Returning Structured Output

This notebook covers how to have an agent return a structured output. By default, most of the agents return a single string. It can often be useful to have an agent return something with more structure.

A good example of this is an agent tasked with doing question-answering over some sources. Let's say we want the agent to respond not only with the answer, but also a list of the sources used. We then want our output to roughly follow the schema below:

```
class Response(BaseModel):
    """Final response to the question being
asked"""
    answer: str = Field(description = "The
final answer to respond to the user")
    sources: List[int] =
Field(description="List of page chunks that
contain answer to the question. Only include
a page chunk if it contains relevant
information")
```

In this notebook we will go over an agent that has a retriever tool and responds in the correct format.

Create the Retriever

In this section we will do some setup work to create our retriever over some mock data containing the “State of the Union” address. Importantly, we will add a “page_chunk” tag to the metadata of each document. This is just some fake data intended to simulate a source field. In practice, this would more likely be the URL or path of a document.

```
%pip install -qU chromadb langchain
langchain-community langchain-openai
```

```
from langchain.text_splitter import
RecursiveCharacterTextSplitter
from langchain_community.document_loaders
import TextLoader
from langchain_community.vectorstores import
Chroma
from langchain_openai import OpenAIEmbeddings
```

```
# Load in document to retrieve over
loader =
TextLoader("../../state_of_the_union.txt")
documents = loader.load()

# Split document into chunks
text_splitter =
RecursiveCharacterTextSplitter(chunk_size=1000,
chunk_overlap=0)
texts =
text_splitter.split_documents(documents)

# Here is where we add in the fake source
information
for i, doc in enumerate(texts):
    doc.metadata["page_chunk"] = i

# Create our retriever
embeddings = OpenAIEmbeddings()
vectorstore = Chroma.from_documents(texts,
embeddings, collection_name="state-of-union")
retriever = vectorstore.as_retriever()
```

Create the tools

We will now create the tools we want to give to the agent. In this case, it is just one - a tool that wraps our retriever.

```
from langchain.tools.retriever import
create_retriever_tool
```

```
retriever_tool = create_retriever_tool(
    retriever,
    "state-of-union-retriever",
    "Query a retriever to get information
about state of the union address",
)
```

Create response schema

Here is where we will define the response schema. In this case, we want the final answer to have two fields: one for the `answer`, and then another that is a list of `sources`

```
from typing import List

from langchain_core.pydantic_v1 import
BaseModel, Field


class Response(BaseModel):
    """Final response to the question being
asked"""

    answer: str = Field(description="The
final answer to respond to the user")
    sources: List[int] = Field(
        description="List of page chunks that
contain answer to the question. Only include
a page chunk if it contains relevant
information"
    )
```

Create the custom parsing logic

We now create some custom parsing logic. How this works is that we will pass the `Response` schema to the OpenAI LLM via their `functions` parameter. This is similar to how we pass tools for the agent to use.

When the `Response` function is called by OpenAI, we want to use that as a signal to return to the user. When any other function is called by OpenAI, we treat that as a tool invocation.

Therefore, our parsing logic has the following blocks:

- If no function is called, assume that we should use the response to respond to the user, and therefore return `AgentFinish`
- If the `Response` function is called, respond to the user with the inputs to that function (our structured output), and therefore return `AgentFinish`

- If any other function is called, treat that as a tool invocation, and therefore return `AgentActionMessageLog`

Note that we are using `AgentActionMessageLog` rather than `AgentAction` because it lets us attach a log of messages that we can use in the future to pass back into the agent prompt.

```
import json

from langchain_core.agents import
AgentActionMessageLog, AgentFinish
```

```
def parse(output):
    # If no function was invoked, return to
user
    if "function_call" not in
output.additional_kwargs:
        return AgentFinish(return_values=
{"output": output.content},
log=output.content)

    # Parse out the function call
    function_call =
output.additional_kwargs["function_call"]
    name = function_call["name"]
    inputs =
json.loads(function_call["arguments"])
```

```
    # If the Response function was invoked,
return to the user with the function inputs
    if name == "Response":
        return
AgentFinish(return_values=inputs,
log=str(function_call))
    # Otherwise, return an agent action
    else:
        return AgentActionMessageLog(
            tool=name, tool_input=inputs,
log="", message_log=[output]
        )
```

Create the Agent

We can now put this all together! The components of this agent are:

- prompt: a simple prompt with placeholders for the user's question and then the `agent_scratchpad` (any intermediate steps)
- tools: we can attach the tools and `Response` format to the LLM as functions
- format scratchpad: in order to format the `agent_scratchpad` from intermediate steps, we will use

the standard `format_to_openai_function_messages`. This takes intermediate steps and formats them as AIMessages and FunctionMessages.

