第一章 大模型智能体（LLM Agent）基础入门

教学目标

1. 什么是大模型?
2. 什么是智能体?
3. 智能体架构
4. 提示词工程
5. 智能体相关技术
6. 单智能体和多智能体
7. 智能体开发技术框架
8. 大模型厂商的智能体平台

1、什么是大模型?

大模型对应的英文是Large Language Model（LLM），即大语言模型，简称大模型。技术层面讲，大模型是一种基于深度学习技术的机器学习模型。

为什么叫大模型呢？它是相对于小模型而言的。传统的机器学习算法一般是解决某个特定领域的问题（例如文本分类），使用的训练数据集规模较小，参数也比较少。而大模型一般是基于互联网上的海量数据训练而成的，模型参数可达数十亿至数万亿。这些参数就像大脑中的神经元连接，数量越多，模型能学习和掌握的细节就越丰富，解决问题的能力也就越强。

训练大模型的过程，类似于一个不断学习和积累经验的过程。它需要喂给模型海量的数据，比如文本文档、图像、语音记录等，通过复杂的算法让模型自己找出数据中的规律和模式。这样一来，当面临新的问题或数据时，大模型就能基于已学习到的知识做出高质量的预测或生成相应的输出，比如精准回答问题、创作文字、识别图像内容等。

大模型的威力在于，它不仅仅局限于某一特定任务，而是具有一定的通用性，能够在多个领域展现出色的表现，比如自然语言处理、图像识别、语音识别等。著名的例子如GPT系列，它们能够进行智能对话、文本生成，甚至展现出一定的创造性思维。

大模型不仅能够处理不同类型的任务，而且支持多种数据格式，例如文本、语音、图片、视频，这就是所谓的"**多模态**"。

当然，大模型的"大"也意味着它有"大胃口"，训练这样的模型需要极强的计算能力和大量的时间，同时伴随高昂的成本。除了GPU资源，训练一次大模型往往需要几周到几个月的时间。所以，除了大公司以及不差钱的创业公司，普通人是很难自己训练一个大模型的。

如果将这一轮ChatGPT引领的AI革命与移动互联网浪潮类比的话，大模型的角色类似于iOS或Andriod系统。对于普通人来说，我们只需要使用大公司训练好的大模型即可。

基础大模型训练好（称为预训练）以后，针对特定领域的任务，还需要进行微调，以便模型达到最佳的性能。所谓微调，就是将少量的经人工标注的高质量数据集喂给大模型，从而得到一个更适合解决特定任务的精细化模型。

微调是一个二次训练的过程，它所需要的算力和成本远低于预训练过程。但对于大多数个人开发者，门槛依然比较高。

目前国内外主要的大模型厂商及产品如下：

- OpenAI：GPT系列, 最新的是GPT-4o，ChatGPT基于GPT-3.5
- Anthropic：Claude
- Google：Gemini
- Meta：LLaMA
- Microsoft：与OpenAI合作，也推出了自己的大模型，如Phi-3
- 百度：文心一言
- 阿里巴巴：通义千问
- 腾讯：混元大模型
- 字节：豆包

此外，还有一众创业公司，例如前段时间火了一把的月之暗面Kimi。

2、什么是智能体?

大模型可以帮我们做很多事情，例如回答问题、写周报&文档、内容总结、翻译等。但普通人和大模型直接交互是不现实的，类GPT聊天机器人是大模型面向普通用户提供的一种接口，能够帮忙人们完成很多任务，但大模型能做的远不止于此，而聊天是一种泛化的场景，很多时候解决问题的效率不是很高。因此，这就需要“智能体”登场了。

智能体（Agent）是人工智能领域中的一个核心概念，指的是具有智能的实体，能够感知其环境、通过学习和推理改变自身状态，并采取行动以实现特定目标。智能体既可以是物理实体，如机器人，也可以是虚拟实体，如软件程序。它们能够自主活动并与环境交互，具备驻留性、反应性、社会性、主动性等特征，构成了一种能够持续自主发挥作用的计算实体。

