Core Concepts for AI Development

Dưới đây là phần giới thiệu tổng quan về **Core Concepts for AI Development** – những khái niệm nền tảng bạn cần hiểu để phát triển hệ thống AI hiện đại, đặc biệt là các ứng dụng sử dụng LLM (Large Language Models):

1. Foundation Models (FM)

Là **mô hình học sâu được huấn luyện trên lượng dữ liệu cực lớn**, dùng được cho nhiều tác vụ khác nhau:

- Ví dụ: GPT (OpenAI), Claude (Anthropic), Gemini (Google)
- Đặc điểm: tự học từ dữ liệu để hiểu ngôn ngữ, lập luận, dịch thuật, viết mã...

2. Prompt Engineering

Là nghệ thuật **thiết kế câu lệnh đầu vào (prompt)** để dẫn dắt LLM trả lời theo ý bạn.

- Prompt tốt = kết quả tốt
- Gồm: Zero-shot, Few-shot, Chain-of-Thought...

□□ *Ví dụ:*

```
Bạn là chuyên gia tài chính. Hãy phân tích bảng sau và tóm tắt kết quả.
```

3. Tool Use / Function Calling

LLM có thể được **kết nối với các công cụ bên ngoài** như:

- Truy xuất cơ sở dữ liệu
- Gọi API
- Đọc file PDF, Excel...

□□ Trong OpenAI, ta dùng `functions=` hoặc `tools=` để định nghĩa các công cụ cho agent.

4. Memory / Context Management

LLM không có "trí nhớ dài hạn" tự nhiên. Để xây dựng hội thoại dài, ta cần:

- Quản lý lịch sử hội thoại
- Gọi vector store (semantic search)
- Tạo context window hiệu quả

□□ Framework hỗ trợ: LangChain, Semantic Kernel...

5. Agentic Reasoning

Là khả năng để LLM **tự đưa ra kế hoạch, chọn công cụ, thực thi nhiều bước** – như một "agent thông minh".

Ví dụ:

- Đọc hóa đơn → trích thông tin → tính tổng → gửi email

□□ Hỗ trợ bởi:

- OpenAI Agent SDK
- LangGraph (LangChain)
- CrewAI, AutoGPT, etc.

6. Retrieval-Augmented Generation (RAG)

Kết hợp **kiến thức bên ngoài** (từ database, PDF, Notion...) vào LLM để trả lời chính xác hơn.

▶ Quy trình:

1. Index tài liệu thành vector
2. Nhận truy vấn → tìm đoạn liên quan
3. Ghép vào prompt → trả về kết quả

7. Evaluation & Tracing

Cần kiểm tra chất lượng kết quả AI:

- Dùng trace (OpenAI) để xem agent làm gì
- Tự viết test case cho prompt
- So sánh kết quả với ground truth

8. Workflow Design Patterns

Kết hợp các pattern như:

- Prompt Chaining
- Routing
- Parallelization
- Orchestrator-Worker

- Evaluator-Optimizer

Tùy bài toán mà chọn pattern phù hợp (đọc tài liệu, hỏi-đáp, xử lý file...)

Phiên bản #1
Được tạo 29 tháng 4 2025 16:36:58 bởi Đỗ Ngọc Tú
Được cập nhật 2 tháng 5 2025 07:45:21 bởi Đỗ Ngọc Tú