Agents

Some applications require a flexible chain of calls to LLMs and other tools based on user input. The **Agent** interface provides the flexibility for such applications. An agent has access to a suite of tools, and determines which ones to use depending on the user input. Agents can use multiple tools, and use the output of one tool as the input to the next.

There are two main types of agents:

- **Action agents**: at each timestep, decide on the next action using the outputs of all previous actions
- **Plan-and-execute agents**: decide on the full sequence of actions up front, then execute them all without updating the plan

Action agents are suitable for small tasks, while plan-and-execute agents are better for complex or long-running tasks that require maintaining long-term objectives and focus. Often the best approach is to combine the dynamism of an action agent with the planning abilities of a plan-and-execute agent by letting the plan-and-execute agent use action agents to execute plans.

For a full list of agent types see agent types. Additional abstractions involved in agents are:

- **Tools**: the actions an agent can take. What tools you give an agent highly depend on what you want the agent to do
- **Toolkits**: wrappers around collections of tools that can be used together a specific use case. For example, in order for an agent to interact with a SQL database it will likely need one tool to execute queries and another to inspect tables

Action agents

At a high-level an action agent:

1. Receives user input
2. Decides which tool, if any, to use and the tool input
3. Calls the tool and records the output (also known as an "observation")
4. Decides the next step using the history of tools, tool inputs, and observations
5. Repeats 3-4 until it determines it can respond directly to the user

Action agents are wrapped in **agent executors**, which are responsible for calling the agent, getting back an action and action input, calling the tool that the action references with the generated input, getting the output of the tool, and then passing all that information back into the agent to get the next action it should take.

Although an agent can be constructed in many ways, it typically involves these components:

- **Prompt template**: Responsible for taking the user input and previous steps and constructing a prompt to send to the language model
- **Language model**: Takes the prompt with use input and action history and decides what to do next
- **Output parser**: Takes the output of the language model and parses it into the next action or a final answer

Plan-and-execute agents

At a high-level a plan-and-execute agent:

1. Receives user input
2. Plans the full sequence of steps to take
3. Executes the steps in order, passing the outputs of past steps as inputs to future steps

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

Get started

```
from langchain.agents import load_tools
from langchain.agents import initialize_agent
from langchain.agents import AgentType
from langchain.llms import OpenAI
```

First, let's load the language model we're going to use to control the agent.

```
llm = OpenAI(temperature=0)
```

Next, let's load some tools to use. Note that the `llm-math` tool uses an LLM, so we need to pass that in.

```
tools = load_tools(["serpapi", "llm-math"], llm=llm)
```

Finally, let's initialize an agent with the tools, the language model, and the type of agent we want to use.

```
agent = initialize_agent(tools, llm, agent=AgentType.ZERO_SHOT_REACT_DESCRIPTION, verbose=True)
```

Now let's test it out!

```
agent.run("Who is Leo DiCaprio's girlfriend? What is her current age raised to the 0.43 power?")
```

```
> Entering new AgentExecutor chain...
 I need to find out who Leo DiCaprio's girlfriend is and then calculate her age raised to the 0.43 power.
Action: Search
Action Input: "Leo DiCaprio girlfriend"
Observation: Camila Morrone
Thought: I need to find out Camila Morrone's age
Action: Search
Action Input: "Camila Morrone age"
Observation: 25 years
Thought: I need to calculate 25 raised to the 0.43 power
Action: Calculator
Action Input: 25^0.43
Observation: Answer: 3.991298452658078

Thought: I now know the final answer
Final Answer: Camila Morrone is Leo DiCaprio's girlfriend and her current age raised to the 0.43 power is 3.991298452658078.

> Finished chain.
```

```
"Camila Morrone is Leo DiCaprio's girlfriend and her current age raised to the 0.43 power is 3.991298452658078."
```