

基本介绍

LangChain提供了一个回调系统，可以让你挂到LLM应用程序的各个阶段。这对记录、监控、流媒体和其他任务很有用。

你可以通过使用整个API中的callbacks参数来订阅这些事件。这个参数是处理程序对象的列表，这些对象应该实现下面详细描述的一个或多个方法。

回调处理程序

CallbackHandlers是实现CallbackHandler接口的对象，它对每个可以订阅的事件都有一个方法。当事件被触发时，CallbackManager将在每个处理程序上调用相应的方法。

```
class BaseCallbackHandler:
    """Base callback handler that can be used to handle callbacks from langchain."""

    def on_llm_start(
        self, serialized: Dict[str, Any], prompts: List[str], **kwargs: Any
    ) -> Any:
        """Run when LLM starts running."""

    def on_chat_model_start(
        self, serialized: Dict[str, Any], messages: List[List[BaseMessage]],
        **kwargs: Any
    ) -> Any:
        """Run when Chat Model starts running."""

    def on_llm_new_token(self, token: str, **kwargs: Any) -> Any:
        """Run on new LLM token. Only available when streaming is enabled."""

    def on_llm_end(self, response: LLMResult, **kwargs: Any) -> Any:
        """Run when LLM ends running."""

    def on_llm_error(
        self, error: Union[Exception, KeyboardInterrupt], **kwargs: Any
    ) -> Any:
        """Run when LLM errors."""

    def on_chain_start(
        self, serialized: Dict[str, Any], inputs: Dict[str, Any], **kwargs: Any
    ) -> Any:
        """Run when chain starts running."""

    def on_chain_end(self, outputs: Dict[str, Any], **kwargs: Any) -> Any:
        """Run when chain ends running."""

    def on_chain_error(
        self, error: Union[Exception, KeyboardInterrupt], **kwargs: Any
    ) -> Any:
        """Run when chain errors."""
```

```

def on_tool_start(
    self, serialized: Dict[str, Any], input_str: str, **kwargs: Any
) -> Any:
    """Run when tool starts running."""

def on_tool_end(self, output: str, **kwargs: Any) -> Any:
    """Run when tool ends running."""

def on_tool_error(
    self, error: Union[Exception, KeyboardInterrupt], **kwargs: Any
) -> Any:
    """Run when tool errors."""

def on_text(self, text: str, **kwargs: Any) -> Any:
    """Run on arbitrary text."""

def on_agent_action(self, action: AgentAction, **kwargs: Any) -> Any:
    """Run on agent action."""

def on_agent_finish(self, finish: AgentFinish, **kwargs: Any) -> Any:
    """Run on agent end."""

```

开始

LangChain提供了一些内置的处理程序，你可以用它们来入门。这些都可以在langchain/callbacks模块中找到。最基本的处理程序是StdOutCallbackHandler，它只是将所有事件记录到stdout。

请注意，当对象上的verbose标志被设置为 "true "时，即使没有明确传入，StdOutCallbackHandler也会被调用。

```

from langchain.callbacks import StdOutCallbackHandler
from langchain.chains import LLMChain
from langchain.llms import OpenAI
from langchain.prompts import PromptTemplate

handler = StdOutCallbackHandler()
llm = OpenAI()
prompt = PromptTemplate.from_template("1 + {number} = ")

# Constructor callback: First, let's explicitly set the StdOutCallbackHandler when
initializing our chain
chain = LLMChain(llm=llm, prompt=prompt, callbacks=[handler])
chain.run(number=2)

# Use verbose flag: Then, let's use the `verbose` flag to achieve the same result
chain = LLMChain(llm=llm, prompt=prompt, verbose=True)
chain.run(number=2)

# Request callbacks: Finally, let's use the request `callbacks` to achieve the same
result
chain = LLMChain(llm=llm, prompt=prompt)

```

```
chain.run(number=2, callbacks=[handler])
```

```
> Entering new LLMChain chain...
```

```
Prompt after formatting:
```

```
1 + 2 =
```

```
> Finished chain.
```

```
> Entering new LLMChain chain...
```

```
Prompt after formatting:
```

```
1 + 2 =
```

```
> Finished chain.
```

```
> Entering new LLMChain chain...
```

```
Prompt after formatting:
```

```
1 + 2 =
```

```
> Finished chain.
```

```
'\n\n3'
```

