量存储。如果你已经将数据导入到向量存储，然后希望以代理方式与其进行交互，推荐的方法是创建一个 RetrievalQA，并在总代理中使用它作为工具。



7-5 Agent 常规应用

- Async API

利用 asyncio 库为Agents提供异步支持

- 自定义Agent

继承 `BaseMultiActionAgent`类，重写`Plan`方法对`Agent`的执行步骤进行控制。

创建自定义代理的指导。一个代理主要由两部分组成：工具和代理类本身。工具是代理可以使用的工具，代理类则决定采取何种行动。下面代码介绍了如何创建一个能预测/执行多步骤的自定义代理。