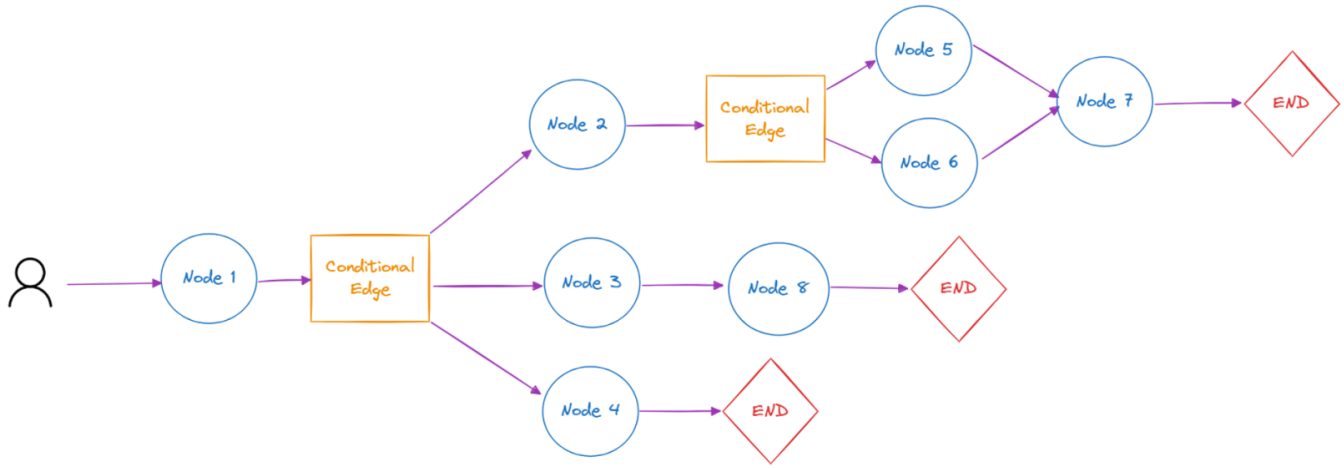
Giới thiệu về LangGraph

LangGraph Tạo LLM Workflow dễ dàng như vẽ đồ thị

Trong thời đại của trí tuệ nhân tạo và mô hình ngôn ngữ lớn (LLM), việc xây dựng các workflow phức tạp để xử lý ngôn ngữ tự nhiên đang trở nên ngày càng quan trọng. Tuy nhiên, quản lý các chuỗi tác vụ, rẽ nhánh logic, và vòng lặp trong quá trình tương tác với LLM có thể nhanh chóng trở nên phức tạp. Đây chính là lý do **LangGraph** ra đời.



LangGraph là gì?

LangGraph là một thư viện mã nguồn mở được phát triển dựa trên **LangChain**, cho phép bạn xây dựng các **đồ thị trạng thái (stateful graphs)** cho các ứng dụng sử dụng LLM. Thay vì chỉ chạy một chuỗi cố định các bước, LangGraph cho phép bạn xây dựng các quy trình linh hoạt hơn, bao gồm:

- Rẽ nhánh điều kiện (conditional branching)
- Vòng lặp (looping)
- Giao tiếp hai chiều giữa người dùng và LLM
- Quản lý trạng thái qua nhiều bước xử lý

LangGraph mang lại cách tiếp cận **kết hợp giữa Lập trình khai báo(**declarative**) và Lập trình mệnh lệnh(**imperative**)**, giúp bạn dễ dàng hình dung và kiểm soát luồng dữ liệu và trạng thái trong quá trình xử lý.

Vì sao nên sử dụng LangGraph?

Dễ hiểu, dễ hình dung

Bạn xây dựng workflow như một **đồ thị gồm các nút (nodes)**, mỗi nút thực hiện một tác vụ cụ thể, như gọi API, xử lý dữ liệu, tương tác với LLM, hay rẽ nhánh theo điều kiện.

Tái sử dụng và mở rộng tốt

Mỗi nút trong LangGraph là một function, nên bạn dễ dàng viết lại, mở rộng hoặc chia sẻ logic giữa các ứng dụng khác nhau.

Tích hợp chặt chẽ với LangChain

LangGraph kế thừa sức mạnh từ LangChain nên bạn có thể dễ dàng kết hợp với các **PromptTemplate**, **Agents**, **Memory**, hay **Tools** trong hệ sinh thái LangChain.

Cách hoạt động của LangGraph

Cấu trúc cơ bản của một LangGraph bao gồm:

- **State**: Biến lưu trữ dữ liệu cần thiết trong quá trình chạy đồ thị.
- **Node**: Hàm xử lý dữ liệu (có thể là gọi LLM, tính toán, rẽ nhánh...).
- **Edges**: Kết nối các node theo logic định sẵn.

Ví dụ đơn giản về đồ thị gồm 3 node:

```
from langgraph.graph import StateGraph, END

# Định nghĩa trạng thái
class MyState(TypedDict):
    input: str
    output: str

# Định nghĩa node
def process_input(state):
    result = state["input"].upper()
    return {"output": result}

# Tạo đồ thị
```

```
builder = StateGraph(MyState)
builder.add_node("process", process_input)
builder.set_entry_point("process")
builder.set_finish_point("process", END)
graph = builder.compile()

# Chạy đồ thị
result = graph.invoke({"input": "hello"})
print(result)
```

Ứng dụng thực tế

LangGraph rất phù hợp cho các bài toán phức tạp với LLM như:

- Chatbot nhiều bước với trạng thái duy trì qua từng lượt chat
- Tóm tắt tài liệu theo yêu cầu với logic tùy chỉnh
- Agent phức tạp với hành vi có điều kiện
- Hệ thống tư vấn, hỏi đáp với nhiều nhánh logic

Tác giả: Đỗ Ngọc Tú

Công Ty Phần Mềm VHTSoft

Phiên bản #1
Được tạo 14 tháng 4 2025 02:06:29 bởi Đỗ Ngọc Tú
Được cập nhật 14 tháng 4 2025 03:38:52 bởi Đỗ Ngọc Tú