

Vì sao LangGraph ra đời và Graph có ý nghĩa gì

Khi các ứng dụng xây dựng trên mô hình ngôn ngữ lớn (LLM) ngày càng trở nên phức tạp – từ chatbot đơn giản đến hệ thống tự động hóa tác vụ – thì nhu cầu về **cách điều phối luồng suy nghĩ, trạng thái, và hành động của LLMs** cũng tăng theo.

Đó là lý do vì sao **LangGraph** ra đời.

LangGraph là một **framework kết hợp LLMs với cấu trúc đồ thị trạng thái (stateful graphs)**, giúp tạo ra các ứng dụng có khả năng:

- Gọi nhiều công cụ khác nhau
- Lặp lại hành động nếu cần thiết
- Chuyển hướng luồng xử lý dựa trên kết quả
- Giữ được trạng thái trong suốt quá trình suy luận

I. Tại sao lại là LangGraph?

1. Giới hạn của mô hình agent truyền thống (LangChain Agents)

Các agents trong LangChain rất linh hoạt, nhưng đôi khi:

- Thiếu khả năng kiểm soát luồng suy luận
- Khó kiểm tra và debug khi có nhiều bước
- Thiếu khả năng giữ trạng thái rõ ràng (statefulness)
- Gặp khó khăn với các vòng lặp (loop), nhánh (branch), hoặc điều kiện lặp lại

Kết quả là khi xây dựng các ứng dụng phức tạp như AutoGPT, trợ lý đa tác vụ, hệ thống hỏi đáp nhiều bước... developer phải viết rất nhiều code xử lý luồng hoặc hack workaround.

2. LangGraph: Giải pháp điều phối LLM theo kiểu đồ thị trạng thái

LangGraph đưa ra một kiến trúc mới, nơi **quá trình suy nghĩ và hành động của LLM được mô hình hóa như một đồ thị (graph)**.

Mỗi node trong đồ thị:

- Có thể là một hành động cụ thể (gọi API, suy nghĩ, hỏi người dùng...)
- Có thể gọi LLM để đưa ra quyết định chuyển hướng (routing)

Lợi ích chính:

Tính năng	Ý nghĩa
Có trạng thái	Lưu giữ thông tin giữa các bước
Có vòng lặp	Hỗ trợ dễ dàng quá trình phản xạ (reflection), retry, sửa lỗi
Modular	Mỗi node độc lập, dễ tái sử dụng
Dễ debug	Có thể quan sát từng node và hướng đi trong quá trình xử lý
Tùy chọn luồng	Có thể dùng LLM để quyết định đi hướng nào trong graph

II. Tại sao lại dùng từ “Graph”?

1. Graph = Đồ thị trạng thái có hướng

LangGraph sử dụng **lý thuyết đồ thị (directed graph)** như một cách để **mô hình hóa quy trình tư duy** của một agent.

Trong graph:

- **Node** là một bước trong quy trình (ví dụ: "Phân tích yêu cầu", "Tìm kiếm trên web", "Tóm tắt dữ liệu", "Phản hồi")
- **Edge** là hướng đi từ một bước đến bước kế tiếp, có thể **phụ thuộc vào điều kiện** (ví dụ: nếu LLM nói "Chưa đủ dữ liệu", hãy quay lại bước tìm kiếm)

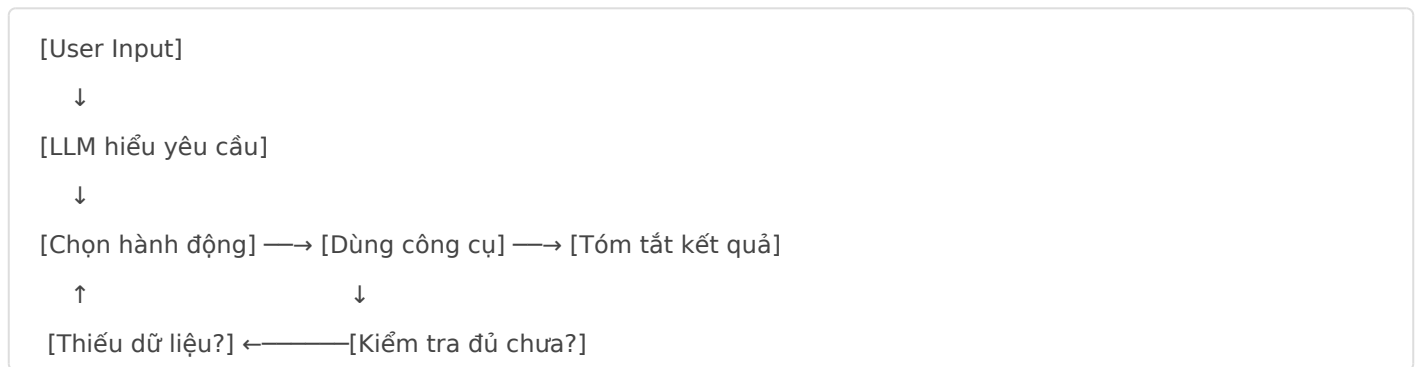
Graph này **giống như flowchart thông minh**, nhưng được điều khiển bởi LLM.

2. Graph giúp dễ hình dung quy trình phức tạp

Dưới dạng graph, developer có thể:

- Nhìn thấy toàn bộ hành trình của agent
- Điều chỉnh từng bước mà không ảnh hưởng đến toàn hệ thống
- Thêm logic phức tạp (loop, điều kiện, xử lý lỗi) một cách trực quan

Ví dụ:



III. Khi nào nên dùng LangGraph?

LangGraph **cực kỳ phù hợp** khi bạn xây dựng:

Multi-step reasoning (suy luận nhiều bước)

Agents có khả năng reflection / retry / repair

Workflow có trạng thái (stateful flows)

Hệ thống cần kiểm soát logic chặt chẽ

Quy trình có loop, branching hoặc fallback logic

Ví dụ ứng dụng:

- Trợ lý hỏi đáp tài liệu nhiều bước (hay bị "không đủ dữ liệu")
- Tác nhân viết báo cáo (chia nhỏ tác vụ và lặp lại từng phần)
- QA system với logic xác minh độ chính xác
- Tác nhân phân tích dữ liệu và tư vấn lỗi truy vấn

Câu hỏi	Trả lời
Vì sao LangGraph?	Vì các ứng dụng LLM ngày càng phức tạp, cần khả năng điều phối, giữ trạng thái, và xử lý logic phức tạp.
Vì sao "Graph"?	Vì LangGraph mô hình hóa tư duy của agent dưới dạng đồ thị trạng thái có hướng — giúp kiểm soát, mở rộng, debug dễ dàng hơn.

Phiên bản #1

Được tạo 14 tháng 4 2025 03:31:32 bởi Đỗ Ngọc Tú

Được cập nhật 14 tháng 4 2025 03:38:52 bởi Đỗ Ngọc Tú