Công ty cổ phần VCCorp

BÁO CÁO TUẦN 2

Tìm hiểu Prompting Engineering for Developer

Tác giả: Ngô Minh Đức

Người hướng dẫn: Anh Ngô Văn Vĩ

Hà Hội, 6/2025

Tóm tắt nội dung

Báo cáo này trình bày những nội dung quan trọng về khái niệm, đặc điểm, cấu trúc, các phương pháp và ứng dụng của Prompting Engineer đối với Developer.

MỤC LỤC

Chương 1: Giới thiệu chung 4
Chương 2: Khái niệm 5
Chương 3: Đặc điểm 5
Chương 4: Cấu trúc cơ bản 6
4.1 Instruction 6
4.2 Role 7
4.3 Context 7
4.4 Output Format 7
Chương 5: Các phương pháp Prompting Engineer phổ biến 8
5.1 Zero-shot Prompting 8
5.2 One-shot và Few-shot Prompting 8
5.3 System, Role, Contextual Prompting 8
5.4 Step-back Prompting 9
5.5 Chain of Thought (CoT) 9
5.6 Self-consistency Prompting 9
5.7 Tree of Thoughts (ToT) 9
5.8 ReAct (Reason + Act) 9
5.9 Automatic Prompt Engineering (APE) 9
Chương 6: Ứng dụng 10
Chương 7: Kết luận 10

Chương 1: Giới thiệu chung

Trong kỷ nguyên AI hiện đại, việc giao tiếp hiệu quả với mô hình ngôn ngữ lớn (LLM) đã trở thành một kỹ năng cần thiết đối với developer. Prompt Engineering, hay kỹ thuật thiết kế prompt, cho phép developer tối ưu hóa đầu vào để nhận được đầu ra mong muốn từ AI.

Việc nắm vững Prompt Engineering không chỉ giúp tiết kiệm thời gian và chi phí mà còn góp phần tăng hiệu suất phát triển ứng dụng và khai thác tối đa sức mạnh của AI.

Chương 2: Khái niệm

Prompt Engineering là quá trình thiết kế, tối ưu hóa, và kiểm tra đầu vào văn bản (prompt) gửi cho mô hình ngôn ngữ như ChatGPT, Claude, Gemini,... để mô hình AI có thể hiểu, diễn giải, nhằm đạt được đầu ra đúng mong đợi.

Promt có thể là một từ, một cụm từ hoặc cả một đoạn văn bản.

Prompt input (đầu vào) hoặc query (truy vấn) mà người dùng cung cấp để gợi ra các phản hồi cụ thể từ một mô hình AI.

Trong bối cảnh lập trình, Prompt Engineering giúp Developer:

- Tạo nhanh giao diện mã.
- Sinh ra test case.
- Tối ưu hóa query với ngôn ngữ SQL, NoSQL.
- Sinh tài liệu kỹ thuật (API docs, comment mã, README, ...).

Chương 3: Đặc điểm

- Với mỗi lượt, trực tiếp chỉ cho model cách nó nên phản hồi . Prompt có thể chứa hướng dẫn rất cụ thể, yêu cầu mô hình trả lời bằng ngôn ngữ, phong cách, hoặc định dạng cụ thể**.** Ví dụ: "Hãy trả lời tôi như một chuyên gia bảo mật hệ thống và trình bày theo bullet point."
- Nhanh và dễ sử dụng. Không cần huấn luyện mô hình hay setup phức tạp. Chỉ cần văn bản và mô hình là có thể tương tác. Ví dụ**:** Gửi đoạn văn bản code hoặc yêu cầu bằng tiếng Việt để ChatGPT sinh mã.
- Chỉ cần một lượng rất nhỏ ví dụ . Với few-shot learning, chỉ cần 1–3 ví dụ để mô hình hiểu được mẫu phản hồi mong muốn. Ví dụ**:** "Đây là một ví dụ tóm tắt bài báo, hãy làm tương tự với bài sau."
- Cho phép kết hợp kiến thức miền (domain knowledge) và thông tin theo ngữ cảnh để cải thiện độ chính xác và mức độ phù hợp. Có thể truyền vào prompt các thông tin chuyên ngành hoặc tình huống cụ thể để mô hình trả lời sát với thực tế hơn. Ví dụ: Khi viết prompt sinh mã y tế, có thể thêm vào "Giả sử bạn là

lập trình viên trong ngành chăm sóc sức khỏe, xử lý hồ sơ bệnh án điện tử (EHR)".

