> *本文由 简悦 SimpRead 转码， 原文地址 pgmr.cloud*

> *我们听到的最常见的要求之一是为创建自定义代理提供更好的功能和文档。*

我们听到的最常见的要求之一是为创建自定义代理提供更好的功能和文档。这一直有点棘手，因为在我们看来，实际上还不清楚“代理”到底是什么，因此它们的“正确”抽象可能是什么。最近，我们感觉到一些抽象开始融合在一起，所以我们在 Python 和 TypeScript 模块上做了一个大的努力，以更好地执行和记录这些抽象。请参阅下面的技术文档链接，然后是我们介绍的抽象和未来方向的描述。

- Python Custom Agent Docs
- TypeScript Custom Agent Docs

TL；DR：我们引入了 BaseSingleActionAgent 作为代理的最高级别抽象，它可以在我们当前的 AgentExecutor 中使用。我们添加了一个更实用的 LLMSingleActionAgent，它以一种简单且可扩展的方式实现了这个接口（PromptTemplate+LLM+OutputParser）。

BaseSingleActionAgent

我们引入的最基本的抽象是 BaseSingleActionAgent。正如您从名称中可以看出的那样，我们并不认为这是所有代理的基本抽象。相反，我们认为这是一个代理家族的基本抽象，它一次预测一个动作。

在我们当前的 AgentExecutor 中使用了 SingleActionAgent。这个 AgentExecutor 在很大程度上可以被认为是一个循环，它：

1. 将用户输入和任何以前的步骤传递给 Agent
2. 如果 Agent 返回 AgentFinish，则将其直接返回给用户
3. 如果 Agent 返回 AgentAction，则使用它调用工具并获取 Observation
4. 重复，将 AgentAction 和 Observation 传递回 Agent，直到发出 AgentFinish。

AgentAction 是一个由 action 和 action_input 组成的响应。action 指的是要使用的工具，action_input 指的是该工具的输入。

AgentFinish 是一个响应，其中包含要发送回用户的最终消息。这应该用于结束代理运行。

如果你对这个级别的可定制性感兴趣，可以看看这个笔记本。然而，对于大多数用例，我们建议使用下面的抽象。

LLMSingleActionAgent

我们引入的另一个类是 LLMSingleActionAgent。这是 BaseSingleActionAgent 的具体实现，但高度模块化，因此具有高度可定制性。

LLMSingleActionAgent 由四个部分组成：

- PromptTemplate：这是一个提示模板，可以用来指导语言模型做什么
- LLM：这是为代理提供动力的语言模型
- stop sequence：指示 LLM 在找到此字符串后立即停止生成
- OutputParser：它确定如何将 LLM 的输出解析为 AgentAction 或 AgentFinish 对象

将这些结合起来的逻辑是：

- 使用 PromptTemplate 将输入变量（包括用户输入和任何以前的 AgentAction、Observation 对）转换为提示
- 将提示传递给 LLM，并带有特定的停止序列
- 将 LLM 的输出解析为 AgentAction 或 AgentFinish 对象

这些抽象可以用于以多种方式自定义代理。例如：

- 想给你的经纪人一些个性吗？使用 PromptTemplate！
- 想要以特定方式格式化之前的 AgentAction，Observation 对吗？使用 PromptTemplate！
- 想要使用自定义或本地模型吗？编写一个自定义 LLM 包装并将其作为 LLM 传入！
- 输出解析是否过于脆弱，或者您希望以不同的方式处理错误？使用自定义 OutputParser！

（最后一个是粗体，因为这可能是我们听到最多的一个）

我们认为这是最实用的抽象。请参阅博客开头的文档链接，以获取具体的 Python/TypeScripts 指南的链接。

未来发展方向

我们希望这些抽象概念已经澄清了我们对代理的一些想法，并为我们希望社区能够做出贡献开辟了空间。特别地：

我们对 SingleActionAgents 的其他例子感到非常兴奋，比如：

- 在调用 LLM 之前使用嵌入进行工具选择
- 使用本构链代替 LLMChain 来提高可靠性

我们也对其他类型的代理感到兴奋（这将需要新的代理执行器），如：

- 多作用代理
- 计划执行代理

如果其中任何一个听起来很有趣，我们总是愿意与人们合作来实现他们的想法！最好的方法可能是做一些初始工作，打开一个 RFC 拉取请求，我们很乐意从那里开始：）