第 1 章

大模型 Agent 基础

本章将探讨大模型 Agent 的基础知识。首先，我们将梳理从人工智能到大模型 Agent 的发展过程，明确传统 Agent 和现代大模型 Agent 之间的差异，并阐明大模型如何推动 Agent 进入全新的发展阶段。接着，我们将讲解大模型 Agent 的基本定义、关键组成部分，以及大语言模型如何赋能 Agent 的决策、推理和任务执行能力。最后，我们通过介绍当前主流的 Agent 开发框架，分析其特点、适用场景及实际应用，使读者对相关工具有初步认识。

本章知识架构如下：

1.1 AI Agent 发展历程

在本节中，我们将从人工智能的起源谈起，逐步过渡到 AI Agent 的发展历程。我们将介绍 AI Agent 的定义，探讨其如何从最初的规则驱动系统演变为如今具备自主决策与任务执行能力的 Agent。同时，我们将深入了解 Agent 的构造过程，从 API 调用，到基于大语言模型的 Agent。最后，我们通过了解 Agent 在现实世界中的广泛应用，展示 AI Agent 如何在现代企业中创造价值，推动行业变革。

1.1.1 从人工智能到现代 AI Agent

1955 年，约翰·麦卡锡（John McCarthy）等人首次提出了人工智能（artificial intelligence，AI）的概念，标志着这一领域的正式起步。1956 年，在达特茅斯会议上，AI 被正式定义为“使机器能够执行通常需要人类智慧的任务”。在 AI 的早期发展阶段，学者尝试通过符号主义和规则推理来模拟人类的认知，解决一些数学推理、棋牌游戏等结构化任务。此方法依赖于人工编写的知识库和运行的逻辑规则。同年，由阿兰·图灵提出的“图灵测试”成为衡

量机器是否具备智能的重要标准。

1969—1986 年，AI 的研究重点转向了专家系统。专家系统通过一套规则和推理引擎模拟专家的决策过程，因此被广泛应用于医学、金融等领域。然而，这些系统仍然严重受限于人工编写的规则，并且无法处理不确定和模糊的信息。

2000 年后，随着计算能力的提升和数据的积累，深度学习技术取得了显著的进展。大数据的出现也为深度学习的发展提供了重要支撑。大规模的训练数据和强大的计算资源（如 GPU）使深度学习算法能够在多个领域取得突破性进展。

2017 年，论文 *Attention Is All You Need* 提出了 Transformer 架构，这一架构成为现代大语言模型（LLM）的基础。Transformer 架构通过自注意力机制，使模型能够更好地处理长序列数据，成为自然语言处理（NLP）领域的一项革命性突破。正是基于 Transformer 架构，OpenAI 推出了 GPT（generative pretrained transformer）系列模型，从 GPT-1 到 GPT-3，模型的规模和能力逐步提升。

2020 年，GPT-3 预训练模型正式发布。该模型拥有 1750 亿个参数，能够执行文本生成、机器翻译、问答等多种任务，但当时主要在学术界和业界引发关注。直到 2022 年底，OpenAI 团队在 GPT 模型的基础上正式推出了一个能够像真人一样与用户进行交流的 AI 聊天机器人——ChatGPT，这一举措引发了全球性的关注。AI 的发展仿佛瞬间从科研领域走进了日常生活：AI 不再是遥不可及的科学概念，而是成为了人人可用的工具，融入了每个人的生活。

LLM 尽管在文本生成和理解方面表现出色，但仍然存在一些局限性。

- ☑ 缺乏自主决策能力：LLM 不具备自行规划和决策的能力。作为生成式模型，LLM 的输出完全依赖于用户输入的提示信息或相关上下文，无法主动发起任务或在动态环境中做出决策。
- ☑ 缺乏长期记忆和完整任务执行能力：LLM 模型通常只能处理单一任务，且缺乏长期记忆。这导致它们在执行复杂任务时，无法有效衔接多个步骤以完成整体任务。

近两年来，随着基础 LLM 性能的快速迭代，AI 的发展逐步走向了现代的、基于 LLM 的 Agent 系统。通过与强化学习、规划、知识检索、多模态等技术的结合，AI Agent 正逐步发展成为能够自主决策、适应变化并执行复杂任务的 Agent 系统。