- Cho phép kiểm soát và tùy chỉnh tốt hơn các phản hồi của mô hình. Prompt giúp bạn định hình phong cách, độ dài, độ chi tiết và thậm chí là quan điểm trả lời. Tuy nhiên, nếu yêu cầu đặc thù và cần độ chính xác cao (ví dụ: dịch vụ y tế, luật pháp), fine-tuning có thể hiệu quả hơn.
- Sẵn có các model đã được đào tạo để tuân theo chỉ dẫn một cách rất hiệu quả. Các mô hình như ChatGPT (gpt-3.5-turbo, gpt-4), Claude 3 hay Gemini Pro đều đã được fine-tune theo kiểu instruction-following. Ứng dụng**:** Dễ dàng tích hợp các model này vào backend ứng dụng hoặc hệ thống hỗ trợ khách hàng qua API.
- Mất chi phí tính theo số lượng token của prompt và response. Chi phí sử dụng phụ thuộc vào độ dài prompt + phản hồi. Prompt càng chi tiết và dài thì tốn chi phí hơn. Ví dụ: Sử dụng gpt-3.5-turbo với prompt ~700 từ tiếng Anh + response ngắn có thể chỉ tốn $0.002/lượt (tương đương 50 VNĐ). Trong khi một prompt phức tạp nhiều bước có thể tốn 10–20 lần chi phí

Chương 4: Cấu trúc cơ bản

4.1 Instruction

AI hiện đại cũng có thể làm theo các chỉ dẫn phức tạp hơn nhiều, nhất là những mô hình được train với Reinforcement Learning from Human Feedback (RLHF) như ChatGPT.

- Instruction càng rõ ràng, cụ thể, dễ hiểu càng tốt.
- Lặp lại nhiều lần những yêu cầu, chỉ dẫn quan trọng
- Dùng lời nhắc mang tính khẳng định thay vì phủ định
- Dùng markup language (các dấu ngoặc, dấu quote, gạch đầu dòng, vv.) để phân tách rõ ràng giữa các yêu cầu khác nhau, giữa các phần khác nhau của prompt
- Cung cấp chỉ dẫn một cách tuần tự, từng bước một. Có thể chia một prompt lớn thành các bước nhỏ, đơn giản hơn.

4.2 Role

Một kỹ thuật gợi ý khác là gán vai trò (role) cho AI. Ví dụ: lời nhắc của bạn có thể bắt đầu bằng "You are a doctor" hoặc "Act as a prominent lawyer" và sau đó yêu cầu AI trả lời một số câu hỏi về y tế hoặc pháp lý. Khi gán một role cho AI, chúng ta đã cung cấp cho nó một số bối cảnh giúp nó hiểu câu hỏi tốt hơn và ta cũng phần nào đó định hình được cách mà nó phản hồi (như phong cách viết, phạm vi thông tin mà nó cung cấp)

4.3 Context

Thông tin bổ sung, có thể là dữ liệu, background, hoặc định hướng suy nghĩ. Đây là phần mà ta sẽ cung cấp nhiều thông tin hơn cho mô hình, giúp nó hiểu về mục tiêu, lĩnh vực, logic, hoặc giả định trước khi trả lời. Ví dụ: "Bạn là chuyên gia DevOps tại công ty startup đang chuẩn bị phát hành sản phẩm ra thị trường. Hãy đánh giá script CI/CD dưới góc độ hiệu quả triển khai và bảo mật."

