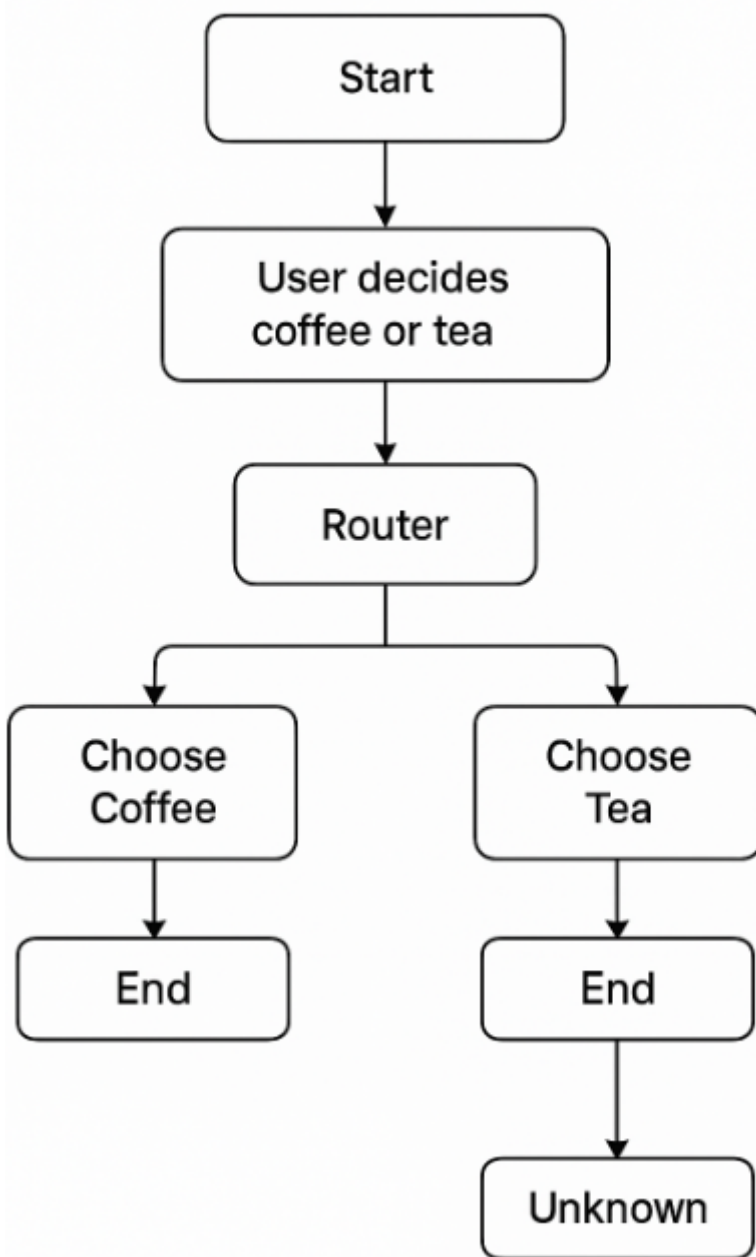


Bài Thực Hành Router Trong LangGraph

“ Mục tiêu: Tạo một LangGraph gồm các bước đơn giản, giúp người dùng quyết định sẽ uống Coffee hay Trà.



I. Cài đặt

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

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
```

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"]
```

II. Kết nối LangChain

```
#!pip install langgraph langchain openai
```

III. Định nghĩa **State Schema** (State = Memory)

```
from typing import TypedDict, Literal

class DrinkState(TypedDict):
    preference: Literal["coffee", "tea", "unknown"]
```

TypedDict

- Đây là một **tính năng của Python (từ `typing` module)** cho phép bạn khai báo **dictionary có cấu trúc rõ ràng**, giống như một `class`.
- Mục đích là để **giúp IDE và trình kiểm tra kiểu như MyPy bắt lỗi sớm** nếu bạn dùng sai key hoặc giá trị.

DrinkState

- Là tên của lớp, đại diện cho **trạng thái (state)** của ứng dụng LangGraph của bạn.

`preference: Literal["coffee", "tea", "unknown"]`

- Trường `preference` chỉ chấp nhận **một trong ba giá trị cụ thể**: `"coffee"`, `"tea"` hoặc `"unknown"`.
- Đây là cách giới hạn giá trị hợp lệ, giúp bạn tránh lỗi như `"cofee"` hay `"t"` do gõ sai.

IV. Định nghĩa các `Nodes` (mỗi Node là một Step)

```
# Node: Bắt đầu
def start_node(state: DrinkState) -> DrinkState:
    print("☺ Bạn thích uống gì hôm nay?")
    choice = input("Nhập 'coffee' hoặc 'tea': ").strip().lower()
    if choice not in ["coffee", "tea"]:
        choice = "unknown"
    return {"preference": choice}

# Node: Nếu chọn coffee
def coffee_node(state: DrinkState) -> DrinkState:
    print("☺ Bạn đã chọn Coffee! Tuyệt vời!")
    return state

# Node: Nếu chọn tea
def tea_node(state: DrinkState) -> DrinkState:
    print("☺ Bạn đã chọn Trà! Thanh tao ghê!")
    return state

# Node: Nếu nhập sai
def fallback_node(state: DrinkState) -> DrinkState:
    print("Không rõ bạn chọn gì. Vui lòng thử lại.")
    return state
```

V. Định nghĩa Router

```
def drink_router(state: DrinkState) -> str:
    if state["preference"] == "coffee":
        return "coffee_node"
    elif state["preference"] == "tea":
        return "tea_node"
    else:
        return "fallback_node"
```

VI. Xây dựng Graph

```
from langgraph.graph import StateGraph

builder = StateGraph(DrinkState)

# Thêm các nodes
builder.add_node("start", start_node)
builder.add_node("coffee_node", coffee_node)
builder.add_node("tea_node", tea_node)
builder.add_node("fallback_node", fallback_node)
builder.add_node("router", drink_router)

# Tạo flow giữa các bước
builder.set_entry_point("start")
builder.add_edge("start", "router")

builder.add_conditional_edges("router", {
    "coffee_node": "coffee_node",
    "tea_node": "tea_node",
    "fallback_node": "fallback_node"
})

# Gắn điểm kết thúc cho mỗi nhánh
builder.set_finish_point("coffee_node")
builder.set_finish_point("tea_node")
builder.set_finish_point("fallback_node")

# Compile graph
app = builder.compile()
```

VII. Chạy thử

```
# Trạng thái khởi đầu rỗng
initial_state: DrinkState = {"preference": "unknown"}
final_state = app.invoke(initial_state)
```

Kết quả

Tùy người dùng nhập vào `coffee` hay `tea`, graph sẽ tự động điều hướng tới đúng bước xử lý, và in kết quả tương ứng.

Tác giả: Đỗ Ngọc Tú
Công Ty Phần Mềm VHTSoft

Phiên bản #3
Được tạo 18 tháng 4 2025 02:17:27 bởi Đỗ Ngọc Tú
Được cập nhật 21 tháng 4 2025 05:50:54 bởi Đỗ Ngọc Tú