在哪里传入回调

回调参数在整个API的大多数对象（链、模型、工具、代理等）上有两个不同的地方：

- 构造函数回调：在构造函数中定义，例如LLMChain(callbacks=[handler], tags=['a-tag'])，它将被用于对该对象的所有调用，并且将只针对该对象，例如，如果你向LLMChain构造函数传递一个handler，它将被用于该链的所有调用，而不会被该链的Model使用。
- 请求回调：定义在用于发出请求的call()/run()/apply()方法中，例如chain.call(inputs, callbacks=[handler])，它将被用于该特定请求，以及它包含的所有子请求（例如，对LLMChain的调用会触发对Model的调用，该Model使用call()方法中传递的相同handler）。

verbose参数在整个API的大多数对象（链、模型、工具、代理等）上都可以作为构造参数使用，例如LLMChain(verbose=True)，它相当于将ConsoleCallbackHandler传递给该对象和所有子对象的callbacks参数。这对调试很有用，因为它将把所有事件记录到控制台。

你想在什么时候使用这些东西呢？

- 构造函数回调对诸如日志、监控等用例最有用，这些用例不是针对单个请求，而是针对整个链。例如，如果你想记录所有向LLMChain发出的请求，你可以向构造函数传递一个处理程序。
- 请求回调对流媒体等用例最有用，你想把单个请求的输出流到特定的websocket连接，或其他类似用例。例如，如果你想把单个请求的输出流到一个websocket，你会把一个处理程序传递给call()方法

异步回调

如果你打算使用异步API，建议使用AsyncCallbackHandler以避免阻塞运行循环。

如果你在使用异步方法运行你的llm/chain/tool/agent时使用同步CallbackHandler，它仍然可以工作。然而，在引擎盖下，它将被run_in_executor调用，如果你的CallbackHandler不是线程安全的，就会导致问题。

```
import asyncio
from typing import Any, Dict, List

from langchain.chat_models import ChatOpenAI
from langchain.schema import LLMResult, HumanMessage
from langchain.callbacks.base import AsyncCallbackHandler, BaseCallbackHandler

class MyCustomSyncHandler(BaseCallbackHandler):
    def on_llm_new_token(self, token: str, **kwargs) -> None:
        print(f"Sync handler being called in a `thread_pool_executor`: token: {token}")

class MyCustomAsyncHandler(AsyncCallbackHandler):
    """Async callback handler that can be used to handle callbacks from langchain."""

    async def on_llm_start(
        self, serialized: Dict[str, Any], prompts: List[str], **kwargs: Any
    ) -> None:
        """Run when chain starts running."""
        print("zzzz...")
        await asyncio.sleep(0.3)
        class_name = serialized["name"]
        print("Hi! I just woke up. Your llm is starting")

    async def on_llm_end(self, response: LLMResult, **kwargs: Any) -> None:
        """Run when chain ends running."""
        print("zzzz...")
        await asyncio.sleep(0.3)
        print("Hi! I just woke up. Your llm is ending")

# To enable streaming, we pass in `streaming=True` to the ChatModel constructor
# Additionally, we pass in a list with our custom handler
chat = ChatOpenAI(
    max_tokens=25,
    streaming=True,
    callbacks=[MyCustomSyncHandler(), MyCustomAsyncHandler()],
)

await chat.agenerate([[HumanMessage(content="Tell me a joke")]])

zzzz....
```

```
Hi! I just woke up. Your llm is starting
Sync handler being called in a `thread_pool_executor`: token:
Sync handler being called in a `thread_pool_executor`: token: why
Sync handler being called in a `thread_pool_executor`: token: don
Sync handler being called in a `thread_pool_executor`: token: 't
Sync handler being called in a `thread_pool_executor`: token: scientists
Sync handler being called in a `thread_pool_executor`: token: trust
Sync handler being called in a `thread_pool_executor`: token: atoms
Sync handler being called in a `thread_pool_executor`: token: ?
Sync handler being called in a `thread_pool_executor`: token:
```

```
Sync handler being called in a `thread_pool_executor`: token: Because
Sync handler being called in a `thread_pool_executor`: token: they
Sync handler being called in a `thread_pool_executor`: token: make
Sync handler being called in a `thread_pool_executor`: token: up
Sync handler being called in a `thread_pool_executor`: token: everything
Sync handler being called in a `thread_pool_executor`: token: .
Sync handler being called in a `thread_pool_executor`: token:
zzzz....
```

```
Hi! I just woke up. Your llm is ending
```

```
LLMResult(generations=[[ChatGeneration(text="why don't scientists trust atoms?
\n\nBecause they make up everything.", generation_info=None,
message=AIMessage(content="why don't scientists trust atoms? \n\nBecause they make
up everything.", additional_kwargs={}, example=False))]], llm_output={'token_usage':
{}, 'model_name': 'gpt-3.5-turbo'})
```

