Thực hành bộ nhớ ngắn hạn

Trong bài học này, chúng ta sẽ học cách thêm bộ nhớ ngắn hạn vào agent của mình.

1. Cài đặt

Tại console:

```
* cd project_name
* pyenv local 3.11.4
* poetry install
* poetry shell
* jupyter lab
```

2. Tạo file .env

```
* OPENAI_API_KEY=your_openai_api_key
* LANGCHAIN_TRACING_V2=true
* LANGCHAIN_ENDPOINT=https://api.smith.langchain.com
* LANGCHAIN_API_KEY=your_langchain_api_key
* LANGCHAIN_PROJECT=your_project_name
```

3. Kết nối với tệp .env nằm trong cùng thư mục của notebook

```
#pip install python-dotenv
```

```
import os
from dotenv import load_dotenv, find_dotenv
_ = load_dotenv(find_dotenv())
openai_api_key = os.environ["OPENAI_API_KEY"]
```

3. Cài đặt LangChain

```
#!pip install langchain
#!pip install langchain-openai
```

```
from langchain_openai import ChatOpenAI

chatModel35 = ChatOpenAI(model="gpt-3.5-turbo-0125")
chatModel4o = ChatOpenAI(model="gpt-4o")
```

4. LLM kết hợp với nhiều công cụ hỗ trợ

```
from langchain_openai import ChatOpenAI
from langchain.tools import tool

@tool
def multiply(a: int, b: int) -> int:
    """Multiply a and b.

    Args:
        a: first int
        b: second int
    """
    return a * b

@tool
def add(a: int, b: int) -> int:
    """Adds a and b.

    Args:
        a: first int
        b: second int
    """
    return a + b

@tool
def divide(a: int, b: int) -> float:
    """Divide a and b.

    Args:
        a: first int
        b: second int
    """
    return a / b
```

```
tools = [add, multiply, divide]
llm_with_tools = chatModel4o.bind_tools(tools, parallel_tool_calls=False)
```

parallel_tool_calls=True

Đây là **gọi công cụ song song** (chạy nhiều công cụ cùng lúc).

- **Ý tưởng:** Nếu mô hình cần gọi nhiều công cụ để trả lời người dùng, nó có thể gọi tất cả các công cụ cùng một lúc, không cần chờ từng cái chạy xong.
- **Ưu điểm:** Nhanh hơn! Vì các công cụ chạy đồng thời.
- **Khi nào nên dùng:** Khi các công cụ **không phụ thuộc** vào kết quả của nhau. Ví dụ: bạn hỏi đồng thời "thời tiết ở TP. Hồ Chí Minh?" và "giá Bitcoin hôm nay?" — hai việc này độc lập nhau.

parallel_tool_calls=False

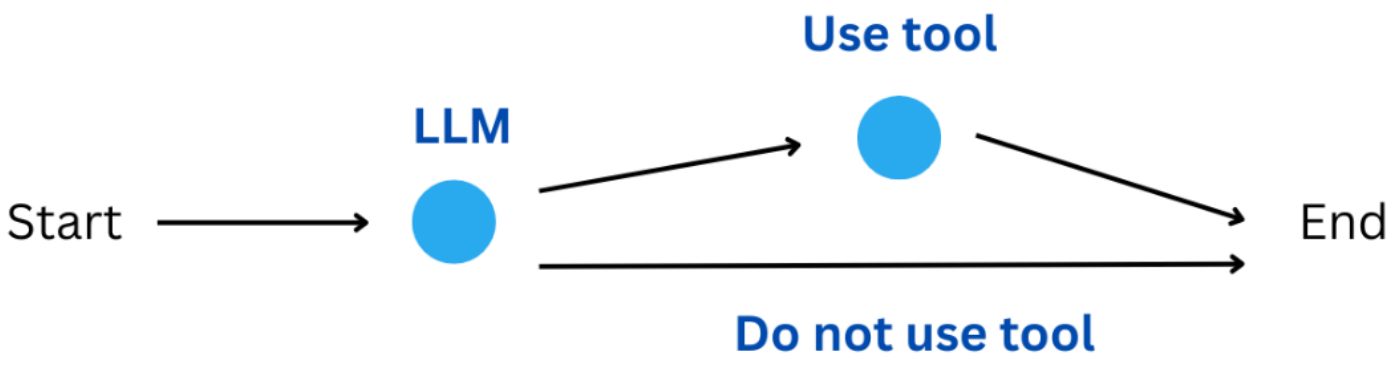
Đây là **gọi công cụ tuần tự** (chạy lần lượt từng cái một).

- **Ý tưởng:** Mỗi công cụ được gọi **theo thứ tự**, cái sau có thể dùng kết quả của cái trước.
- **Ưu điểm:** Giữ đúng logic nếu cần kết quả từng bước.
- **Khi nào nên dùng:** Khi các công cụ **phụ thuộc lẫn nhau**. Ví dụ:
  - Bước 1: Tính tổng
  - Bước 2: Dùng kết quả tổng để chia 2
    → Phải làm từng bước theo thứ tự.

Chi tiết tại

https://python.langchain.com/docs/how_to/tool_calling_parallel/

5. Xây dựng một ứng dụng (gọi là đồ thị trong langgraph) quyết định xem có nên trò chuyện bằng LLM hay sử dụng công cụ



5.1 Định nghĩa State schema

```
from langgraph.graph import MessagesState

class MessagesState(MessagesState):
    # Add any keys needed beyond messages, which is pre-built
    pass
```

5.2 Định nghĩa Node đầu tiên

```
from langgraph.graph import MessagesState
from langchain_core.messages import HumanMessage, SystemMessage

# Khai báo System message
sys_msg = SystemMessage(content="Bạn là trợ lý hữu ích có nhiệm vụ thực hiện phép tính số học trên một tập hợp dữ liệu đầu vào.")

# Định nghĩa Node
def assistant(state: MessagesState):
   return {"messages": [llm_with_tools.invoke([sys_msg] + state["messages"])]}
```

[sys_msg] + state["messages"]

- sys_msg là 1 **SystemMessage** định hướng vai trò AI, ví dụ: SystemMessage(content="Bạn là trợ lý toán học.")
- `state["messages"]`: là các tin nhắn trước đó (giữa người dùng và AI).