4.4 Output Format

Đây là phần giúp định hướng cho mô hình biết phải trả lời theo định dạng cụ thể nào để dễ tích hợp vào hệ thống hay phục vụ cho mục đích nhất định.

- JSON: Thường dùng trong frontend/backend để parsing dữ liệu tự động. Ví dụ: "Trích xuất tên, email và độ tuổi từ đoạn văn sau và trả kết quả theo định dạng JSON."
- Markdown: Dùng để hiển thị tài liệu trên GitHub, blog, Notion... Ví dụ: "Hãy viết tài liệu hướng dẫn sử dụng API này bằng Markdown.
- Plain Text: Thích hợp cho mô hình trả lời đơn giản, ngắn gọn, dễ copy. Ví dụ: "Giải thích ngắn gọn tại sao async/await lại hữu ích trong JavaScript. Trả lời bằng văn bản thuần túy.". "Async/await giúp viết code bất đồng bộ dễ đọc, giống như code đồng bộ, tránh callback hell."

Chương 5: Các phương pháp Prompting Engineer phổ biến

5.1 Zero-shot Prompting

- Khái niệm: Chỉ cung cấp yêu cầu, không có ví dụ nào.
- Ví dụ: "Phân loại bài đánh giá phim này là TÍCH CỰC, TRUNG LẬP hoặc TIÊU CỰC: 'Her là một tác phẩm tuyệt vời, nhưng cũng đầy lo lắng về tương lai của AI.'"
- Ưu điểm: Đơn giản, nhanh gọn.
- Nhược điểm: Với tác vụ phức tạp, độ chính xác có thể thấp.

5.2 One-shot và Few-shot Prompting

- Khái niệm: Cung cấp một hoặc vài ví dụ mẫu.
- Ví dụ: "Phân tích đơn hàng pizza thành JSON. EXAMPLE: 'Tôi muốn pizza nhỏ với phô mai, sốt cà.' JSON: {size: 'small', ingredients: ['cheese', 'tomato sauce']}"
- Ưu điểm: Tăng độ chính xác rõ rệt, đặc biệt với tác vụ định dạng đầu ra.
- Mẹo: Bao gồm cả edge case.

5.3 System, Role, Contextual Prompting

- System: Định nghĩa nhiệm vụ tổng thể.
- Role: Gán vai trò cụ thể cho mô hình.
- Contextual: Cung cấp ngữ cảnh chi tiết.
- Ví dụ kết hợp:
  - System: "Bạn là chuyên gia tư vấn tài chính."
  - Context: "Khách hàng đang tìm cách tiết kiệm khi về hưu."
  - User: "Đưa ra kế hoạch tiết kiệm trong 10 năm."

5.4 Step-back Prompting

- Khái niệm: Trả lời câu hỏi tổng quát trước, sau đó đi vào chi tiết.
- Ví dụ:
  - Bước 1: "Các bối cảnh phổ biến trong game bắn súng là gì?"
  - Bước 2: "Viết cốt truyện cho level ở căn cứ quân sự bỏ hoang."

5.5 Chain of Thought (CoT)

- Khái niệm: Yêu cầu mô hình tư duy từng bước.
- Ví dụ: "Khi tôi 3 tuổi, bạn tôi gấp 3 lần tuổi tôi. Bây giờ tôi 20 tuổi, bạn tôi bao nhiêu tuổi? Hãy suy nghĩ từng bước."
- Ưu điểm: Rất tốt cho bài toán logic, toán học, lập trình.

5.6 Self-consistency Prompting

- Khái niệm: Gửi nhiều lần cùng một prompt, chọn kết quả phổ biến nhất.
- Ứng dụng: Các bài toán suy luận cần độ chính xác cao.

5.7 Tree of Thoughts (ToT)

- Khái niệm: Mở rộng từ CoT, mô hình xem xét nhiều nhánh suy luận.
- Ứng dụng: Phù hợp với tác vụ sáng tạo hoặc phân tích đa chiều.