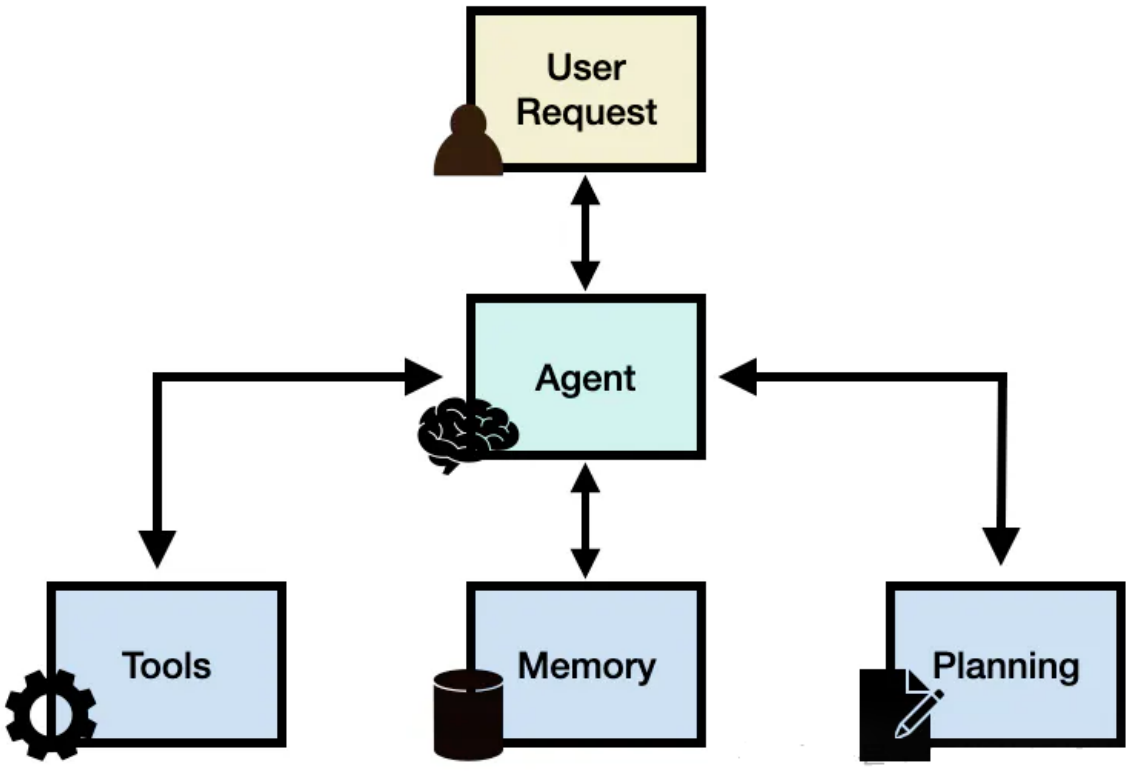
想象一下科幻电影里的场景，一个机器人边工作边和人类对话，可以根据人类指令完成任务，还可以独自开飞船，维修设备，甚至照看小孩等。可以将这种智能机器人视为一种智能体。

基于**大模型的智能体**是一种具有**自我管理、自我学习、自我适应、自我决策**能力的机器人或软件。它可以在没有人工参与的情况下工作，这与传统的自动化程序是不同的。自动化程序是将固定的流程自动执行，假如其中某个依赖项不可用时，自动化程序一般会失败。智能体能够感知环境，自我学习和自我决策，能够创造性地解决问题。

大语言模型（LLM）智能体，是一种利用大语言模型进行复杂任务执行的应用。如果将大模型（LLM）比作底层操作系统（如iOS和Andriod）的话，那么智能体（Agent）就是上层App。这正是工程技术人员擅长的领域。

3、智能体架构

构建大语言模型智能体（LLM Agent）时，LLM充当着控制中心或“大脑”的角色，负责管理完成任务或响应用户请求所需的一系列操作。智能体Agent通过调用大模型的能力完成任务，并需要依赖于规划、记忆以及工具使用等关键模块。