1.1.2 Agent 发展路径

1. 初期阶段：简单 API 调用与规则驱动系统

在 AI 发展的初期阶段，计算机系统依赖于规则的编写和简单 API 的调用，可以执行一些特定的任务，这样的计算机系统在当时就被称为“智能体”。随着 AI 技术的发展，在 2000 年左右，Agent 的概念日益丰富。例如：1997 年，IBM 的“深蓝”计算机在国际象棋对抗赛中击败了人类世界冠军；2000 年 Cynthia Breazeal 教授开发了第一个可以用面部模拟人类情感的机器人，它拥有眼睛、眉毛、耳朵和嘴巴，被称为 Kismet；2011 年，由 IBM 创建的名为 Watson 的计算机系统被编程来回答问题，在电视转播的智力比赛 *Jeopardy*！中战胜了两位前冠军；此外，2011 年，苹果公司发布了 Siri，成为第一个全球流行的虚拟助手。

这些 Agent 根据其感知程度的不同，被罗素、诺维格、彼得等人在 *Artificial Intelligence: A Modern Approach* 一书中分为以下五类。

- ☑ 简单反射智能体（Simple Reflex Agent）：根据当前感知选择动作，忽略感知历史的其余部分。例如，空调温控器，我们设定当温度高于某一上限数值时，它就开启空调制冷。这类 Agent 只依据当前的温度来判断开关，并不考虑温度变化或任何其他的历史

信息。

- ☑ 基于模型的反射型智能体（Model-Based Reflex Agent）：在简单反射智能体的基础上，如果结合转移模型和传感器模型，让智能体在传感受限的情况下尽可能地跟踪环境状态，便得到了基于模型的反射型智能体。例如，带有室内导航功能的智能扫地机器人，其内部配置的传感器可以感知空间障碍，并在内部构建出当前的环境地图模型。当智能扫地机器人遇到新的障碍物时，它不仅凭借当前感知转动方向，还会利用内部的地图模型和此前的运动经验，规划一条可以绕过障碍的路径。通过这种追踪环境状态的方式，智能体能够在部分可观测环境下做出更合理的反应。
- ☑ 基于目标的智能体（Goal-Based Agent）：在简单反射型智能体的基础上，如果结合描述某种理想情况的目标信息，智能体可以通过搜索和规划明确“要实现这个目标应该怎么做”，便使其具有了主动性。例如，导航应用会根据用户预设的地点规划路线，评估多条可以抵达目标的路径后选择一条最合理的直达路线。
- ☑ 基于效用的智能体（Utility-Based Agent）：在大多数情况下，仅仅依靠目标并不足以产生高质量的行为。如果我们为智能体添加一个效用函数去度量其性能，再选择要执行的动作，这样就可以使智能体的效用最大化。例如，智能路线规划助手包含评估每条路线的综合价值、时间长短、路况的安全性、是否有收费路段等。效用函数为智能体提供了一套细化的评价标准，使其能够在达到同一目标的多个方案中选出最符合偏好或利益最大的那一个。
- ☑ 学习型智能体（Learning Agent）：具有从过去的经验中学习的能力，并根据学习能力采取行动或做出决定。它从过去获得基础知识，并利用这些学习来自动行动和适应。此类智能体一般包含四部分，分别是学习元素、评估者、性能元素和问题生成器。任何类型的智能体（如基于模型、基于目标、基于效用的智能体）都可以构建学习型智能体，通过学习来提升性能。

2. LLM 的引入：从 ChatGPT 到初步的 Agent

伴随着大语言模型的飞速发展，AI 系统在语言理解和生成方面大幅提升。这些模型能够处理复杂、通用的语言任务，例如文本生成、机器翻译、问答对话等，这为 Agent 的发展奠定了基础。尽管 LLM 在语言处理方面表现出色，但其响应仍然是被动的，缺乏自主决策和任务执行的能力。

3. AutoGPT 的出现：迈向自主执行的 Agent

AutoGPT 的发布标志着 AI Agent 发展的一个重要里程碑。AutoGPT 是一个开源的自主 AI Agent 项目，它在 LLM 的基础上，通过多轮推理和具体任务的分解，可以实现从目标设定到任务执行的全流程自主执行。AutoGPT 的发布引起了广泛关注，被视为实现自主 AI Agent 的重要一步。

4. BabyAGI 的提出：通过自主学习向人工智能迈进

紧随其后，学习型智能体的思想被加入，BabyAGI 通过学习人类婴儿学习和执行任务的方式，使 Agent 具备自我学习和任务管理的能力。它可以设定一个主要目标，自动生成任务列表，并根据任务的完成情况进行调整和优化。BabyAGI 的提出，为 AI Agent 的发展提供了新的思路，强调了自我学习和任务管理的重要性。

