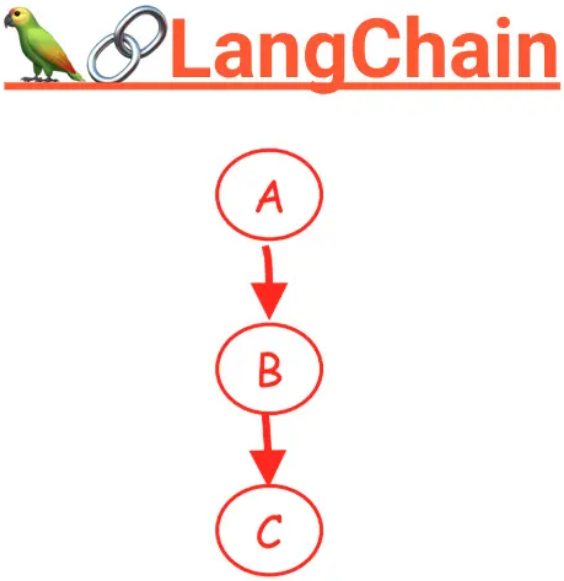
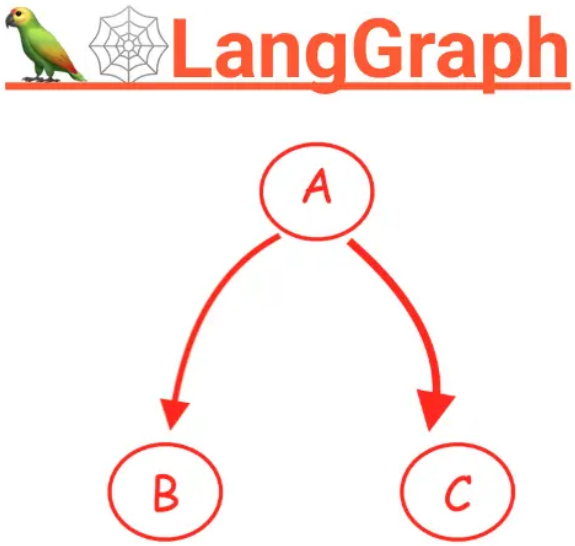
So sánh LangGraph và LangChain





Với sự phát triển nhanh chóng của các mô hình ngôn ngữ lớn như GPT, Claude, Mistral..., các thư viện như **LangChain** và **LangGraph** ra đời để giúp lập trình viên dễ dàng xây dựng ứng dụng sử dụng LLM. Nhưng giữa hai cái tên quen thuộc này, bạn nên chọn cái nào? Hoặc khi nào nên dùng cả hai?

Bài viết này sẽ giúp bạn hiểu rõ **sự khác biệt, điểm mạnh, và cách sử dụng** của **LangChain** và **LangGraph**.

I. Tổng quan

Đặc điểm	LangChain	LangGraph
Mục tiêu	Tạo **pipeline** (chuỗi) LLM logic	Tạo **workflow** (đồ thị trạng thái) phức tạp
Cấu trúc	Tuyến tính (linear chain / agent)	Đồ thị trạng thái có vòng lặp, rẽ nhánh
Mô hình dữ liệu	Stateless (không lưu trạng thái)	Stateful (quản lý trạng thái qua nhiều bước)
Tích hợp LLM	Rất mạnh, nhiều tools tích hợp	Kế thừa từ LangChain
Độ linh hoạt	Cao với chuỗi đơn giản hoặc agents	Rất cao với workflow phức tạp

Đặc điểm	LangChain	LangGraph
Sử dụng chính	Chatbot, Q&A, Retrieval, Agents	Bot đa bước, phân nhánh logic, memory flows

II. Kiến trúc hoạt động

LangChain

LangChain cung cấp một tập hợp các **abstractions** như:

- `LLMChain`: Gọi một LLM với prompt
- `Chain`: Kết hợp nhiều bước xử lý nối tiếp nhau
- `Agent`: Cho phép mô hình quyết định hành động tiếp theo
- `Memory`: Lưu trạng thái cuộc hội thoại tạm thời

Mỗi chain là **tuyến tính** — dữ liệu đi từ đầu đến cuối qua từng bước.

