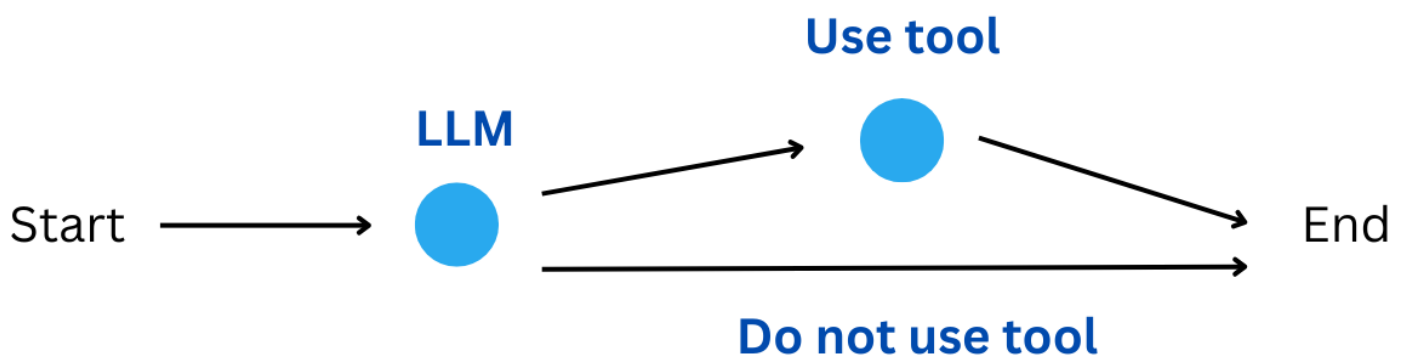


# Xây dựng một ứng dụng quyết định xem có nên trò chuyện bằng LLM hay sử dụng công cụ



## 1. Cài đặt

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

## 2. Tạo .env file

```
* 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 file .env

```
#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"]
```

#### 4. 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")
```

#### 5. LLM với các công cụ

```
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)
```

## Define the State schema

```
from langgraph.graph import MessagesState

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

## Define the first Node

```
from langchain_core.messages import HumanMessage, SystemMessage

# System message, system prompt
sys_msg = SystemMessage(content="You are a helpful assistant tasked with performing arithmetic on a set of inputs.")

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

\* Các nút của đồ thị được định nghĩa là các hàm python.

\* Đối số đầu tiên của hàm Node là trạng thái. Do đó, trong bài tập này, mỗi nút có thể truy cập khóa `messages`, với `state['messages']`.

\* Trong ví dụ này, chúng ta sẽ bắt đầu bằng một nút tương tự như nút chúng ta đã tạo trong bài tập trước, \*\*nhưng lần này bằng SystemMessage\*\*:

## Kết hợp các nút và cạnh để xây dựng đồ thị

```

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",
    # If the latest message (result) from assistant is a tool call -> tools_condition routes to tools
    # If the latest message (result) from assistant is a not a tool call -> tools_condition routes to END
    tools_condition,
)

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

# PAY ATTENTION: No END edge.

# Compile the graph
react_graph = builder.compile()

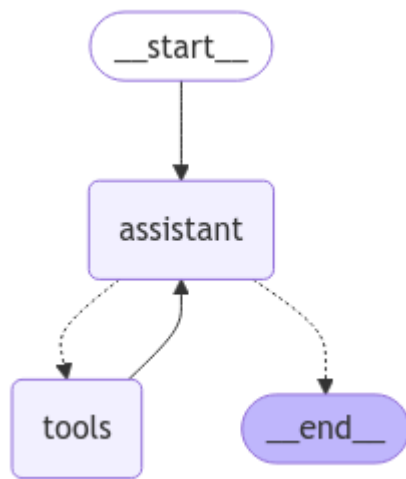
# Visualize the graph
display(Image(react_graph.get_graph(xray=True).draw_mermaid_png()))

```

\* act - LLM gọi một công cụ.

\* observer - công cụ chuyển đầu ra trở lại LLM.

\* reason - LLM quyết định phải làm gì tiếp theo (gọi một công cụ khác hoặc phản hồi trực tiếp).



## Chạy ứng dụng

```
messages = [HumanMessage(content="What was the relationship between Marilyn and JFK?")]
messages = react_graph.invoke({"messages": messages})
```

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

```
===== Human Message
=====
```

What was the relationship between Marilyn and JFK?

```
===== Ai Message
=====
```

Marilyn Monroe and President John F. Kennedy (JFK) are often rumored to have had a romantic relationship, though concrete details about the nature and extent of their relationship remain speculative and are part of popular lore. The most famous public connection between them was Marilyn Monroe's sultry performance of "Happy Birthday, Mr. President" at a Democratic Party fundraiser and early birthday celebration for Kennedy in May 1962. However, both contemporaneous accounts and historical research have not conclusively proven a long-term affair, and much of the speculation is based on anecdotal evidence and hearsay.

```
messages = [HumanMessage(content="Add 3 and 4. Multiply the output by 2. Divide the output by 5")]
messages = react_graph.invoke({"messages": messages})
```

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

===== Human Message

=====

Add 3 and 4. Multiply the output by 2. Divide the output by 5

===== Ai Message

=====

Tool Calls:

add (call\_UR7Cp0pbVpRyof8Dyq4kZUHm)

Call ID: call\_UR7Cp0pbVpRyof8Dyq4kZUHm

Args:

a: 3

b: 4

===== Tool Message

=====

Name: add

7

===== Ai Message

=====

Tool Calls:

multiply (call\_OiwYtGUW13AY8CB3qY3Y3Vlt)

Call ID: call\_OiwYtGUW13AY8CB3qY3Y3Vlt

Args:

a: 7

b: 2

===== Tool Message

=====

Name: multiply

14

...

2.8

===== Ai Message

=====

The final result after adding 3 and 4, multiplying the result by 2, and then dividing by 5 is 2.8.

---

Phiên bản #2

Được tạo 19 tháng 4 2025 10:03:28 bởi Đỗ Ngọc Tú

Được cập nhật 22 tháng 4 2025 02:16:11 bởi Đỗ Ngọc Tú