1.1.3 AI Agent 的应用

近年，AI Agent 应用进一步落地，从客服助理、代码生成工具，再到金融分析助手，成熟

的应用案例大批涌现，企业也迎来了降本增效、创新改革的新机遇。本节将聚焦当前主流领域的应用实践，介绍并分析 AI Agent 如何在企业场景中发挥价值。

1. 智能客服领域的 AI Agent

在客户服务领域，AI Agent 已经广泛落地，其中最为典型的就是智能客服机器人。它通过自然语言处理与语音识别技术，可以模拟人工客服为用户提供解答，从而大幅提升服务效率与客户满意度。此外，智能客服机器人还可以多端多渠道 7×24 小时实时响应咨询，大幅缓解人工压力的同时保证快速响应。例如：阿里巴巴自研的阿里云智能对话机器人；百度依托“文心一言”和 UNIT 对话平台，为企业提供的端到端的智能客服解决方案。

2. 编程助手与代码生成 AI Agent

AI Agent 在软件开发领域的应用为程序员提供了全新的生产力工具。编程助手能够根据自然语言描述或上下文代码，自动补全代码片段、生成函数实现，显著降低开发人员的工作量。最具代表性的产品便是 GitHub 与 OpenAI 合作推出的 GitHub Copilot，它可以理解多种编程语言的语法和语义，在代码开发过程中实时给出建议。除了 Copilot，业界还有许多类似或相关的工具与项目，如 Cursor、CodeGeeX、通义灵码（Tongyi Lingma）、文心快码（Baidu Comate）等。

3. 金融分析领域的 AI Agent

当前，大模型 AI Agent 在金融服务中的应用已经涵盖客户服务、投资分析、风险控制、市场研报等多个方面。在金融领域的 AI 模型中，专用大型语言模型开始涌现并展现出强大能力。通过海量金融数据的再次训练，金融专用大模型 BloombergGPT 问世，它在各种金融自然语言处理任务上的表现远超通用模型。开源社区也有针对金融领域的大模型，例如：AI4Finance 开发的开源金融大模型 FinGPT，尝试用低成本的方法微调金融数据；腾讯也在其金融云服务中融入了 AI Agent 技术，推出高精度的反欺诈风控引擎、自动化 eKYC 审核等解决方案，甚至通过部署数字人为银行客户提供交互服务，目前已在全球 20 多个国家赋能了上万家企业。

4. 其他企业应用场景的 AI Agent

在运营和营销领域，AI Agent 被用于自动生成营销文案、撰写产品描述、制作个性化广告素材，从而加速市场推广并节约创作成本。在制造与供应链领域，AI Agent 可作为智能调度或质量检测助手：通过分析物联网传感器数据，预测设备何时需要维护；根据历史订单和库存数据，自动调整生产计划，提高供应链效率。企业内部知识管理也是 AI Agent 大显身手的场景之一。企业可部署自己的大模型问答系统，让员工与内部知识库进行对话查询，快速找到解决问题的方案。

AI Agent 正加速融入各行各业的企业级应用中，成为推动业务创新和提升效率的重要引擎。行业专属的大模型将不断涌现，多模态 Agent 将提供更好的人机交互体验，其可控性和安全性也必将进一步提升。

1.2 大模型 Agent 的核心概念

在本节中，我们将回顾大模型 Agent 的基本定义，明确其核心特性。接着，我们将讨论大模型 Agent 在开发过程中的关键组成部分，包括 Tools、Planning、Action 和 Memory 四大模块，并解释这些模块如何协同工作，以完成从任务规划到任务执行的全过程。最后，我们会进

一步探讨 LLM 在驱动 Agent 底层功能方面的优势，展示它如何提升 Agent 在复杂任务中的自我学习、优化和多模态处理能力。

1.2.1 什么是大模型 Agent

大模型 Agent 是基于大规模预训练语言模型构建的一种 Agent。与传统的 Agent 概念不同，现在“Agent”的简称在行业中更多情况下指的是大模型驱动的 Agent 应用。大模型 Agent 在底层大模型的基础上增加了智能决策层，它不仅能够根据用户输入的文本生成合适的回应，还能通过任务规划、执行、反馈等过程，自主完成复杂的任务。