定制回调处理器

你也可以创建一个自定义的处理程序来设置在该对象上。在下面的例子中，我们将用一个自定义处理程序来实现流媒体。

```
from langchain.callbacks.base import BaseCallbackHandler
from langchain.chat_models import ChatOpenAI
from langchain.schema import HumanMessage

class MyCustomHandler(BaseCallbackHandler):
    def on_llm_new_token(self, token: str, **kwargs) -> None:
        print(f"My custom handler, token: {token}")

# To enable streaming, we pass in `streaming=True` to the ChatModel constructor
# Additionally, we pass in a list with our custom handler
chat = ChatOpenAI(max_tokens=25, streaming=True, callbacks=[MyCustomHandler()])
```

```
chat([HumanMessage(content="Tell me a joke")])
```

```
My custom handler, token:
My custom handler, token: why
My custom handler, token: don
My custom handler, token: 't
My custom handler, token: scientists
My custom handler, token: trust
My custom handler, token: atoms
My custom handler, token: ?
My custom handler, token:
```

```
My custom handler, token: Because
My custom handler, token: they
My custom handler, token: make
My custom handler, token: up
My custom handler, token: everything
My custom handler, token: .
My custom handler, token:
```

```
AIMessage(content="why don't scientists trust atoms? \n\nBecause they make up everything.", additional_kwargs={}, example=False)
```

定制链中使用回调

当你创建一个自定义链时，你可以很容易地将其设置为使用与所有内置链相同的回调系统。链/LLM/聊天模型/代理/工具上的`call`、`generate`、`_run`和等效的异步方法现在会接收一个名为`run_manager`的第二个参数，该参数与该运行绑定，并包含该对象可使用的日志方法（即`on_llm_new_token`）。这在构建自定义链的时候很有用。关于如何创建自定义链并在其中使用回调的更多信息，请参见本指南。https://python.langchain.com/docs/modules/chains/how_to/custom_chain.html

记录日志到文件

这个例子展示了如何打印日志到文件。它展示了如何使用`FileCallbackHandler`，它与`StdOutCallbackHandler`做同样的事情，但却将输出写入文件。它还使用`loguru`库来记录处理程序未捕获的其他输出。

```
from loguru import logger

from langchain.callbacks import FileCallbackHandler
from langchain.chains import LLMChain
from langchain.llms import OpenAI
from langchain.prompts import PromptTemplate
```

```

logfile = "output.log"

logger.add(logfile, colorize=True, enqueue=True)
handler = FileCallbackHandler(logfile)

llm = OpenAI()
prompt = PromptTemplate.from_template("1 + {number} = ")

# this chain will both print to stdout (because verbose=True) and write to
# 'output.log'
# if verbose=False, the FileCallbackHandler will still write to 'output.log'
chain = LLMChain(llm=llm, prompt=prompt, callbacks=[handler], verbose=True)
answer = chain.run(number=2)
logger.info(answer)

> Entering new LLMChain chain...
Prompt after formatting:
1 + 2 =

•[32m2023-06-01 18:36:38.929•[0m | •[1mINFO      •[0m |
•[36m__main__•[0m:•[36m<module>•[0m:•[36m20•[0m - •[1m

3•[0m

> Finished chain.