5.8 ReAct (Reason + Act)

- Khái niệm: Mô hình lý luận và thực thi hành động (ví dụ: tra cứu, gọi API).
- Ứng dụng: Hệ thống chatbot, truy xuất dữ liệu thời gian thực.

5.9 Automatic Prompt Engineering (APE)

- Khái niệm: Dùng AI để tạo hoặc tối ưu prompt.
- Ví dụ: "Viết 10 cách khác nhau để đặt mua áo Metallica size S."
- Ứng dụng: Chatbot đào tạo, hệ thống hỏi đáp đa dạng.

Chương 6: Ứng dụng

- Sinh test case tự động: Viết các test case bằng Jest cho hàm sau:

  ```
  function add(a, b) {
  return a + b;
  }
  ```

- Sinh REST API document: Từ đoạn code Express sau, hãy viết REST API documentation theo định dạng Markdown.
- Tối ưu truy vấn MongoDB: Tôi muốn truy vấn MongoDB để tìm tất cả user có age > 25 và sắp xếp theo name. Viết query MongoDB tối ưu và nhận xét.
- Sinh component React: Tạo một React component hiển thị danh sách sản phẩm theo dữ liệu JSON sau.
- Tự sinh README.md: Sinh README cho dự án Node.js có router Express, MongoDB, và JWT Auth.
- Tự sinh Swagger/OpenAPI: Chuyển router Express dưới đây sang tài liệu Swagger 3.0.
- Refactor code: Hãy refactor đoạn code sau để ngắn gọn và hiệu quả hơn.
- Sinh form React + validation: Tạo form React có trường email, password, và validate với React Hook Form.
- Sinh unit test bằng Vitest: Tạo unit test cho component Button sử dụng Vitest + Testing Library.
- Sinh API mock data: Sinh dữ liệu ngẫu nhiên cho 10 user với Faker.js hoặc API mock tool.

Chương 7: Kết luận

Prompt Engineering không chỉ là xu hướng mà sẽ trở thành kỹ năng bắt buộc trong kỷ nguyên AI-first. Việc nắm chắc kỹ thuật này giúp developer tiết kiệm thời gian, tăng hiệu suất, và đồng thời khai thác tối đa khả năng tự động hóa của AI.

Một số lưu ý khi sử dụng Prompt Engineer:

- Không nên phụ thuộc hoàn toàn vào AI: Kết quả do mô hình tạo ra có thể thiếu chính xác hoặc không cập nhật theo thời gian thực. Developer cần kiểm tra kỹ lưỡng trước khi triển khai vào sản phẩm thật.
- Đảm bảo bảo mật và không đưa dữ liệu nhạy cảm: Không nên gửi API key, thông tin người dùng, dữ liệu cá nhân hay thông tin nhạy cảm vào prompt. Luôn giả lập hoặc ẩn dữ liệu thật.
- Kiểm soát độ dài prompt hợp lý: Prompt quá dài không chỉ gây tốn chi phí mà còn dễ khiến mô hình "mất trọng tâm". Cố gắng viết ngắn gọn, rõ ràng, chia khối logic theo từng phần.
- Luôn kiểm thử với nhiều kiểu dữ liệu đầu vào: Với cùng một prompt, kết quả có thể khác nhau nếu input thay đổi. Nên thử với nhiều biến thể dữ liệu để đánh giá tính ổn định và độ phù hợp của prompt.
- Theo dõi và log kết quả phản hồi từ AI: Việc log kết quả giúp bạn dễ dàng debug, cải thiện prompt theo thời gian, cũng như phục vụ kiểm định chất lượng đầu ra.
- Cập nhật theo sự thay đổi của model: Các mô hình LLM thường xuyên được cập nhật. Prompt từng hoạt động tốt có thể không còn tối ưu nữa. Nên kiểm tra định kỳ và điều chỉnh theo bản cập nhật mới.
- Tận dụng công cụ hỗ trợ Prompt Engineering: Có thể dùng các công cụ như PromptLayer, LangChain, hoặc OpenAI Playground để ghi log, thử nghiệm và tối ưu prompt.