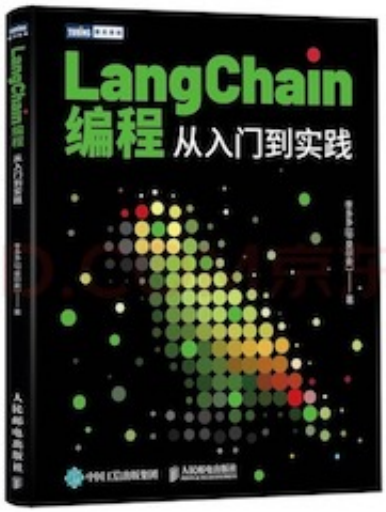
最近关于 AIGC 的热点新闻中，出现了许多以 Agent 方式使用 LLM 的项目，像 AutoGPT、BabyAGI、CAMEL 和 Generative Agents 这样的项目获得极大关注。笔者在研究和借助 LangChain 社区工具实现过这些项目之后，试图简单谈谈它们之间的区别以及每个项目的特性。

我的新书《LangChain编程从入门到实践》已经开售！推荐正在学习AI应用开发的朋友购买阅读！



背景知识

在 LangChain的术语中 Agent 一般指的是将语言模型作为一个推理引擎，并将其与两个关键部分连接起来：工具和记忆。

工具有助于将LLM连接到其他数据或计算的来源。工具的例子包括搜索引擎、API和其他数据存储。工具是有用的，因为LLM只拥有他们被训练的知识。这种知识可能很快就会过时了。为了克服这种限制，工具可以获取最新的数据，并将其作为上下文插入到提示中。工具也可以用来采取行动（例如，运行代码，修改文件等），然后该行动的结果可以被LLM观察到，并被纳入他们关于下一步行动的决定。

记忆帮助 Agent 记住以前的互动。这些互动可以是与其他实体（人类或其他 Agent ），也可以是与工具。这些记忆可以是短期的，也可以是长期的。

在LangChain中 "Agent "称为决定采取什么行动的LLM；"Tools "是Agent可以采取的行动；"Memory "是提取以前事件的行为，而AgentExecutor是在while-loop中运行Agent的逻辑，直到满足某些停止条件。

LangChain Agent 是基于Yao等人在2022年11月提出的推理和行动（ReAct）框架。这种方法的特点是有以下算法：

> 1. 用户给 Agent 一个任务
> 2. 思考： Agent "思考 "要做什么
> 3. 行动/行动输入： Agent 决定采取什么行动（又称使用什么工具）以及该工具的输入应该是什么
> 4. 工具的输出

重复步骤2-4，直到 Agent "认为 "它已经完成。在讨论其他实现和框架时，我们将把它们与这个算法进行比较。

自助 Agent 和 Agent 模拟

首先理清两个概念，自助 Agent 和 Agent 模拟

- **“自主 Agent ”项目（BabyAGI，AutoGPT）的创新之处在于它们的长期目标，这需要新型的规划技术和不同的记忆使用方式。**
- **“ Agent 模拟”项目（CAMEL，Generative Agents）的创新之处在于它们的模拟环境和长期记忆，能够根据事件反映和适应。**

AutoGPT

- LangChain 实现

AutoGPT项目与传统的LangChain Agent 之间的主要区别可以归结为不同的目标。在AutoGPT中，目标往往是更开放的，而且是长期的。这意味着AutoGPT有一个不同的AgentExecutor和不同的内存方式（这两者都是为长期运行的任务而优化的）。以前，LangChain中的 Agent 记忆有两种形式：

- Agent 步骤的记忆：这是通过保留一个与该任务相关的中间 Agent 步骤的列表来完成的，并将完整的列表传递给LLM调用
- 系统记忆：它记住了最终的输入和输出（但忘记了中间的 Agent 步骤）。

由于AutoGPT的运行时间比较长，将 Agent 步骤的完整列表传递给LLM调用不再可行。相反，AutoGPT在中间 Agent 步骤上增加了一个基于检索的记忆。在引擎盖下，这个基于检索的内存正在对嵌入进行语义搜索，使用VectorStore。请注意，LangChain也有这种基于检索的记忆，但它以前是应用于用户与 Agent 之间的互动，而不是 Agent 与工具之间的互动。