```

我们可以打开output.log文件，看到输出已经被捕获。

```

pip install ansi2html > /dev/null

from IPython.display import display, HTML
from ansi2html import Ansi2HTMLConverter

with open("output.log", "r") as f:
    content = f.read()

conv = Ansi2HTMLConverter()
html = conv.convert(content, full=True)

display(HTML(html))

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>

```

```

<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title></title>
<style type="text/css">
.ansi2html-content { display: inline; white-space: pre-wrap; word-wrap: break-word;
}
.body_foreground { color: #AAAAAA; }
.body_background { background-color: #000000; }
.inv_foreground { color: #000000; }
.inv_background { background-color: #AAAAAA; }
.ansi1 { font-weight: bold; }
.ansi3 { font-style: italic; }
.ansi32 { color: #00aa00; }
.ansi36 { color: #00aaaa; }
</style>
</head>
<body class="body_foreground body_background" style="font-size: normal;" >
<pre class="ansi2html-content">

<span class="ansi1">&gt; Entering new LLMChain chain...</span>
Prompt after formatting:
<span class="ansi1 ansi32"></span><span class="ansi1 ansi3 ansi32">1 + 2 = </span>

<span class="ansi1">&gt; Finished chain.</span>
<span class="ansi32">2023-06-01 18:36:38.929</span> | <span class="ansi1">INFO
  </span> | <span class="ansi36">__main__</span>:<span class="ansi36">&lt;module>
</span>:<span class="ansi36">20</span> - <span class="ansi1">

3</span>

</pre>
</body>

</html>

```

多个回调处理程序

在前面的例子中，我们通过使用callbacks=在创建对象时传入回调处理程序。在这种情况下，回调将被扩展到该特定对象。

然而，在许多情况下，在运行对象时传入处理程序反而是有利的。当我们在执行一个运行时使用callbacks关键字arg传递CallbackHandlers时，这些回调将由所有参与执行的嵌套对象发出。例如，当一个处理程序被传递给一个代理时，它将被用于所有与代理有关的回调，以及所有参与代理执行的对象，在这种情况下，工具、LLMChain和LLM。

这就避免了我们手动将处理程序附加到每个单独的嵌套对象。

```

from typing import Dict, Union, Any, List

```



```

from langchain.callbacks.base import BaseCallbackHandler
from langchain.schema import AgentAction
from langchain.agents import AgentType, initialize_agent, load_tools
from langchain.callbacks import tracing_enabled
from langchain.llms import OpenAI

# First, define custom callback handler implementations
class MyCustomHandlerOne(BaseCallbackHandler):
    def on_llm_start(
        self, serialized: Dict[str, Any], prompts: List[str], **kwargs: Any
    ) -> Any:
        print(f"on_llm_start {serialized['name']}")

    def on_llm_new_token(self, token: str, **kwargs: Any) -> Any:
        print(f"on_new_token {token}")

    def on_llm_error(
        self, error: Union[Exception, KeyboardInterrupt], **kwargs: Any
    ) -> Any:
        """Run when LLM errors."""

    def on_chain_start(
        self, serialized: Dict[str, Any], inputs: Dict[str, Any], **kwargs: Any
    ) -> Any:
        print(f"on_chain_start {serialized['name']}")

    def on_tool_start(
        self, serialized: Dict[str, Any], input_str: str, **kwargs: Any
    ) -> Any:
        print(f"on_tool_start {serialized['name']}")

    def on_agent_action(self, action: AgentAction, **kwargs: Any) -> Any:
        print(f"on_agent_action {action}")

class MyCustomHandlerTwo(BaseCallbackHandler):
    def on_llm_start(
        self, serialized: Dict[str, Any], prompts: List[str], **kwargs: Any
    ) -> Any:
        print(f"on_llm_start (I'm the second handler!!) {serialized['name']}")

# Instantiate the handlers
handler1 = MyCustomHandlerOne()
handler2 = MyCustomHandlerTwo()

# Setup the agent. Only the `llm` will issue callbacks for handler2
llm = OpenAI(temperature=0, streaming=True, callbacks=[handler2])
tools = load_tools(["llm-math"], llm=llm)
agent = initialize_agent(tools, llm, agent=AgentType.ZERO_SHOT_REACT_DESCRIPTION)