大模型 Agent 的核心特性如下。

- ☑ 自然语言理解和生成：基于大语言模型，能够理解复杂的文本输入，并生成语法合理、语义连贯的回复。
- ☑ 任务执行与决策：与传统的基于规则的 Agent 不同，大模型 Agent 能够根据上下文和目标自主生成解决方案、做出决策，并通过执行实现目标。
- ☑ 跨领域应用能力：大模型强大的通用能力，使得其驱动下的 Agent 可以在多种应用场景下进行迁移和应用。
- ☑ 自我学习与优化：伴随技术的快速发展，部分 Agent 开始具备自我学习的能力。通过与用户的互动，它们能够不断优化自身的表现，提高任务执行的效率和准确性。

本书讨论的所有 Agent 均为基于 LLM 的 Agent。

1.2.2 Agent 的核心组成部分

在 Agent 开发过程中，我们通常将 Agent 分解为不同的功能模块，这些模块协同工作以执行任务。下面我们来了解开发 Agent 时常见的核心组成模块。如图 1.1 所示。

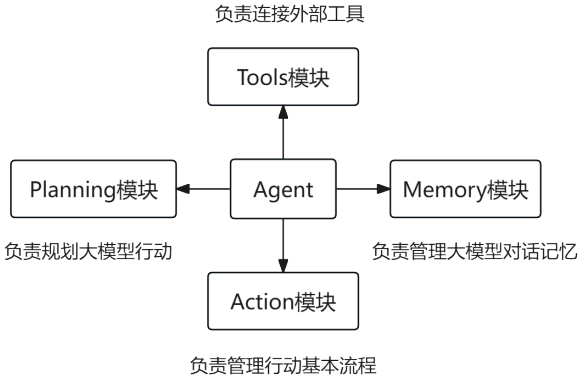

图 1.1 Agent 的核心组成模块

1. Tools 模块

Tools 模块是一个“工具”，它负责提供外部系统或工具的接口，使 Agent 能够执行完成目标所需的外部操作。例如，当我们询问大模型“明天北京的天气怎么样？”时，大模型由于无法从模型训练的数据中找到明天未知的信息，因此无法直接给出答复。在这种情况下，我们需要一个查询天气的 Tools 模块，以便让大模型使用该工具来获取明天天气的情况。

该模块为 Agent 提供连接外部服务或应用的能力，通常包括 API 调用、外部数据库连接、

文件操作等功能。

2. Planning 模块

大模型 Agent 的重要特性之一是自主性，而 Planning 模块负责自主分析并规划执行步骤。Planning 模块会根据当前任务的目标，制定一系列需要执行的行动步骤。通过任务分解、路径选择和资源分配，Planning 模块可以帮助 Agent 设定实现目标的完整策略。此外，它还会根据环境的变化调整计划，确保 Agent 能够灵活应对不同的情境和挑战。

3. Action 模块

Action 模块负责执行由 Planning 模块制定的行动计划。它是 Agent 的“执行器”，会根据计划中的指令执行具体操作。这些操作可能是外部系统交互、处理数据、触发硬件设备等。

4. Memory 模块

Memory 模块是 Agent 的记忆系统，负责存储历史信息、经验和状态。它可以保存 Agent 在任务执行过程中获得的所有相关数据，如历史交互、决策记录、任务执行结果等。正是有了 Memory 模块，Agent 才能够对过去的行为进行反思和优化，从而在未来任务中表现得更加高效和准确。可以说，Memory 模块是 Agent 实现长期学习和优化的基础。

这些模块在 Agent 的构建过程中密切协作，共同推动 Agent 执行复杂任务。Tools 模块提供必要的工具和资源，Planning 模块负责策略制定和决策，Action 模块执行具体的操作，而 Memory 模块则用于优化决策和行为。在本书的后续章节中，我们将更深入探讨这些功能模块，并介绍它们的实现与优化方法。

1.2.3 LLM 在 Agent 中的作用

在当今的技术领域中，我们注意到多数 Agent 依赖大型模型作为其核心驱动力。那么，这些大型模型具体是如何驱动 Agent 的呢？作为 Agent 的基础架构，它们又具备哪些独特优势呢？

1. LLM 增强 Agent 的自然语言理解能力

LLM 在 Agent 中最重要的作用是自然语言理解（NLU）。传统的 Agent 可能依赖规则或简单算法来处理用户的输入，而 LLM 则是基于大规模语料库训练的，能够处理复杂的句子结构、隐含意义、歧义和上下文关系，从而帮助 Agent 准确理解用户的需求和意图。

