Agents

Contents

Note

Conceptual Guide

Some applications will require not just a predetermined chain of calls to LLMs/other tools, but potentially an unknown chain that depends on the user's input. In these types of chains, there is a "agent" which has access to a suite of tools. Depending on the user input, the agent can then decide which, if any, of these tools to call.
At the moment, there are two main types of agents:

1. "Action Agents": these agents decide an action to take and take that action one step at a time
2. "Plan-and-Execute Agents": these agents first decide a plan of actions to take, and then execute those actions one at a time.

When should you use each one? Action Agents are more conventional, and good for small tasks. For more complex or long running tasks, the initial planning step helps to maintain long term objectives and focus. However, that comes at the expense of generally more calls and higher latency. These two agents are also not mutually exclusive - in fact, it is often best to have an Action Agent be in charge of the execution for the Plan and Execute agent.

Action Agents

□□

High level pseudocode of agents looks something like:

CTRL + K

Skip to main content

- The *agent* decides which *tool* - if any - to use, and what the input to that tool should be
- That *tool* is then called with that *tool input*, and an *observation* is recorded (this is just the output of calling that tool with that tool input)
- That history of *tool*, *tool input*, and *observation* is passed back into the *agent*, and it decides what step to take next
- This is repeated until the *agent* decides it no longer needs to use a *tool*, and then it responds directly to the user.

The different abstractions involved in agents are as follows:

- **Agent: this is where the logic of the application lives. Agents expose an interface that takes in user input along with a list of previous steps the agent has taken, and returns either an *AgentAction* or *AgentFinish***
    - *AgentAction* corresponds to the tool to use and the input to that tool
    - *AgentFinish* means the agent is done, and has information around what to return to the user
- Tools: these are the actions an agent can take. What tools you give an agent highly depend on what you want the agent to do
- Toolkits: these are groups of tools designed for a specific use case. For example, in order for an agent to interact with a SQL database in the best way it may need access to one tool to execute queries and another tool to inspect tables.
- Agent Executor: this wraps an agent and a list of tools. This is responsible for the loop of running the agent iteratively until the stopping criteria is met.

The most important abstraction of the four above to understand is that of the agent. Although an agent can be defined in whatever way one chooses, the typical way to construct an agent is with:

- PromptTemplate: this is responsible for taking the user input and previous steps and constructing a prompt to send to the language model
- Language Model: this takes the prompt constructed by the PromptTemplate and returns some output
- Output Parser: this takes the output of the Language Model and parses it into an *AgentAction* or *AgentFinish* object.

In this section of documentation, we first start with a Getting Started notebook to d
use all things related to agents in an end-to-end manner.

□□
CTRL + K
Skip to main content

Tools

In this section we cover the different types of tools LangChain supports natively. We then cover how to add your own tools.

Agents

In this section we cover the different types of agents LangChain supports natively. We then cover how to modify and create your own agents.

Toolkits

In this section we go over the various toolkits that LangChain supports out of the box, and how to create an agent from them.

Agent Executor

In this section we go over the Agent Executor class, which is responsible for calling the agent and tools in a loop. We go over different ways to customize this, and options you can use for more control.

Go Deeper

Tools

Agents

Toolkits

Agent Executors

Plan-and-Execute Agents

High level pseudocode of agents looks something like:

- Some user input is received
- The planner lists out the steps to take
- The executor goes through the list of steps, executing them

The most typical implementation is to have the planner be a language model, and the executor be an action agent.

Go Deeper

Plan and Execute

Imports

Tools

Planner, Executor, and Agent

Run Example

□□

CTRL + K