```

```

# Callbacks for handler1 will be issued by every object involved in the
# Agent execution (llm, llmchain, tool, agent executor)
agent.run("What is 2 raised to the 0.235 power?", callbacks=[handler1])

on_chain_start AgentExecutor
on_chain_start LLMChain
on_llm_start OpenAI
on_llm_start (I'm the second handler!!) OpenAI
on_new_token I
on_new_token need
on_new_token to
on_new_token use
on_new_token a
on_new_token calculator
on_new_token to
on_new_token solve
on_new_token this
on_new_token .
on_new_token
Action
on_new_token :
on_new_token Calculator
on_new_token
Action
on_new_token Input
on_new_token :
on_new_token 2
on_new_token ^
on_new_token 0
on_new_token .
on_new_token 235
on_new_token
on_agent_action AgentAction(tool='Calculator', tool_input='2^0.235', log=' I
need to use a calculator to solve this.\nAction: Calculator\nAction Input: 2^0.235')
on_tool_start Calculator
on_chain_start LLMMathChain
on_chain_start LLMChain
on_llm_start OpenAI
on_llm_start (I'm the second handler!!) OpenAI
on_new_token
on_new_token ```text
on_new_token

on_new_token 2
on_new_token **
on_new_token 0
on_new_token .
on_new_token 235
on_new_token

on_new_token ```

```

```
on_new_token ...
on_new_token num
on_new_token expr
on_new_token .
on_new_token evaluate
on_new_token ("
on_new_token 2
on_new_token **
on_new_token 0
on_new_token .
on_new_token 235
on_new_token ")
on_new_token ...
on_new_token

on_new_token
on_chain_start LLMChain
on_llm_start OpenAI
on_llm_start (I'm the second handler!!) OpenAI
on_new_token I
on_new_token now
on_new_token know
on_new_token the
on_new_token final
on_new_token answer
on_new_token .
on_new_token
Final
on_new_token Answer
on_new_token :
on_new_token 1
on_new_token .
on_new_token 17
on_new_token 690
on_new_token 67
on_new_token 372
on_new_token 187
on_new_token 674
on_new_token
```

'1.1769067372187674'

标签

你可以通过向call()/run()/apply()方法传递一个标签参数来为你的回调添加标签。这对于过滤你的日志很有用，例如，如果你想记录所有对特定LLMChain的请求，你可以添加一个标签，然后通过该标签过滤你的日志。你可以将标签传递给构造函数和请求回调，详情请见上面的例子。这些标签然后被传递到 "开始 "回调方法的标签参数中，即on_llm_start、on_chat_model_start、on_chain_start、on_tool_start。

token计数

LangChain提供了一个上下文管理器，可以让你计算令牌。

```
import asyncio

from langchain.callbacks import get_openai_callback
from langchain.llms import OpenAI

llm = OpenAI(temperature=0)
with get_openai_callback() as cb:
    llm("What is the square root of 4?")

total_tokens = cb.total_tokens
assert total_tokens > 0

with get_openai_callback() as cb:
    llm("What is the square root of 4?")
    llm("What is the square root of 4?")

assert cb.total_tokens == total_tokens * 2

# You can kick off concurrent runs from within the context manager
with get_openai_callback() as cb:
    await asyncio.gather(
        *[llm.agenerate(["What is the square root of 4?"]) for _ in range(3)]
    )

assert cb.total_tokens == total_tokens * 3

# The context manager is concurrency safe
task = asyncio.create_task(llm.agenerate(["What is the square root of 4?"]))
with get_openai_callback() as cb:
    await llm.agenerate(["What is the square root of 4?"])

await task
assert cb.total_tokens == total_tokens
```