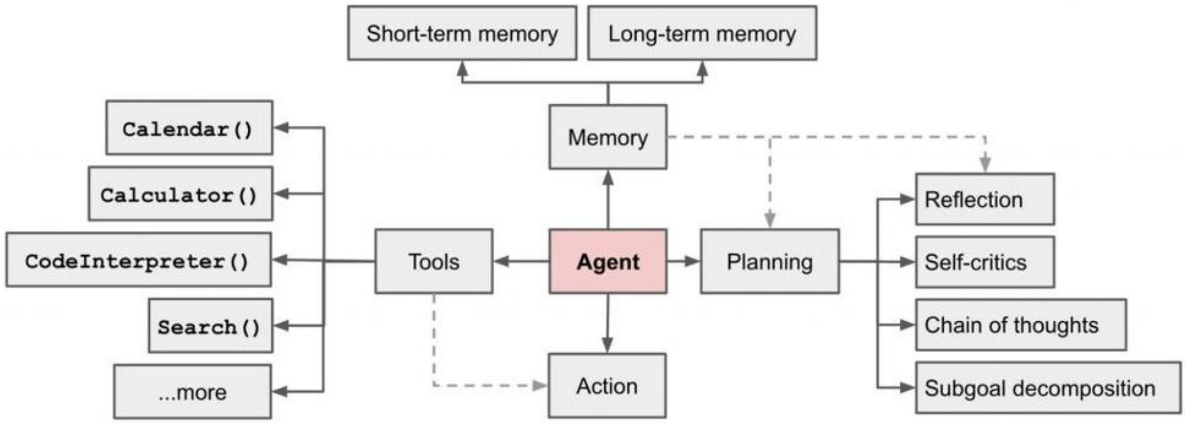


通常，一个大语言模型智能体框架包含以下核心部分：

- 用户请求： 表达用户的问题或请求
- 智能体/大脑： 作为协调者的智能体核心
- 规划： 助于智能体为未来行动做规划
- 记忆： 管理智能体的历史行为和经验，分为长期记忆和短期记忆。
- 工具使用： 调用工具与外部环境交互

通过这些组成部分的协同工作，大模型智能体能够处理从简单到复杂的各种请求，不仅能够提供直接的答案，还能解决需要深度分析和多步骤操作的问题。这种智能体的能力，使其在处理复杂信息查询、数据分析和可视化表示等方面具有巨大潜力。

下图展示了更详细的智能体架构。



对于没有接触过智能体的人，上述框架可能有点抽象，下面我们以一个例子来说明智能体的架构，假如我们想让AI完成以下任务：

> 用户：明天上午要去A地出差，下午与X开会，帮我订好机票、酒店和会议日程。

对于人类助理来说，这是很简单的事情，但以目前大模型的能力，还难以解决这个问题。而从技术角度来说，可以开发一个基于大模型的智能体来解决这类问题，这个智能体主要包含以下部分：

> 1. 规划：通过调用大模型的能力将任务分解，先分为出差行程安排和会议预约两个子任务。

1.1 出差行程安排继续分解为：

1.1.1 从用户所在地到所在城市机场：

a. 其中用户所在地通过感知环境信息获得（可以通过用户初始设置或过往历史存放在记忆组件里面）

b. 决定出行时间

c. 决定出行方式：如乘坐出租车，预约

1.1.2 从用户所在城市机场到A地机场：

a. 决定出行时间

b. 决定出行方式，预约

1.1.3 从A地机场到A地某酒店：

a. 预订A地某酒店

1. 决定入住时间

b. 决定从A地机场到达酒店方式：如乘坐出租车，预约

1.2 会议预约继续分解为：

1.2.1 查看用户和X的日程，找到二者都空闲的时间段

1.2.2 确定会议时间，预约会议

2. 记忆：

2.1 用户所在地应存放在长期记忆中。

2.2 根据规划的分解，后面的步骤会依赖前面步骤的结果，那么需要把前面步骤的执行结果存放在短期记忆中。

3. 工具使用：

3.1 通过调用网约车平台接口预约出租车（需要用户授权，授权信息也可以存储在长期记忆中）

3.2 通过调用航班信息查询接口获取航班信息，调用航班预订接口订机票（需要用户授权，授权信息也可以存储在长期记忆中）

3.3 通过调用OTA平台接口完成酒店预订（需要用户授权，授权信息也可以存储在长期记忆中）

3.4 通过调用支付接口完成支付（需要用户授权，授权信息也可以存储在长期记忆中）

3.5 通过调用日历接口完成日程查询和会议预订

以上是假想的一个智能体应用场景，目前市面上应该还没有这种产品出现。从技术角度来说，这是可行的。

4、提示词工程

什么是提示词?

