XML Agent

Some language models (like Anthropic's Claude) are particularly good at reasoning/writing XML. This goes over how to use an agent that uses XML when prompting.

- Use with regular LLMs, not with chat models.
- Use only with unstructured tools; i.e., tools that accept a single string input.
- See AgentTypes documentation for more agent types.

```
from langchain import hub
from langchain.agents import AgentExecutor,
create_xml_agent
from langchain_community.chat_models import
ChatAnthropic
from langchain_community.tools.tavily_search
import TavilySearchResults
```

Initialize Tools

We will initialize the tools we want to use

```
tools = [TavilySearchResults(max_results=1)]
```

Create Agent

```
# Get the prompt to use - you can modify
this!
prompt = hub.pull("hwchase17/xml-agent-
convo")
```

```
# Choose the LLM that will drive the agent
llm = ChatAnthropic(model="claude-2")

# Construct the XML agent
agent = create_xml_agent(llm, tools, prompt)
```

Run Agent

```
# Create an agent executor by passing in the
agent and tools
agent_executor = AgentExecutor(agent=agent,
tools=tools, verbose=True)
```

```
agent_executor.invoke({"input": "what is
LangChain?"})
```

```
> Entering new AgentExecutor chain...
 <tool>tavily_search_results_json</tool>
<tool_input>what is LangChain?[{'url':
'https://aws.amazon.com/what-is/langchain/',
'content': 'What Is LangChain? What is
LangChain?  How does LangChain work?  Why is
LangChain important?  that LangChain provides
to reduce development time.LangChain is an
open source framework for building
applications based on large language models
(LLMs). LLMs are large deep-learning models
pre-trained on large amounts of data that can
generate responses to user queries—for
example, answering questions or creating
images from text-based prompts.'}]
<final_answer>LangChain is an open source
framework for building applications based on
large language models (LLMs). It allows
developers to leverage the power of LLMs to
create applications that can generate
responses to user queries, such as answering
questions or creating images from text
prompts. Key benefits of LangChain are
reducing development time and effort compared
```

```
to building custom LLMs from scratch.
</final_answer>

> Finished chain.
```

```
{'input': 'what is LangChain?',
 'output': 'LangChain is an open source
framework for building applications based on
large language models (LLMs). It allows
developers to leverage the power of LLMs to
create applications that can generate
responses to user queries, such as answering
questions or creating images from text
prompts. Key benefits of LangChain are
reducing development time and effort compared
to building custom LLMs from scratch.'}
```

Using with chat history

```
from langchain_core.messages import
AIMessage, HumanMessage

agent_executor.invoke(
    {
        "input": "what's my name? Only use a
tool if needed, otherwise respond with Final
Answer",
```

```
        # Notice that chat_history is a
string, since this prompt is aimed at LLMs,
not chat models
        "chat_history": "Human: Hi! My name
is Bob\nAI: Hello Bob! Nice to meet you",
    }
)
```

```
> Entering new AgentExecutor chain...
 <final_answer>Your name is Bob.
</final_answer>

Since you already told me your name is Bob, I
do not need to use any tools to answer the
question "what's my name?". I can provide the
final answer directly that your name is Bob.

> Finished chain.
```

```
{'input': "what's my name? Only use a tool if
needed, otherwise respond with Final Answer",
 'chat_history': 'Human: Hi! My name is
Bob\nAI: Hello Bob! Nice to meet you',
 'output': 'Your name is Bob.'}
```