追踪

有两种推荐的方法来追踪你的LangChains：

- 将LANGCHAIN_TRACING环境变量设置为 "true"。
- 使用带有tracing_enabled()的上下文管理器来追踪某个特定的代码块。

注意，如果环境变量被设置，所有的代码都会被追踪，不管它是否在上下文管理器中。

```

import os

from langchain.agents import AgentType, initialize_agent, load_tools
from langchain.callbacks import tracing_enabled
from langchain.llms import OpenAI

# To run the code, make sure to set OPENAI_API_KEY and SERPAPI_API_KEY
llm = OpenAI(temperature=0)
tools = load_tools(["llm-math", "serpapi"], llm=llm)
agent = initialize_agent(
    tools, llm, agent=AgentType.ZERO_SHOT_REACT_DESCRIPTION, verbose=True
)

questions = [
    "Who won the US Open men's final in 2019? What is his age raised to the 0.334 power?",
    "Who is Olivia Wilde's boyfriend? What is his current age raised to the 0.23 power?",
    "Who won the most recent Formula 1 Grand Prix? What is their age raised to the 0.23 power?",
    "Who won the US Open women's final in 2019? What is her age raised to the 0.34 power?",
    "Who is Beyoncé's husband? What is his age raised to the 0.19 power?",
]
os.environ["LANGCHAIN_TRACING"] = "true"

# Both of the agent runs will be traced because the environment variable is set
agent.run(questions[0])
with tracing_enabled() as session:
    assert session
    agent.run(questions[1])

WARNING:root:Failed to load default session, using empty session:
HTTPConnectionPool(host='localhost', port=8000): Max retries exceeded with url: /sessions?name=default (Caused by
NewConnectionError('<urllib3.connection.HTTPConnection object at 0x12f8b36d0>: Failed to establish a new connection: [Errno 61] Connection refused'))

> Entering new AgentExecutor chain...
I need to find out who won the US Open men's final in 2019 and then calculate his age raised to the 0.334 power.
Action: Search
Action Input: "US Open men's final 2019 winner"
Observation: Rafael Nadal defeated Daniil Medvedev in the final, 7-5, 6-3, 5-7, 4-6, 6-4 to win the men's singles tennis title at the 2019 US Open. It was his fourth US ...
Thought: I need to find out the age of the winner
Action: Search
Action Input: "Rafael Nadal age"

```

Observation: 37 years
Thought: I now need to calculate the age raised to the 0.334 power
Action: Calculator
Action Input: $37^{0.334}$
Observation: Answer: 3.340253100876781
Thought:

WARNING:root:Failed to persist run: HTTPConnectionPool(host='localhost', port=8000): Max retries exceeded with url: /chain-runs (Caused by NewConnectionError('<urllib3.connection.HTTPConnection object at 0x12f8c0f50>: Failed to establish a new connection: [Errno 61] Connection refused'))

WARNING:root:Failed to load default session, using empty session: HTTPConnectionPool(host='localhost', port=8000): Max retries exceeded with url: /sessions?name=default (Caused by NewConnectionError('<urllib3.connection.HTTPConnection object at 0x12f8e6f50>: Failed to establish a new connection: [Errno 61] Connection refused'))

I now know the final answer

Final Answer: Rafael Nadal, aged 37, won the US Open men's final in 2019 and his age raised to the 0.334 power is 3.340253100876781.

> Finished chain.

> Entering new AgentExecutor chain...

I need to find out who Olivia wilde's boyfriend is and then calculate his age raised to the 0.23 power.

Action: Search

Action Input: "Olivia wilde boyfriend"

Observation: Sudeikis and wilde's relationship ended in November 2020. wilde was publicly served with court documents regarding child custody while she was presenting Don't Worry Darling at CinemaCon 2022. In January 2021, wilde began dating singer Harry Styles after meeting during the filming of Don't Worry Darling.

Thought: I need to find out Harry Styles' age.

Action: Search

Action Input: "Harry Styles age"

Observation: 29 years

Thought: I need to calculate 29 raised to the 0.23 power.

Action: Calculator

Action Input: $29^{0.23}$

Observation: Answer: 2.169459462491557

Thought:

WARNING:root:Failed to persist run: HTTPConnectionPool(host='localhost', port=8000): Max retries exceeded with url: /chain-runs (Caused by NewConnectionError('<urllib3.connection.HTTPConnection object at 0x12f8fa590>: Failed to establish a new connection: [Errno 61] Connection refused'))

I now know the final answer.