2. LLM 支持多任务和跨领域应用

现在发布的基座大模型整体上都是通用大模型，这些模型本身拥有跨多个领域的通用知识。与传统的单一任务、单一领域的 Agent 不同，用 LLM 驱动的 Agent 能够应用于文本生成、机器翻译、问答系统、内容总结等跨领域的多项任务，从而极大地扩展了 Agent 的应用范围。

3. LLM 增强 Agent 的推理和决策能力

除了语言的理解和生成，LLM 还能够在一定程度上进行推理和决策。通过训练，LLM 能够根据上下文信息做出合理的推理和判断，支持更复杂的任务处理。LLM 可以在面对不确定性或缺失信息的情况下，结合已有的知识和经验，通过推理补充来预测缺失的部分。这样的推理能力使得 LLM Agent 能够在动态环境中做出决策，适应复杂的任务需求。

4. LLM 支持 Agent 的自我学习和优化

LLM 不仅可以执行静态任务，还可以通过与环境的互动进行自我学习和优化。通过强化学习和在线学习，LLM Agent 可以利用过去的经验不断调整和优化其行为，提升任务完成效率和准确性。

5. LLM 提升 Agent 的多模态处理能力

随着技术的快速发展，LLM 不局限于处理文本数据，它还能够结合其他类型的数据，如图像、视频和音频，实现更加丰富和复杂的任务。在多模态对话系统中，LLM 可以同时处理文本和语音输入，生成符合语境的响应；在智能视觉系统中，LLM 可以帮助理解图像和视频的语义内容，提供基于图像信息的自然语言输出。这种多模态能力使得基于 LLM 的 Agent 能够处理更为复杂和多样化的任务，极大地拓宽了 Agent 在实际应用中的适用范围。

1.3 主流 Agent 开发框架介绍

1.3.1 低代码开发框架

低代码开发框架是一种不需要编写代码的开发方式。用户只需通过网页端的点击、拖曳等可视化操作，就可以轻松构建功能丰富的 Agent 系统。这类框架的最大特点是简化复杂的编程工作，使得没有编程基础的用户也能够快速上手构建 Agent，并实现业务需求。

1. Coze

Coze 是目前火热的低代码开发框架之一，由字节跳动发布。它提供了一个直观的用户界面，让用户无须了解编程语言或复杂的技术细节，就能通过可视化界面创建任务、设计流程，并调用 Agent 所需的工具。Coze 的插件商店十分活跃，开发中所需的大部分功能几乎都可以找到对应的插件。Coze 的主要特点如下。

- ☑ 无须编程：通过可视化界面，用户只需拖放组件和配置选项，就能创建完整的 Agent 应用。
- ☑ 高效集成：提供了多种常见工具和服务的集成接口，用户可以方便地接入数据库、API 或其他外部服务，从而丰富 Agent 的功能。
- ☑ 快速原型开发：用户能够快速构建原型并实时测试，从而加速产品的验证和迭代。
- ☑ 社区支持：Coze 拥有庞大的用户社区和丰富的教程，帮助用户快速上手并解决开发过程中的问题。

2. Dify

Dify 是语灵人工智能科技公司开发的低代码框架。与 Coze 相比，Dify 提供了更多的功能和定制化选项，适合需要高度自定义的 Agent 开发场景。然而，使用 Dify 需要额外的安装和部署，这使得它适用于有一定技术背景的用户或企业级用户。此外，Dify 的 API 接口支持多语言和多地区部署，可以帮助开发者快速将产品推向全球市场。Dify 的主要特点如下。

- ☑ 功能全面：Dify 提供了比 Coze 更强大的功能支持，如自定义脚本编写、更加复杂的业务流程管理、深度集成外部 API 等。
- ☑ 部署需求：与 Coze 的在线平台不同，Dify 需要用户在本地进行安装和部署。这种方式虽然能够提供更多的自定义选项，但也增加了系统的复杂度和维护成本。
- ☑ 灵活性更高：Dify 允许开发者使用更多的编程语言进行自定义扩展，适合希望在低代码框架上进行更深度开发的用户。

3. LangFlow

LangFlow 是基于 LangChain 构建的开源可视化低代码框架，专注于为用户提供灵活的流程

设计和集成功能。用户可以根据业务需求设计自定义流程，并轻松集成各种工具和服务。与 Dify 类似，LangFlow 需要一定的技术背景进行安装部署。LangFlow 的主要特点如下。