在langchain.experimental中加入了这一版本--在我们弄清适当的抽象时，我们把更多的实验性和更新的代码放在这里。具体来说，我们已经实现了所使用的提示模板逻辑，以及用于运行 Agent 的while循环。我们已经使其与LangChain LLM包装器、LangChain VectorStores和LangChain工具兼容。

BabyAGI

- LangChain 实现

BabyAGI项目在以下方面与传统的LangChain Agent 不同：

- 与AutoGPT类似，它将基于检索的记忆应用于中间的 Agent -工具步骤。
- 它有独立的计划和执行步骤，它一次性计划一连串的行动（而不是只计划下一个）。

与AutoGPT类似，BabyAGI是为更多的长时间运行的任务而设计的，这导致了上述两点不同。让我们来扩展一下第二点，因为这是更重要的实质性差异之一。在传统的LangChain Agent 框架（以及AutoGPT框架）中， Agent 每次都会提前一步思考。对于一个给定的世界状态，它思考它的下一个即时行动应该是什么，然后做那个行动。

BabyAGI的不同之处在于，它明确地计划了一连串的行动。然后，它执行第一个动作，然后使用这个结果来做另一个计划步骤并更新它的任务列表。我们的直觉是，这使它能够更好地执行更复杂和更多的任务，通过使用计划步骤，基本上是一个状态跟踪系统。我们观察到（轶事），对于需要许多步骤的任务，传统的LangChain Agent 有时会在几个步骤后忘记其最初的目标，所以提前规划所有的步骤可能是有益的。

Camel

- LangChain 实现

这个项目的新颖之处在于什么?

这个项目的主要新颖性来自于采用两个 Agent ，每个 Agent 都有自己的个性，并让他们相互交谈。在这个意义上，有两个新的组成部分：让两个 Agent 以协作的方式相互交流的想法，以及具体的模拟环境。

两个 Agent 人互动的想法并不完全是新的。鉴于LangChain的模块化性质，我们长期以来一直支持让 Agent 使用其他 Agent 作为工具。然而，这种互动的新颖之处在于，两个 Agent 的地位是平等的--在以前的LangChain实现中，总是有一个 Agent 调用另一个作为工具，采用 "堆叠 "的方式。这种将两个 Agent 置于平等地位的想法，而不是让一个将另一个作为工具来使用，这让人感到特别有趣，可以看到不断演变的行为出现。

请注意，这些 Agent 可以有不同的工具可用，并可以围绕这一点进行专业化。例如，你可以让一个 Agent 拥有编码所需的工具，另一个拥有与线性互动的工具，等等。因此，仍然有可能实现 "堆叠 "效果（你有不同的 Agent 负责不同的事情）。

第二个新颖的组成部分是特定的模拟环境。这是一个双面的对话，并不是非常复杂，但仍然是我们看到的第一个在研究环境中的实现。

Generative Agents

Links:

- 论文地址
- Retriever Implementation
- LangChain Memory Implementation

这个项目有两个新的方面（而且相当复杂）。首先是模拟环境，它由25个不同的 Agent 组成。这似乎相当具体，而且非常复杂，所以我们没有过多地研究这个问题。另一个新的方面是他们为这些 Agent 创建的长期记忆。

我们在本周早些时候对此进行了深入研究。 Agent 人的记忆是由以下部分组成的：

重要性反思步骤，给每个观察结果打一个重要性分数。这个分数可以在下一步的检索中使用，以获取特别重要的记忆，而忽略基本的记忆。

反思步骤，"暂停 "并思考 Agent 人学到了哪些概括。然后，这些反思可以与正常记忆一起被检索。这个反思步骤可以起到浓缩信息和观察最近记忆的模式的作用

一个结合了回顾性、与情况的关联性和重要性的检索器。这可以让那些与当前情况类似的、不久前发生的、特别重要的记忆浮现出来。所有这些似乎都是自然反映我们人类如何 "检索 "记忆的属性。

所有这些记忆成分都是相当新颖的，而且让我们非常兴奋。