Final Answer: Harry Styles is Olivia Wilde's boyfriend and his current age raised to the 0.23 power is 2.169459462491557.

> Finished chain.

Now, we unset the environment variable and use a context manager.

```
if "LANGCHAIN_TRACING" in os.environ:  
    del os.environ["LANGCHAIN_TRACING"]
```

```
# here, we are writing traces to "my_test_session"  
with tracing_enabled("my_test_session") as session:  
    assert session  
    agent.run(questions[0]) # this should be traced
```

```
agent.run(questions[1]) # this should not be traced
```

WARNING:root:Failed to load my_test_session session, using empty session:
HTTPConnectionPool(host='localhost', port=8000): Max retries exceeded with url: /sessions?name=my_test_session (Caused by
NewConnectionError('<urllib3.connection.HTTPConnection object at 0x12f8e41d0>:
Failed to establish a new connection: [Errno 61] Connection refused'))

> Entering new AgentExecutor chain...

I need to find out who won the US Open men's final in 2019 and then calculate his age raised to the 0.334 power.

Action: Search

Action Input: "US Open men's final 2019 winner"

Observation: Rafael Nadal defeated Daniil Medvedev in the final, 7-5, 6-3, 5-7, 4-6, 6-4 to win the men's singles tennis title at the 2019 US Open. It was his fourth US ...

Thought: I need to find out the age of the winner

Action: Search

Action Input: "Rafael Nadal age"

Observation: 37 years

Thought: I now need to calculate the age raised to the 0.334 power

Action: Calculator

Action Input: $37^{0.334}$

Observation: Answer: 3.340253100876781

Thought:

WARNING:root:Failed to persist run: HTTPConnectionPool(host='localhost', port=8000): Max retries exceeded with url: /chain-runs (Caused by
NewConnectionError('<urllib3.connection.HTTPConnection object at 0x12f8d0a50>:
Failed to establish a new connection: [Errno 61] Connection refused'))

I now know the final answer

Final Answer: Rafael Nadal, aged 37, won the US Open men's final in 2019 and his age raised to the 0.334 power is 3.340253100876781.

> Finished chain.

> Entering new AgentExecutor chain...

I need to find out who Olivia wilde's boyfriend is and then calculate his age raised to the 0.23 power.

Action: Search

Action Input: "Olivia wilde boyfriend"

Observation: Sudeikis and wilde's relationship ended in November 2020. wilde was publicly served with court documents regarding child custody while she was presenting Don't Worry Darling at CinemaCon 2022. In January 2021, wilde began dating singer Harry Styles after meeting during the filming of Don't Worry Darling.

Thought: I need to find out Harry Styles' age.

Action: Search

Action Input: "Harry Styles age"

Observation: 29 years

Thought: I need to calculate 29 raised to the 0.23 power.

Action: Calculator

Action Input: $29^{0.23}$

Observation: Answer: 2.169459462491557

Thought: I now know the final answer.

Final Answer: Harry Styles is Olivia wilde's boyfriend and his current age raised to the 0.23 power is 2.169459462491557.

> Finished chain.

"Harry Styles is Olivia wilde's boyfriend and his current age raised to the 0.23 power is 2.169459462491557."

```
import asyncio

# The context manager is concurrency safe:
if "LANGCHAIN_TRACING" in os.environ:
    del os.environ["LANGCHAIN_TRACING"]

# start a background task
task = asyncio.create_task(agent.arun(questions[0])) # this should not be traced
with tracing_enabled() as session:
    assert session
    tasks = [agent.arun(q) for q in questions[1:3]] # these should be traced
    await asyncio.gather(*tasks)

await task
```



```
WARNING:root:Failed to load default session, using empty session:
HTTPConnectionPool(host='localhost', port=8000): Max retries exceeded with url:
/sessions?name=default (Caused by
NewConnectionError('<urllib3.connection.HTTPConnection object at 0x12f916ed0>:
Failed to establish a new connection: [Errno 61] Connection refused'))
```

> Entering new AgentExecutor chain...

> Entering new AgentExecutor chain...