- ☑ 流畅的流程设计：LangFlow 提供了丰富的流程图设计工具，让用户能够直观地设计任务的执行步骤，并根据需要调整和优化流程。
- ☑ 插件支持：LangFlow 提供了广泛的插件支持，用户可以轻松将 Agent 与第三方工具或服务（如数据库、API、云服务等）连接。

低代码框架在大模型 Agent 的开发中扮演着重要角色，使得 Agent 的构建不再受限于编程能力或工程背景。通过可视化的拖曳式界面与模块化设计，用户只需简单进行配置，即可实现复杂的 Agent 功能，从而显著降低开发门槛和技术成本。这不仅加快了产品原型的验证周期，也使企业和个人能够更快速、高效地将大语言模型应用于实际场景，推动了 Agent 开发从技术驱动走向业务驱动的转变。

当然，辩证来看，低代码开发框架构建的 Agent 能力仍然是有限的，尤其在面对高度复杂的任务和需求时，它们往往无法像使用传统代码开发的 Agent 一样具备更高的灵活性与扩展性。

1.3.2 基础框架

基础框架是指一种通过利用大型模型的原生能力来构建 Agent 的方法，Agent 无须依赖额外的开发框架或工具，而是直接利用大模型本身的功能来执行任务。随着大语言模型的不断发展，许多现代模型已经内置了强大的 Function Calling 或 Tools Use 能力。例如，OpenAI 为 GPT-3.5 模型增加了名为 Function Calling 的功能，这使得模型能够直接连接并使用外部工具或服务。Function Calling 的基本使用流程如图 1.2 所示。

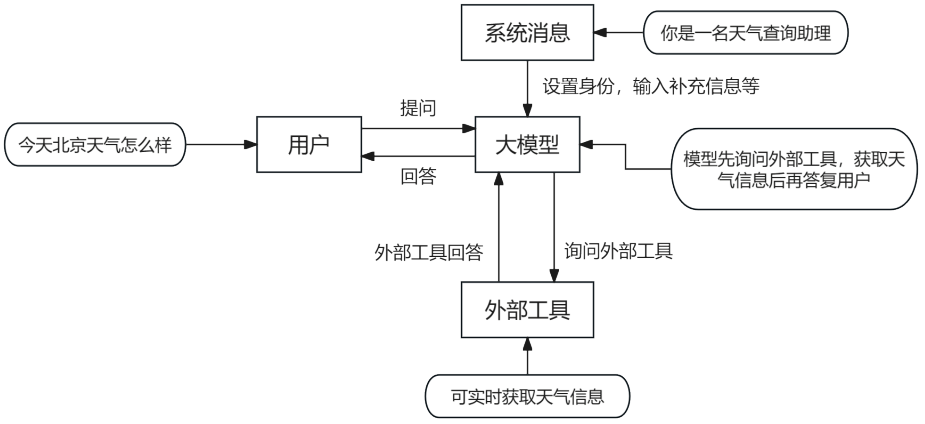

图 1.2 Function Calling 的基本使用流程

总体而言，基础框架充分利用了大模型的强大能力，降低了 Agent 开发的复杂度，使开发者可以直接利用现有的大模型功能来构建 Agent，极大提升了开发效率和灵活性。但是，基础框架的 Agent 通常受到大模型本身功能的限制，面对复杂的业务需求和高度定制化的任务时，原生功能往往无法满足所有的操作和优化需求。

1.3.3 代码框架

在大模型 Agent 的开发中，代码框架提供了更高的灵活性和可定制性，尤其适用于需要复

杂逻辑处理和高度定制化的任务。与低代码或基础框架相比，代码框架通常通过模块化的组件和类来帮助开发者更精细地控制 Agent 的行为，并在构建过程中实现更复杂的功能。

1. LangChain

LangChain 起初是一个针对语言模型的简化开发工具，随着 LLM 的发展、使用者的需求增加和功能的不断扩展，它逐步演化为一个强大的 Agent 开发框架。LangChain 的主要特点如下。

- ☑ 功能强大的类和工具：LangChain 的核心优势之一是它为开发者提供了很多封装好的工具和类，使得构建 Agent 变得简单高效。例如，Agent 构建的四个核心模块，如果从头开始手写代码将十分复杂，而 LangChain 将这些功能封装成易于使用的类，开发者只需要配置和调用这些类，就可快速构建自己的 Agent。
- ☑ 与外部工具集成：LangChain 支持与多种外部工具和服务的无缝集成，包括数据库、API 和文件存储系统。使得开发者可以轻松调用外部系统以获取数据，增强 Agent 的执行能力。
- ☑ 灵活的扩展性：虽然 LangChain 提供了很多封装好的功能和类，开发者仍然可以根据需求定制化开发，可以修改现有类或者添加新的模块，支持更复杂的任务和场景。