当我们与类GPT聊天机器人对话时，一般都是一问一答形式。用户先发送一句（段）话，后台大模型生成响应、返回结果（通常是流式输出）。这是用户与大模型交互的典型方式。

所谓**提示词**，就是我们发送给大模型的这句（段）话。提示词可以是任意的自然语言，一般情况下，大模型都能给出不错的响应。但有时我们可能需要多次尝试，才能得到比较满意的结果。

提示词工程就是关于如何组织并优化发送给大模型的语言以便得到最佳响应结果的一套策略方法。

提示词可以包含以下任意要素：

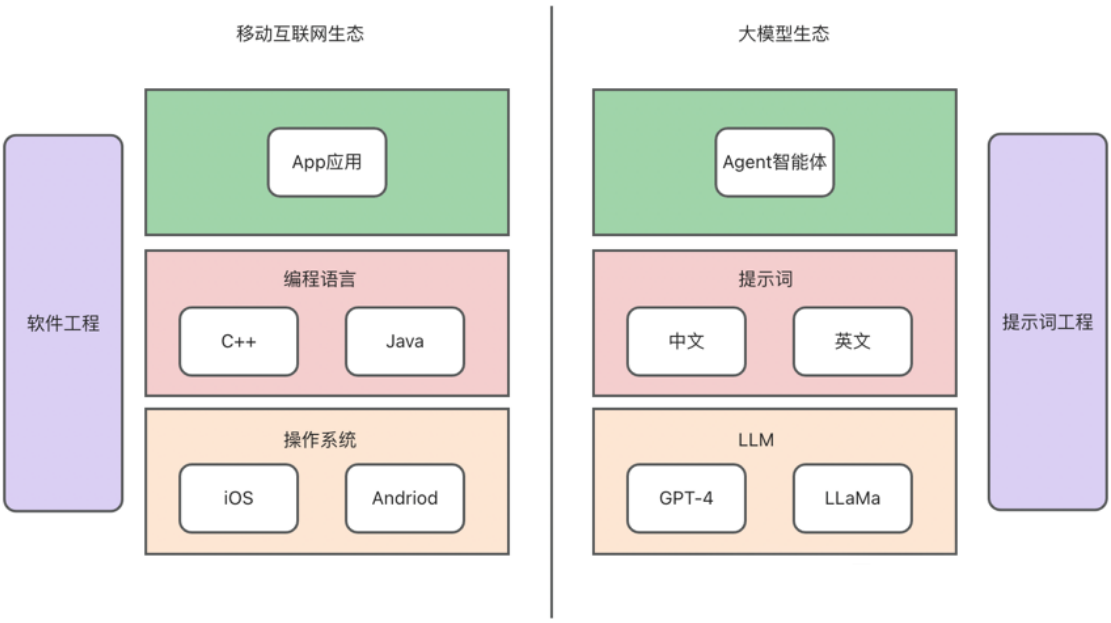
- **指令**：想要模型执行的特定任务或指令。
- **上下文**：包含外部信息或额外的上下文信息，引导语言模型更好地响应。
- **输入数据**：用户输入的内容或问题。
- **输出指示**：指定输出的类型或格式。

这些要素并不一定都出现，也不一定按照某种固定的格式来构造。

一个好的提示词应当具有**具体性、简洁性和明确性**的特点。例如：

```
`帮我写一篇文章`   //这句话可能不是一个好的提示词
`请帮我写一篇关于大模型现状和未来发展的文章，包括大模型发展的时间线、主流厂商及产品简介、未来发展趋势以及总结，字数1000以内。`    //这句话比上一句好很多
```

如果说智能体是大模型的上层应用的话，那么提示词就是开发大模型应用的编程语言，而提示词工程就是关于该语言编程技巧的学科，下图展示了大模型和移动互联网的类比。



了解提示词工程有助于我们写出高质量的提示词，从而让大模型高效地产出优质的结果。下面是一些提示词工程方法。

零样本提示

所谓样本，就是我们在写提示词时，可以包含一个示例，以便让大模型从中学习，从而更好地完成指令。

零样本提示，就是不包含任何样本的提示。这个术语很高级，但其实就是指普通的一句话，一般我们和大模型聊天机器人的对话，大部分都是零样本提示，这两个示例都是零样本提示：

```
`帮我写一篇文章`   //这句话可能不是一个好的提示词
`请帮我写一篇关于大模型现状和未来发展的文章，包括大模型发展的时间线、主流厂商及产品简介、未来发展趋势以及总结，字数1000以内。`    //这句话比上一句好很多
```