> Entering new AgentExecutor chain...

I need to find out who Olivia wilde's boyfriend is and then calculate his age raised to the 0.23 power.

Action: Search

Action Input: "Olivia wilde boyfriend" I need to find out who won the grand prix and then calculate their age raised to the 0.23 power.

Action: Search

Action Input: "Formula 1 Grand Prix winner" I need to find out who won the US Open men's final in 2019 and then calculate his age raised to the 0.334 power.

Action: Search

Action Input: "US Open men's final 2019 winner"

Observation: Sudeikis and wilde's relationship ended in November 2020. wilde was publicly served with court documents regarding child custody while she was presenting Don't Worry Darling at CinemaCon 2022. In January 2021, wilde began dating singer Harry Styles after meeting during the filming of Don't Worry Darling.

Thought:

Observation: Rafael Nadal defeated Daniil Medvedev in the final, 7-5, 6-3, 5-7, 4-6, 6-4 to win the men's singles tennis title at the 2019 US Open. It was his fourth US ...

Thought:

Observation: The first Formula One world Drivers' Champion was Giuseppe Farina in the 1950 championship and the current title holder is Max Verstappen in the 2022 season.

Thought: I need to find out Harry Styles' age.

Action: Search

Action Input: "Harry Styles age" I need to find out the age of the winner

Action: Search

Action Input: "Rafael Nadal age"

Observation: 29 years

Thought:

Observation: 37 years

Thought: I need to find out Max Verstappen's age.

Action: Search

Action Input: "Max Verstappen Age" I need to calculate 29 raised to the 0.23 power.

Action: Calculator

Action Input: $29^{0.23}$ I now need to calculate the age raised to the 0.334 power

Action: Calculator
Action Input: $37^{0.334}$
Observation: Answer: 2.169459462491557
Thought:
Observation: 25 years
Thought:
Observation: Answer: 3.340253100876781
Thought:

WARNING:root:Failed to persist run: HTTPConnectionPool(host='localhost', port=8000): Max retries exceeded with url: /chain-runs (Caused by NewConnectionError('<urllib3.connection.HTTPConnection object at 0x12f95dbd0>: Failed to establish a new connection: [Errno 61] Connection refused'))

I now know the final answer.
Final Answer: Harry Styles is olivia wilde's boyfriend and his current age raised to the 0.23 power is 2.169459462491557.

> Finished chain.
I need to calculate 25 raised to the 0.23 power.
Action: Calculator
Action Input: $25^{0.23}$ I now know the final answer
Final Answer: Rafael Nadal, aged 37, won the US Open men's final in 2019 and his age raised to the 0.334 power is 3.340253100876781.

> Finished chain.

Observation: Answer: 2.096651272316035
Thought:

WARNING:root:Failed to persist run: HTTPConnectionPool(host='localhost', port=8000): Max retries exceeded with url: /chain-runs (Caused by NewConnectionError('<urllib3.connection.HTTPConnection object at 0x12f95de50>: Failed to establish a new connection: [Errno 61] Connection refused'))

I now know the final answer.
Final Answer: Max Verstappen, aged 25, won the most recent Formula 1 Grand Prix and his age raised to the 0.23 power is 2.096651272316035.

> Finished chain.

"Rafael Nadal, aged 37, won the US Open men's final in 2019 and his age raised to the 0.334 power is 3.340253100876781."

集成

一些相关的例子，这里不展开了，可自行查看：<https://python.langchain.com/docs/modules/callbacks/integrations/>

总结

回调一般是在程序的某个阶段执行某种特定的操作，这在许多不同的程序之中都很有用。到这里，就把langchain文档中的基本知识过了一遍。不仅仅是对于langchain，其思想也可以借鉴于我们自己开发LLM程序，比如数据接入、prompt管理、各种NLP任务、智能体选择不同的任务、错误处理、日志追踪、历史信息记录、数据输出等。

学习基础知识是枯燥的，而且包含的东西太多了，我们需要在实践中了解和巩固相关知识。当然，上述都是针对于英文的，或者说是针对于openai chatgpt的，如何将langchain应用到中文大语言模型中也是我们接下来要学习的。