LangChain 适用于需要快速开发并且集成多种工具的 Agent 应用。它特别适用于自动化客服系统、内容生成平台、数据处理等任务，因为在这些任务中 Agent 需要与外部系统进行频繁的交互和任务规划。

2. LangGraph

LangGraph 是在 LangChain 的基础上发展而来的，更加注重构建复杂的图状结构并管理多步骤任务。LangGraph 于 2022 年发布，旨在为开发者提供更复杂的图状结构开发工具，特别适用于多任务的协同工作与任务分配。LangGraph 主要特点如下。

- ☑ 支持图状结构：LangGraph 的最大特点是它支持图状结构设计，使得开发者可以将 Agent 的任务和流程组织成一个图形化的形式。每个节点代表一个任务或操作，每条边代表任务之间的依赖关系。这样就可以更加直观地管理任务的执行流程和逻辑关系。
- ☑ 多任务协作：LangGraph 允许多个 Agent 或多个任务并行执行，并通过图结构管理这些任务之间的交互。这使得它特别适用于需要处理多层次、多步骤任务的应用，如复杂的业务流程自动化、跨部门协作等。
- ☑ 灵活的扩展机制：LangGraph 支持开发者自定义任务节点和边的类型，可以根据业务需求扩展任务的执行和交互方式。此外，LangGraph 还允许开发者灵活调整图形结构，优化任务执行的顺序和效率。
- ☑ 支持高级功能：相比 LangChain，LangGraph 提供了更复杂的功能，如动态任务生成、任务优化、依赖管理等。这些功能使得 LangGraph 在处理大规模的任务调度和多 Agent 协作时具有更强的能力。

LangGraph 适用于需要复杂任务协作和图形化任务管理的场景，尤其在跨部门合作、复杂的业务流程管理以及多 Agent 中表现出色。

3. LlamaIndex

LlamaIndex（原名 GPT Index）是一个经典的开源框架，专门用于构建以大语言模型为基础的 Agent 系统。它发布于 2020 年，并且在处理大规模数据集和信息检索方面表现优异。尽管 LlamaIndex 的发展早于 LangChain，但它的使用也更加复杂，对初学者而言入手难度较高，

适合对数据处理和查询优化有较高要求的开发者。LlamaIndex 的主要特点如下。

- ☑ 高效的信息检索与查询能力：LlamaIndex 最突出的特点是其在处理和查询大规模数据集方面的优势。它特别适用于需要快速查询、索引和处理大量信息的场景，如构建搜索引擎、知识图谱等。
- ☑ 灵活的数据结构支持：LlamaIndex 提供了多种数据结构，用于存储和组织不同类型的数据。它可以处理文本数据、结构化数据和多模态数据，为开发者提供了高度自定义的查询方式。
- ☑ 复杂查询优化：LlamaIndex 支持更复杂的查询优化机制，能够在大规模数据中高效地查询和匹配。它为需要处理海量数据的应用提供了优越的性能和可扩展性。

LlamaIndex 适用于大数据和信息检索密集型的应用场景。它特别适合需要高效处理和检索海量信息的 Agent 应用。

1.3.4 Multi-Agent 框架

什么是 Multi-Agent？如果一个任务过于复杂，单个 Agent 难以独立执行，那么就可以通过多个 Agent 分工合作来高效地完成任务。每个 Agent 可以专注于执行任务中的一部分功能，通过协调合作，实现任务的整体目标。

例如，在应用程序开发中，一个完整的项目可能涉及多个角色：需求分析、代码开发、功能测试、性能优化等。如果将每个角色作为一个 Agent 来处理，它们通过相互协作，便可以共同完成应用程序的开发过程。这种方式能够将复杂的任务拆解成多个较小的、可以独立执行的部分，提高工作效率和任务执行的精度。

在实际开发中，Multi-Agent 系统可以通过手动编写多个 Agent 来实现，也可以使用现有的 Multi-Agent 框架简化开发工作。

1. CrewAI

CrewAI 是一个轻量级的多 Agent 框架，它允许开发者通过定义任务、角色和目标，快速组建协作型的多个 Agent 团队。CrewAI 支持多种协作模式，如顺序执行、层级结构和异步工作流，适用于自动化写作、客户支持和项目管理等场景。同时该框架强调高性能和低延迟，适合对响应速度要求较高的应用。