- output parser: we will use our custom parser above to parse the response of the LLM
- AgentExecutor: we will use the standard AgentExecutor to run the loop of agent-tool-agent-tool…

```
from langchain.agents import AgentExecutor
from langchain.agents.format_scratchpad
import format_to_openai_function_messages
from langchain_core.prompts import
ChatPromptTemplate, MessagesPlaceholder
from langchain_openai import ChatOpenAI
```

```
prompt = ChatPromptTemplate.from_messages(
    [
        ("system", "You are a helpful assistant
        ("user", "{input}"),

MessagesPlaceholder(variable_name="agent_scratc
    ]
)
```

```
llm = ChatOpenAI(temperature=0)
```

```
llm_with_tools =
llm.bind_functions([retriever_tool,
Response])
```

```
agent = (
    {
        "input": lambda x: x["input"],
        # Format agent scratchpad from
intermediate steps
        "agent_scratchpad": lambda x:
format_to_openai_function_messages(
            x["intermediate_steps"]
        ),
    }
    | prompt
    | llm_with_tools
    | parse
)
```

```
agent_executor = AgentExecutor(tools=
[retriever_tool], agent=agent, verbose=True)
```

Run the agent

We can now run the agent! Notice how it responds with a dictionary with two keys: `answer` and `sources`

```
agent_executor.invoke(
    {"input": "what did the president say about ketanji brown jackson"},
    return_only_outputs=True,
)
```

```



> Entering new AgentExecutor chain...
Tonight. I call on the Senate to: Pass the Freedom to Vote Act. Pass the John Lewis Voting Rights Act. And while you’re at it, pass the Disclose Act so Americans can know who is funding our elections.

Tonight, I’d like to honor someone who has dedicated his life to serve this country: Justice Stephen Breyer—an Army veteran, Constitutional scholar, and retiring Justice of the United States Supreme Court. Justice Breyer, thank you for your service.

One of the most serious constitutional responsibilities a President has is nominating someone to serve on the United States Supreme Court.
```

```
And I did that 4 days ago, when I nominated
Circuit Court of Appeals Judge Ketanji Brown
Jackson. One of our nation’s top legal minds,
who will continue Justice Breyer’s legacy of
excellence.

And for our LGBTQ+ Americans, let’s finally
get the bipartisan Equality Act to my desk.
The onslaught of state laws targeting
transgender Americans and their families is
wrong.

As I said last year, especially to our
younger transgender Americans, I will always
have your back as your President, so you can
be yourself and reach your God-given
potential.

While it often appears that we never agree,
that isn’t true. I signed 80 bipartisan bills
into law last year. From preventing
government shutdowns to protecting Asian-
Americans from still-too-common hate crimes
to reforming military justice.

And soon, we’ll strengthen the Violence
Against Women Act that I first wrote three
decades ago. It is important for us to show
the nation that we can come together and do
big things.
```

```
So tonight I’m offering a Unity Agenda for
the Nation. Four big things we can do
together.

First, beat the opioid epidemic.

Madam Speaker, Madam Vice President, our
First Lady and Second Gentleman. Members of
Congress and the Cabinet. Justices of the
Supreme Court. My fellow Americans.

Last year COVID-19 kept us apart. This year
we are finally together again.

Tonight, we meet as Democrats Republicans and
Independents. But most importantly as
Americans.

With a duty to one another to the American
people to the Constitution.

And with an unwavering resolve that freedom
will always triumph over tyranny.

Six days ago, Russia’s Vladimir Putin sought
to shake the foundations of the free world
thinking he could make it bend to his
menacing ways. But he badly miscalculated.

He thought he could roll into Ukraine and the
```

```
world would roll over. Instead he met a wall
of strength he never imagined.

He met the Ukrainian people.

From President Zelenskyy to every Ukrainian,
their fearlessness, their courage, their
determination, inspires the world.

A former top litigator in private practice. A
former federal public defender. And from a
family of public school educators and police
officers. A consensus builder. Since she’s
been nominated, she’s received a broad range
of support—from the Fraternal Order of Police
to former judges appointed by Democrats and
Republicans.

And if we are to advance liberty and justice,
we need to secure the Border and fix the
immigration system.

We can do both. At our border, we’ve
installed new technology like cutting-edge
scanners to better detect drug smuggling.

We’ve set up joint patrols with Mexico and
Guatemala to catch more human traffickers.

We’re putting in place dedicated immigration
judges so families fleeing persecution and
```

```
violence can have their cases heard faster.

We’re securing commitments and supporting
partners in South and Central America to host
more refugees and secure their own borders.
{'arguments': '{\n"answer": "President Biden
nominated Ketanji Brown Jackson for the
United States Supreme Court and described her
as one of our nation\'s top legal minds who
will continue Justice Breyer\'s legacy of
excellence.",\n"sources": [6]\n}', 'name':
'Response'}

> Finished chain.
```

```
{'answer': "President Biden nominated Ketanji
Brown Jackson for the United States Supreme
Court and described her as one of our
nation's top legal minds who will continue
Justice Breyer's legacy of excellence.",
 'sources': [6]}
```