Custom MRKL agent

This notebook goes through how to create your own custom MRKL agent.

A MRKL agent consists of three parts:

```
- Tools: The tools the agent has available to use.
- LLMChain: The LLMChain that produces the text that is parsed in a
certain way to determine which action to take.
- The agent class itself: this parses the output of the LLMChain to
determine which action to take.
```

In this notebook we walk through how to create a custom MRKL agent by creating a custom LLMChain.

Custom LLMChain

The first way to create a custom agent is to use an existing Agent class, but use a custom LLMChain. This is the simplest way to create a custom Agent. It is highly recommended that you work with the `ZeroShotAgent`, as at the moment that is by far the most generalizable one.

Most of the work in creating the custom LLMChain comes down to the prompt. Because we are using an existing agent class to parse the output, it is very important that the prompt say to produce text in that format. Additionally, we currently require an `agent_scratchpad` input variable to put notes on previous actions and observations. This should almost always be the final part of the prompt. However, besides those instructions, you can customize the prompt as you wish.

To ensure that the prompt contains the appropriate instructions, we will utilize a helper method on that class. The helper method for the `ZeroShotAgent` takes the following arguments:

- tools: List of tools the agent will have access to, used to format the prompt.
- prefix: String to put before the list of tools.

- suffix: String to put after the list of tools.
- input_variables: List of input variables the final prompt will expect.

For this exercise, we will give our agent access to Google Search, and we will customize it in that we will have it answer as a pirate.

```
from langchain.agents import ZeroShotAgent, Tool, AgentExecutor
from langchain import OpenAI, SerpAPIWrapper, LLMChain
```

API Reference:

- ZeroShotAgent from `langchain.agents`
- Tool from `langchain.agents`
- AgentExecutor from `langchain.agents`

```
search = SerpAPIWrapper()
tools = [
    Tool(
        name="Search",
        func=search.run,
        description="useful for when you need to answer questions about
current events",
    )
]
```

```
prefix = """Answer the following questions as best you can, but speaking
as a pirate might speak. You have access to the following tools:"""
suffix = """Begin! Remember to speak as a pirate when giving your final
answer. Use lots of "Args"

Question: {input}
{agent_scratchpad}"""

prompt = ZeroShotAgent.create_prompt(
    tools, prefix=prefix, suffix=suffix, input_variables=["input",
"agent_scratchpad"]
)
```

In case we are curious, we can now take a look at the final prompt template to see what it looks like when its all put together.

```
print(prompt.template)
```

```
    Answer the following questions as best you can, but speaking as a
pirate might speak. You have access to the following tools:

    Search: useful for when you need to answer questions about current
events

    Use the following format:

    Question: the input question you must answer
    Thought: you should always think about what to do
    Action: the action to take, should be one of [Search]
    Action Input: the input to the action
    Observation: the result of the action
    ... (this Thought/Action/Action Input/Observation can repeat N times)
    Thought: I now know the final answer
    Final Answer: the final answer to the original input question

    Begin! Remember to speak as a pirate when giving your final answer.
Use lots of "Args"

    Question: {input}
    {agent_scratchpad}
```

Note that we are able to feed agents a self-defined prompt template, i.e. not restricted to the prompt generated by the `create_prompt` function, assuming it meets the agent's requirements.

For example, for `ZeroShotAgent`, we will need to ensure that it meets the following requirements. There should a string starting with "Action:" and a following string starting with "Action Input:", and both should be separated by a newline.

```
llm_chain = LLMChain(llm=OpenAI(temperature=0), prompt=prompt)
```

```
tool_names = [tool.name for tool in tools]
agent = ZeroShotAgent(llm_chain=llm_chain, allowed_tools=tool_names)
```

```
agent_executor = AgentExecutor.from_agent_and_tools(
    agent=agent, tools=tools, verbose=True
)
```

```
agent_executor.run("How many people live in canada as of 2023?")
```

```
    > Entering new AgentExecutor chain...
    Thought: I need to find out the population of Canada
    Action: Search
    Action Input: Population of Canada 2023
    Observation: The current population of Canada is 38,661,927 as of
Sunday, April 16, 2023, based on Worldometer elaboration of the latest
United Nations data.
    Thought: I now know the final answer
    Final Answer: Arrr, Canada be havin' 38,661,927 people livin' there as
of 2023!

    > Finished chain.
```

```
    "Arrr, Canada be havin' 38,661,927 people livin' there as of 2023!"
```

Multiple inputs

Agents can also work with prompts that require multiple inputs.

```
prefix = """Answer the following questions as best you can. You have
access to the following tools:"""
suffix = """When answering, you MUST speak in the following language:
{language}.
```

```
Question: {input}
{agent_scratchpad}"""

prompt = ZeroShotAgent.create_prompt(
    tools,
    prefix=prefix,
    suffix=suffix,
    input_variables=["input", "language", "agent_scratchpad"],
)
```

```
llm_chain = LLMChain(llm=OpenAI(temperature=0), prompt=prompt)
```

```
agent = ZeroShotAgent(llm_chain=llm_chain, tools=tools)
```

```
agent_executor = AgentExecutor.from_agent_and_tools(
    agent=agent, tools=tools, verbose=True
)
```

```
agent_executor.run(
    input="How many people live in canada as of 2023?", language="italian"
)
```

```
    > Entering new AgentExecutor chain...
    Thought: I should look for recent population estimates.
    Action: Search
    Action Input: Canada population 2023
    Observation: 39,566,248
    Thought: I should double check this number.
    Action: Search
    Action Input: Canada population estimates 2023
    Observation: Canada's population was estimated at 39,566,248 on
January 1, 2023, after a record population growth of 1,050,110 people from
January 1, 2022, to January 1, 2023.
    Thought: I now know the final answer.
```

```
    Final Answer: La popolazione del Canada è stata stimata a 39.566.248
il 1° gennaio 2023, dopo un record di crescita demografica di 1.050.110
persone dal 1° gennaio 2022 al 1° gennaio 2023.

    > Finished chain.




    'La popolazione del Canada è stata stimata a 39.566.248 il 1° gennaio
2023, dopo un record di crescita demografica di 1.050.110 persone dal 1°
gennaio 2022 al 1° gennaio 2023.'
```