但零样本提示并不意味着随便一句话大模型都能给出令人满意的结果。我们在写提示词时，仍然需要尽可能遵循**具体性、简洁性和明确性**的原则。

少样本提示

少样本提示，就是在提示词中包含少量的示例，这样模型可以学习如何生成更好的回答。举个例子：

```
`将以下文本分类为中性、负面或正面。
示例：
    这太棒了！// 正面
    这太糟糕了！// 负面
    明天是星期天！// 中性
    哇，那部电影太棒了！// 正面
文本：我认为这次假期还可以。`
```

这个提示词包含了4条样本，要求大模型对"我认为这次假期还可以。"这句话进行分类。大模型返回结果：

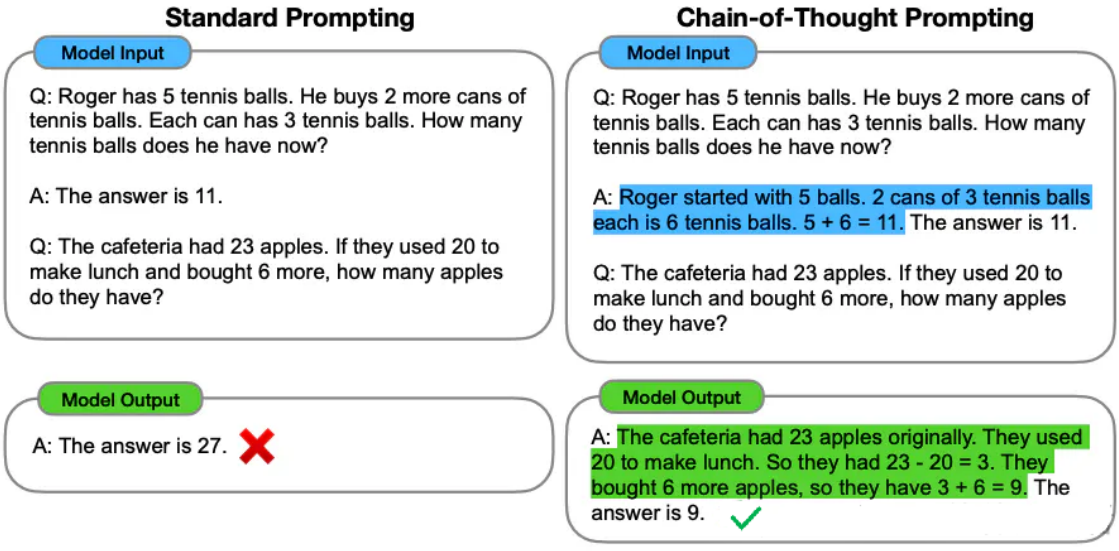
```
中性
```

至于多少样本算少样本，并没有一个客观的标准。例如1条、2条、3条、10条等都属于少量样本。在实际编写提示词时，应结合具体问题和大模型回答的效果加以灵活调整。

链式思考COT

链式思考（Chain of Thought，COT）也叫思维链，是通过指示模型输出中间推理步骤从而得到更好的输出结果。

下图的例子演示了标准提示和链式思考提示的区别。



左图显示的是“标准提示”，这是一个少样本提示，其中（原图英文，下面是翻译）：

> Q：“Roger有5个网球，他又买了2罐网球，每罐有3个网球，问现在他有多少个网球？”
> A：“答案是11。”

是一个样本（不是模型的回答），实际问题是：

> Q：食堂原本有23个苹果，用掉了20个做午餐后又购买了6个。问现在食堂有多少个苹果？

模型输出："答案是27"，显然是错误的。

右图显示的是“链式思考”提示，在样本的答案中增加了推理步骤（蓝色字体）。模型通过学习样本，在回答中也输出了推理步骤（绿色字体），这次的计算结果是正确的。

除了包含少样本的COT之外，还有零样本COT。零样本COT其实非常简单，当你向大模型提出一个问题时，在后面加上一句话"让我们一步一步地思考"。例如，把上面的例子改成零样本提示：