Ưu điểm: Dễ dùng, dễ hiểu, setup nhanh.
Nhược điểm: Hạn chế khi bạn muốn rẽ nhánh, lặp lại, hoặc xử lý logic phức tạp.

LangGraph

LangGraph đưa kiến trúc lên tầm cao hơn: bạn không chỉ nối các bước, bạn **vẽ ra một đồ thị trạng thái**. Trong đó:

- Mỗi node là một bước xử lý
- Bạn có thể rẽ nhánh tùy vào điều kiện
- Có thể lặp lại một node hoặc quay lại node trước
- **Trạng thái được lưu và truyền qua toàn bộ đồ thị**

LangGraph kế thừa toàn bộ các công cụ từ LangChain như `LLMChain`, `PromptTemplate`, `Tool`, `Memory`, `Agent`,... nhưng nâng cấp khả năng tổ chức luồng xử lý lên **mức workflow engine**.

Ưu điểm: Xử lý logic phức tạp, hỗ trợ đa chiều, có vòng lặp.
Nhược điểm: Khó hơn để bắt đầu, yêu cầu hiểu về state và luồng.

III. Khi nào dùng LangChain?

Dùng **LangChain** khi bạn:

- Chỉ cần chạy một chuỗi các bước (pipeline)
- Viết một chatbot đơn giản
- Làm Q&A với Retriever
- Gọi LLM để xử lý văn bản đầu vào và trả lại đầu ra ngay lập tức

- Không cần quản lý nhiều trạng thái phức tạp

Ví dụ: Lấy nội dung từ một file, tóm tắt nó bằng GPT, rồi gửi email.

IV. Khi nào dùng LangGraph?

Dùng **LangGraph** khi bạn:

- Cần xây dựng các **hệ thống xử lý nhiều bước có rẽ nhánh**
- Muốn xây chatbot đa luồng hội thoại (conversation flows)
- Có logic phức tạp: vòng lặp, kiểm tra điều kiện, lùi lại bước trước
- Quản lý trạng thái giữa nhiều bước, hoặc muốn lưu vào database
- Xây dựng agent với logic phức tạp hơn `ReAct`

Ví dụ:
Hệ thống kiểm tra đơn hàng:

1. Người dùng gửi mã đơn →
2. Hệ thống kiểm tra trạng thái đơn →
3. Nếu đơn đang giao: gửi thông báo;
4. Nếu đơn bị huỷ: hỏi lý do → ghi log → gửi xác nhận.

Bạn khó làm điều này gọn gàng với LangChain, nhưng **LangGraph làm rất tốt**.

V. Có thể dùng kết hợp không?

Hoàn toàn có thể!
LangGraph thực tế **được xây dựng trên nền LangChain**, nên bạn có thể dùng `LLMChain`, `RetrievalChain`, `Agents`, `PromptTemplate` bên trong các node của LangGraph.

Ví dụ: Một node trong đồ thị LangGraph có thể gọi một LangChain Agent để xử lý một phần công việc.

VI. So sánh ngắn gọn qua ví dụ

LangChain (linear):

```
chain = SimpleSequentialChain(chains=[chain1, chain2])
result = chain.run("Hello world")
```

LangGraph (graph):

```
builder = StateGraph()
builder.add_node("step1", step1)
builder.add_node("step2", step2)
builder.add_edge("step1", "step2")
builder.set_entry_point("step1")
graph = builder.compile()
graph.invoke({"input": "Hello world"})
```

VII. Kết luận

Nếu bạn cần...	Hãy chọn...
Xây dựng nhanh một chuỗi xử lý	◫ LangChain
Quản lý nhiều trạng thái phức tạp	◫ LangGraph
Làm chatbot đơn giản	◫ LangChain
Làm agent với hành vi linh hoạt	◫ LangGraph
Kết hợp các công cụ và logic rẽ nhánh	◫ LangGraph

Tác giả: Đỗ Ngọc Tú

Công Ty Phần Mềm VHTSoft

Phiên bản #2
Được tạo 14 tháng 4 2025 02:30:10 bởi Đỗ Ngọc Tú
Được cập nhật 14 tháng 4 2025 02:40:55 bởi Đỗ Ngọc Tú