Multi-Input Tools

This notebook shows how to use a tool that requires multiple inputs with an agent. The recommended way to do so is with the `StructuredTool` class.

```
import os

os.environ["LANGCHAIN_TRACING"] = "true"
```

```
from langchain import OpenAI
from langchain.agents import initialize_agent, AgentType

llm = OpenAI(temperature=0)
```

API Reference:

- initialize_agent from `langchain.agents`
- AgentType from `langchain.agents`

```
from langchain.tools import StructuredTool


def multiplier(a: float, b: float) -> float:
    """Multiply the provided floats."""
    return a * b


tool = StructuredTool.from_function(multiplier)
```

API Reference:

- StructuredTool from `langchain.tools`

```
# Structured tools are compatible with the
STRUCTURED_CHAT_ZERO_SHOT_REACT_DESCRIPTION agent type.
```

```
agent_executor = initialize_agent(
    [tool],
    llm,
    agent=AgentType.STRUCTURED_CHAT_ZERO_SHOT_REACT_DESCRIPTION,
    verbose=True,
)
```

```
agent_executor.run("What is 3 times 4")
```

```
> Entering new AgentExecutor chain...

Thought: I need to multiply 3 and 4
Action:
```
{
  "action": "multiplier",
  "action_input": {"a": 3, "b": 4}
}
```

Observation: 12
Thought: I know what to respond
Action:
```
{
  "action": "Final Answer",
  "action_input": "3 times 4 is 12"
}
```

> Finished chain.
```

```
'3 times 4 is 12'
```

Multi-Input Tools with a string format

An alternative to the structured tool would be to use the regular `Tool` class and accept a single string. The tool would then have to handle the parsing logic to extract the relavent values from the text, which tightly couples the tool representation to the agent prompt. This is still useful if the underlying language model can't reliabl generate structured schema.

Let's take the multiplication function as an example. In order to use this, we will tell the agent to generate the "Action Input" as a comma-separated list of length two. We will then write a thin wrapper that takes a string, splits it into two around a comma, and passes both parsed sides as integers to the multiplication function.

```
from langchain.llms import OpenAI
from langchain.agents import initialize_agent, Tool
from langchain.agents import AgentType
```

API Reference:

- OpenAI from `langchain.llms`
- initialize_agent from `langchain.agents`
- Tool from `langchain.agents`
- AgentType from `langchain.agents`

Here is the multiplication function, as well as a wrapper to parse a string as input.

```
def multiplier(a, b):
    return a * b


def parsing_multiplier(string):
    a, b = string.split(",")
    return multiplier(int(a), int(b))
```

```
llm = OpenAI(temperature=0)
tools = [
    Tool(
        name="Multiplier",
```

```
        func=parsing_multiplier,
        description="useful for when you need to multiply two numbers
together. The input to this tool should be a comma separated list of
numbers of length two, representing the two numbers you want to multiply
together. For example, `1,2` would be the input if you wanted to multiply
1 by 2.",
    )
]
mrkl = initialize_agent(
    tools, llm, agent=AgentType.ZERO_SHOT_REACT_DESCRIPTION, verbose=True
)
```

```
mrkl.run("What is 3 times 4")
```

```
> Entering new AgentExecutor chain...
 I need to multiply two numbers
Action: Multiplier
Action Input: 3,4
Observation: 12
Thought: I now know the final answer
Final Answer: 3 times 4 is 12

> Finished chain.
```

```
'3 times 4 is 12'
```