> 食堂原本有23个苹果，用掉了20个做午餐后又购买了6个。问现在食堂有多少个苹果？让我们一步一步地思考

这样模型会输出它的思考过程并给出答案，有助于提高答案的准确性。

除了上面的提到的方法之外，提示词工程还有很多其它的策略方法，这个领域目前也处于广泛的研究和快速发展之中，新的方法不断涌现。对提示词工程感兴趣的，可以看下这个网站https://www.promptingguide.ai/zh。

5、智能体相关技术

网上关于智能体相关技术，经常会看到很多术语，例如：**TOT、提示链、RAG、ART、ReAct、Reflexion**等，这些都是在智能体开发中使用到的技术。从广义上来讲，这些技术也属于提示词工程的范畴，但对普通用户日常在聊天大模型场景中很难应用，更多的是用于构建智能体。下面是对这些术语的简要说明。

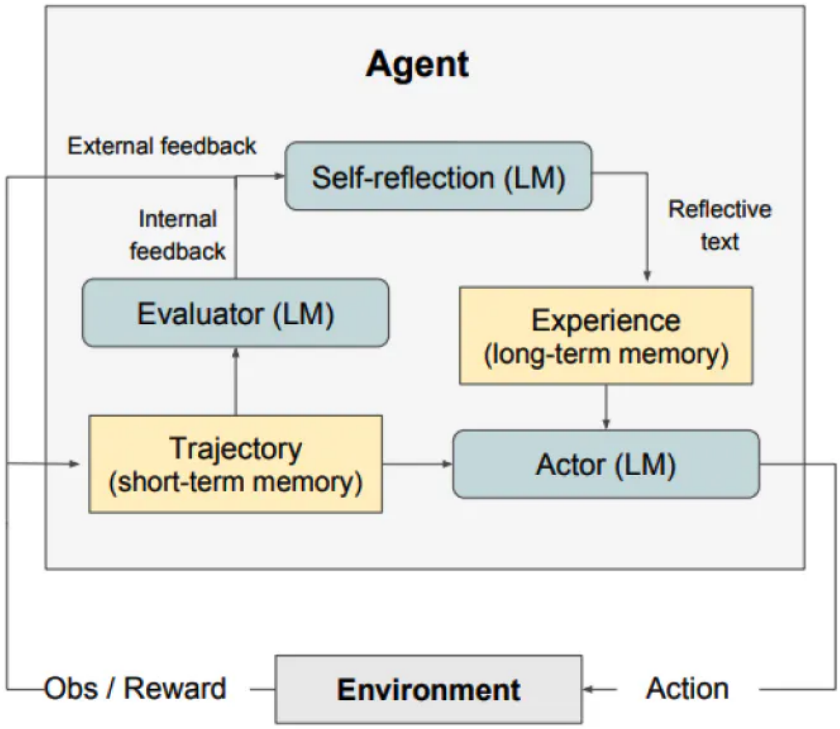
TOT：即思维树（Tree of Thoughts），是思维链（COT）的升级版本。简单来说，它是针对同一个问题，让大模型使用多种不同解决方法，每种方法包含多个步骤，这样构成了一个思维过程的树状结构，大模型可以采用广度优先或深度优先搜索算法遍历这棵树，生成最佳答案。

提示链：提示链是将一个提示的输出作为另一个提示的输入，这样可以组合多个不同的提示词，共同完成预定任务。

RAG：即检索增强生成（Retrieval Augmented Generation），RAG把一个信息检索插件和大模型结合在一起，根据用户的提示词，智能体首先通过信息检索插件检索相关的文档，再将文档连同用户原始输入作为提示词输入到大模型，从而产生更优的结果。其中，RAG的信息检索插件可以是从互联网搜索引擎获取数据，也可以是从本地的向量数据库甚至关系数据库中检索数据。

ART：自动推理并使用工具（Automatic Reasoning and Tool-use），工作原理是：将人工标注好的任务库挂载到大模型，任务库即一组任务示例，包含任务目标和解决步骤。当智能体接到一个新任务的时候，可以在任务库选择合适的示例，自动推理问题求解步骤，并在适当的位置调用外部工具，最终完成任务。