2. OpenAI Swarm

OpenAI 于 2024 年发布的 Swarm 是一个轻量级的多 Agent 协同框架，旨在简化 Agent 之间的协调与执行。Swarm 通过两个基本概念——Agent 和 handoff，实现 Agent 之间的交互。Agent 包含指令和工具，handoff 允许 Agent 将任务交给其他 Agent 处理。这种设计使 Swarm 在处理复杂对话流和任务委派时更加高效。同时因为其简洁清晰的开源项目代码，也十分适用于学习过程中的项目。

3. OpenAI Assistant API

OpenAI 的 Assistant API 也是一个非常强大的工具，它非常适合构建 Agent 并能够处理多种任务，但遗憾的是它并不开源。开发者只能通过在线接口调用模型服务，无法完全控制其内部逻辑，在需要高度定制化的场景下，闭源的 Assistant API 不可避免受到限制。

4. 微软的 AutoGen

AutoGen 是微软开发的一个开源框架，专门为了构建可定制的多 Agent 对话系统而设计。开发者能够在 AutoGen 框架下利用多个 Agent 协同工作，适用于工具使用、任务分配、任务执

行等复杂操作的场景。AutoGen 的主要特点如下。

- ☑ Agent 协作：AutoGen 强调多个 Agent 之间的协作与任务分配。它允许 Agent 在任务执行过程中相互合作，并根据预设的流程和需求调整角色分配。开发者可以定义不同的 Agent 角色，如负责数据采集、代码生成、数据分析等不同功能的角色。
- ☑ 工具集成与扩展性：AutoGen 支持与多种外部工具和 API 集成。它可以帮助 Agent 访问并使用外部工具以进行任务处理，例如调用数据库查询、执行数据分析任务、生成代码等。同时，它也允许开发者根据实际需求定制 Agent 的行为和交互方式。
- ☑ 无代码开发环境：微软也在其基础上提供了一个无代码的开发环境 AutoGen Studio，可以简化 Agent 的构建与调试过程。和此前的低代码开发工具一样，开发者无须编写大量代码，便可以通过可视化界面创建和调试多 Agent 系统，快速构建原型，并进行验证与优化。

5. MetaGPT

MetaGPT 是一个开源的多 Agent 框架，专门用于模拟软件公司内部的协作流程。MetaGPT 通过将不同的角色（如产品经理、架构师、项目经理、工程师）抽象为 Agent，并通过标准化操作流程（SOP）进行协同工作，从而将复杂的软件开发任务拆解成多个子任务，由不同的 Agent 处理。

说明

需要特别说明的是，本书所涉及的大模型 Agent 开发实践将以代码层面的开发为主，侧重于工程化、可部署、可扩展的 Agent 构建方式。虽然当前也有诸如 Coze、Dify 等优秀的低代码框架，它们通过可视化配置即可快速搭建 Agent，但本书不包含这些平台的具体操作流程。对此类低代码工具感兴趣的读者，可访问其官网获取详细使用指南。

1.4 附加资源

（1）AutoGPT GitHub 开源项目地址：https://github.com/Significant-Gravitas/AutoGPT。

（2）Coze 官网地址：https://www.coze.cn/。

（3）Dify 官网地址：https://dify.ai/。

（4）LangFlow 官网地址：https://www.langflow.org/。

（5）OpenAI Function Calling 官方说明文档：https://platform.openai.com/docs/guides/function-calling。

（6）LangChain 官网地址：https://www.langchain.com/。

（7）LlamaIndex 官网地址：https://www.llamaindex.ai/。

（8）LlamaIndex GitHub 开源项目地址：https://github.com/run-llama/llama_index。

（9）CrewAI 官网地址：https://www.crewai.com/。

（10）CrewAI GitHub 开源项目地址：https://github.com/crewAIInc/crewAI。

（11）OpenAI Swarm github 开源项目地址：https://github.com/openai/swarm。

（12）OpenAI Assistant API 官方说明文档：https://platform.openai.com/docs/assistants/overview。

（13）微软 AutoGen GitHub 开源项目地址：https://github.com/microsoft/autogen。

（14）MetaGPT 官网地址：https://deepwisdom.ai/。

（15）MetaGPT GitHub 开源项目地址：https://github.com/FoundationAgents/MetaGPT。