5.3. Kết hợp các Node và Edge để xây dựng Graph

```
from langgraph.graph import START, StateGraph
from langgraph.prebuilt import tools_condition
from langgraph.prebuilt import ToolNode
from IPython.display import Image, display

# Build graph
builder = StateGraph(MessagesState)

builder.add_node("assistant", assistant)
builder.add_node("tools", ToolNode(tools))

# Add the logic of the graph
builder.add_edge(START, "assistant")
```

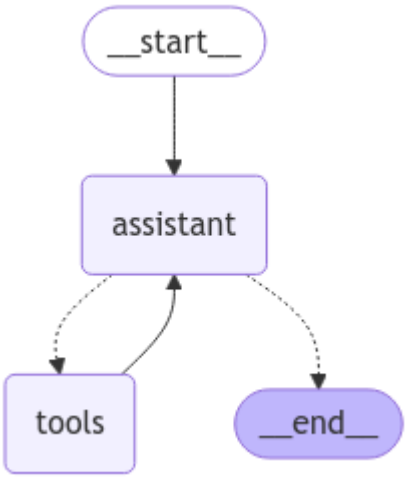
```
builder.add_conditional_edges(
    "assistant",
    # Nếu câu trả lời gần nhất của assistant là một lời gọi tới tool, Thì điều kiện tools_condition sẽ quyết định đi tiếp đến node tools
    # Ngược lại, tools_condition sẽ quyết định chuyển sang node END (kết thúc luồng, trả kết quả ra ngoài).
    tools_condition,
)

# PAY ATTENTION HERE: from the tools node, back to the assistant
builder.add_edge("tools", "assistant")

# PAY ATTENTION: No END edgge.

# Compile the graph
react_graph = builder.compile()

# Visualize the graph
display(Image(react_graph.get_graph(xray=True).draw_mermaid_png()))
```



5.4 Chạy ứng dụng

```
messages = [HumanMessage(content="Add 3 and 4.")]

messages = react_graph.invoke({"messages": messages})

for m in messages['messages']:
    m.pretty_print()
```

================================ Human Message
================================== Add 3 and 4.
================================== Ai Message
================================== Tool Calls: add
(call_AKxpCcslEcimrNFvBEWZ2Ru3) Call ID: call_AKxpCcslEcimrNFvBEWZ2Ru3 Args: a: 3 b: 4
================================ Tool Message
================================== Name: add 7
================================== Ai Message
================================== The sum of 3 and 4 is 7.

5.5 Tiếp tục hỏi Agent

```
messages = [HumanMessage(content="Multiply that by 2.")]


messages = react_graph.invoke({"messages": messages})


for m in messages['messages']:
    m.pretty_print()
```

================================ Human Message
================================== Multiply that by 2.
================================== Ai Message
================================== I need a specific number to multiply by 2.
Could you please provide the number you want to multiply?

> *Như bạn thấy, Agent yêu cầu cung cấp một số cụ thể để thực hiện phép tính nhân, nó không nhớ được kết quả của phép tính cộng đã thực hiện ở trên*

6. Thêm bộ nhớ

```
from langgraph.checkpoint.memory import MemorySaver


memory = MemorySaver()


react_graph_memory = builder.compile(checkpointer=memory)
```

```
# PAY ATTENTION HERE: see how we specify a thread
config = {"configurable": {"thread_id": "1"}}


# Enter the input
messages = [HumanMessage(content="Add 3 and 4.")]
```

```
# PAY ATTENTION HERE: see how we add config to referr to the thread_id
messages = react_graph_memory.invoke({"messages": messages},config)

for m in messages['messages']:
    m.pretty_print()
```

Kết quả

```
================================ Human Message
================================ Add 3 and 4.
================================== Ai Message
================================== Tool Calls: add
(call_7lH6zvBw4rMeTG3Zu66i53eW) Call ID: call_7lH6zvBw4rMeTG3Zu66i53eW Args: a: 3 b: 4
================================ Tool Message
================================ Name: add 7
================================== Ai Message
================================== The sum of 3 and 4 is 7.
```

Tiếp tục yêu cầu phép tính nhân

```
# PAY ATTENTION HERE: see how we check if the app has memory
messages = [HumanMessage(content="Multiply that by 2.")]

# Again, see how we use config here to referr to the thread_id
messages = react_graph_memory.invoke({"messages": messages}, config)

for m in messages['messages']:
    m.pretty_print()
```

```
================================ Human Message
================================ Multiply that by 2.
================================== Ai Message
================================== Tool Calls: multiply
(call_x3tKpGBsTHUCbVwzBr3PiNn1) Call ID: call_x3tKpGBsTHUCbVwzBr3PiNn1 Args:

...
14 ================================== Ai Message
================================== The result of multiplying 7 by 2 is 14.
```

Agent đã nhớ được kết quả của phép tính cộng và thực hiện phép tính nhân

Tác giả: Đỗ Ngọc Tú

Công Ty Phần Mềm VHTSoft

Phiên bản #3
Được tạo 21 tháng 4 2025 02:16:32 bởi Đỗ Ngọc Tú
Được cập nhật 23 tháng 4 2025 09:41:54 bởi Đỗ Ngọc Tú