ReAct：即推理+行为（Reason+Act），可以让大模型以交错的方式生成推理轨迹和任务特定操作。ReAct 框架允许 LLMs 与外部工具交互来获取额外信息，从而给出更可靠和实际的回应。



下图是一个ReAct的示例。

(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]

Reflexion：自我反思 自我反思由三个不同的智能体组成：

参与者（Actor）：根据状态观测量生成文本和动作。参与者在环境中采取行动并接受观察结果，从而形成轨迹。链式思考（CoT）和 ReAct 被用作参与者模型。此外，还添加了记忆组件为智能体提供额外的上下文信息。

评估者（Evaluator）：对参与者的输出进行评价。具体来说，它将生成的轨迹（也被称作短期记忆）作为输入并输出奖励分数。根据人物的不同，使用不同的奖励函数（决策任务使用LLM和基于规则的启发式奖励）。

自我反思（Self-Reflection）：生成语言强化线索来帮助参与者实现自我完善。这个角色由大语言模型承担，能够为未来的试验提供宝贵的反馈。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馈，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。

6、单智能体和多智能体

单智能体即一个独立的智能体，它可以自动完成相关任务。对一些比较复杂的任务，单智能体可能难以胜任。此时，可以构建多个相互协作的智能体，共同构成一个系统来处理用户任务。

例如，考虑一个软件项目，当用户提出需求后：

1.由产品经理和用户沟通明确需求，产出产品设计文档。

2.架构师对系统进行架构设计，产出技术设计文档。

3.项目经理分派任务并跟踪进度。

4.开发人员负责编写模块代码。

5.测试人员负责编写测试用例，执行测试。

6.运维人员负责上线发布。

根据软件项目中的角色，我们可以分别构造不同角色的智能体，例如有一个产品经理智能体可以和用户沟通需求，并生成产品设计文档，另外有一个架构师智能体根据产品文档产出技术设计文档，依此类推。让这些智能体相互协作，构成一个软件开发智能体团队，共同完成用户需求。这不是幻想，目前已经有公司在开发类似产品、进行这方面尝试了。

github上这个项目（https://github.com/assafelovic/gpt-researcher），通过多智能体实现了一个能够根据用户提问，自动收集相关信息，规划研究计划，并产出研究报告的多智能体团队。

7、智能体开发技术框架

关于大模型智能体的开发技术架构，听的最多就是langchain，网上介绍文章也很多。langchain是一个第三方的开源大模型应用开发框架，官方提供了python和javascript两种语言支持，其它语言也有一些开发者/组织做了移植，例如langchain4j、langchaingo。除了langchain，还有AutoGPT，微软的AutoGen等，这些框架支持的主流开发语言大多都是python。对于Java背景的开发人员，Java/Spring生态体系下也有Spring AI框架。

这些开发框架主要针对的是技术开发人员，有一定的学习门槛。为了降低开发成本，各大厂商纷纷推出了低代码或零代码的智能体平台，这样非技术背景的人也能很快地创建自己的智能体。

8、大模型厂商的智能体平台

国内的智能体平台主要有：

- 百度旗下的文心智能体平台，提供了零代码和低代码两种开发方式，试用了一下零代码方式，创建简单的智能体还是很快的。

- 字节的扣子，同样也是低代码和零代码方式，提供了知识库、插件、工作流等扩展。
- 阿里旗下的智能体平台，通义星尘。可以创建角色对话智能体。暂时还未用过。
- 腾讯旗下的腾讯元器。

各个厂商的智能体平台大同小异，目前已有的智能体应用都还比较简单，还没有出现杀手级智能体应用。一方面是智能体生态还处于早期发展阶段，另一方面各个厂商为了竞争和争夺用户，匆匆推出自家产品，很多基础能力也不是很完善。

9、结语

本章简单介绍了大模型以及智能体的基本概念，智能体的架构和使用技术、包括提示词工程，单智能体和多智能体的区别，最后罗列了一下智能体开发技术框架和国内各大厂商的智能体开发平台。

大模型智能体将是下一轮技术浪潮的热点。在上一轮移动互联网浪潮中，各类移动端App如雨后春笋般冒出来，而大模型智能体也将会引发新一轮的技术变革。如同移动时代的App逐渐取代PC时代的电脑软件一样，未来AI时代的智能体也将逐步取代移动App。