BÁO CÁO BÀI TẬP

Kỹ Năng Viết Prompt Hiệu Quả

Tận dụng tối đa khả năng của Mô hình Ngôn ngữ Lớn (LLM)

Sinh viên thực hiện:

[Lê Anh Quang]

MSSV: [25021951] | Lớp: [K70I-CS2]

I. Giới thiệu

Báo cáo này trình bày quá trình thực nghiệm và phân tích kỹ thuật prompt engineering — nghệ thuật viết lệnh hướng dẫn (prompt) để khai thác tối đa tiềm năng của các mô hình ngôn ngữ lớn (LLM) như ChatGPT, Claude hay Gemini trong bối cảnh học tập.

Ba tác vụ học tập phổ biến được chọn để thực nghiệm:

- Tóm tắt một tài liệu/bài đọc học thuật
- Giải thích một khái niệm phức tạp
- Tạo bộ câu hỏi ôn tập cho một chủ đề

Với mỗi tác vụ, ba phiên bản prompt được viết theo cấp độ tăng dần: cơ bản, cải tiến, và nâng cao (áp dụng các kỹ thuật role prompting, chain-of-thought, few-shot examples). Kết quả được thử nghiệm trên ChatGPT và so sánh có hệ thống.

II. Tác vụ 1 — Tóm tắt Tài liệu Học thuật

2.1. Bối cảnh

Chủ đề: Tóm tắt bài báo học thuật về thuật toán Transformer trong học sâu (deep learning).

Mục tiêu: LLM tạo ra bản tóm tắt ngắn gọn, chính xác, nêu rõ vấn đề — phương pháp — kết quả.

2.2. Ba phiên bản Prompt

Prompt Cơ bản

CƠ BẢN	`Tóm tắt bài báo này về Transformer.` `What is a Transformer Model? | IBM`

Kết quả: LLM tóm tắt chung chung, không có cấu trúc rõ ràng. Độ dài không được kiểm soát. Đôi khi bỏ sót thông tin kỹ thuật quan trọng.

Tóm tắt bài báo này về Transformer.
What is a Transformer Model? | IBM

Tóm tắt ngắn gọn bài **"What is a Transformer Model?" của IBM**:

- **Transformer**: mô hình AI xử lý chuỗi (NLP) dựa hoàn toàn vào **self-attention**, không dùng RNN/CNN.
- **Cốt lõi**: mỗi từ "nhìn" toàn bộ câu để hiểu ngữ cảnh → học quan hệ xa tốt hơn.
- **Thành phần chính**:
  - Embedding + Positional Encoding
  - Multi-head Self-Attention
  - Feedforward layers
- **Ưu điểm**:
  - Song song hóa tốt → train nhanh
  - Hiểu ngữ cảnh sâu → hiệu năng cao
- **Ứng dụng**: dịch máy, chatbot, tóm tắt văn bản, sinh nội dung
- **Mô hình nổi bật**: BERT, GPT

→ Ý chính: Transformer = nền tảng cho hầu hết AI ngôn ngữ hiện đại.

Prompt Cải tiến

CẢI TIẾN	`Tóm tắt bài báo học thuật sau trong 150-200 từ. Cấu trúc bản tóm tắt gồm 3 phần: 1. Vấn đề / Động lực nghiên cứu 2. Phương pháp đề xuất 3. Kết quả và đóng góp chính` `What is a Transformer Model? | IBM`

Kết quả: Đầu ra có cấu trúc, đúng độ dài. Nội dung bao quát đủ 3 chiều quan trọng của bài báo.

1. Vấn đề / Động lực nghiên cứu
Các mô hình xử lý ngôn ngữ trước đây như RNN và LSTM gặp hạn chế trong việc ghi nhớ quan hệ dài hạn và khó song song hóa, dẫn đến huấn luyện chậm và hiệu năng chưa tối ưu. Nhu cầu đặt ra là một kiến trúc có thể xử lý ngữ cảnh toàn cục hiệu quả hơn.

2. Phương pháp đề xuất
Transformer (giới thiệu trong Attention Is All You Need) loại bỏ hoàn toàn cơ chế tuần tự, thay bằng **self-attention** để mô hình hóa quan hệ giữa các từ trong toàn bộ chuỗi cùng lúc. Kiến trúc gồm embedding, positional encoding để giữ thông tin thứ tự, cùng với multi-head attention và feedforward layers nhằm học biểu diễn ngữ nghĩa sâu và đa chiều.

3. Kết quả và đóng góp chính
Transformer cho phép huấn luyện song song hiệu quả hơn, cải thiện đáng kể khả năng nắm bắt ngữ cảnh dài và đạt hiệu năng vượt trội trên nhiều tác vụ NLP. Kiến trúc này trở thành nền tảng cho các mô hình hiện đại như BERT và GPT, đóng vai trò then chốt trong sự phát triển của AI ngôn ngữ hiện nay.

Prompt Nâng cao

NÂNG CAO	`Bạn là một nhà nghiên cứu AI với 10 năm kinh nghiệm đọc và đánh giá bài báo khoa học. Nhiệm vụ: Tóm tắt bài báo dưới đây theo định dạng structured abstract chuẩn học thuật. Yêu`

	cầu đầu ra: - Background (1-2 câu): Bối cảnh và nghiên cứu - Objective (1 câu): Mục tiêu cụ thể của bài báo - Method (2-3 câu): Kiến trúc / thuật toán / dữ liệu sử dụng - Results (2-3 câu): Kết quả định lượng chính (nêu số liệu cụ thể) - Conclusion (1-2 câu): Ý nghĩa và giới hạn Sau đó đề xuất 3 câu hỏi mà một người đọc nên tự hỏi khi đánh giá bài báo này. What is a Transformer Model? | IBM

Kết quả: Đầu ra chuyên nghiệp, có chiều sâu phân tích. LLM áp dụng góc nhìn của chuyên gia, chủ động cung cấp số liệu và đặt câu hỏi phản biện.

Structured Abstract

Background

Các mô hình tuần tự như RNN/LSTM bị giới hạn trong việc học quan hệ dài hạn và khó song song hóa. Điều này tạo nhu cầu cho một kiến trúc hiệu quả hơn trong xử lý ngôn ngữ tự nhiên.

Objective

Trình bày và giải thích mô hình Transformer như một giải pháp thay thế hiệu quả cho các kiến trúc tuần tự truyền thống.

Method

Transformer, xuất phát từ Attention Is All You Need, sử dụng cơ chế self-attention để xử lý toàn bộ chuỗi đầu vào đồng thời. Kiến trúc gồm embedding, positional encoding để giữ thứ tự, và các khối lặp gồm multi-head attention kết hợp feedforward layers. Mô hình loại bỏ hoàn toàn recurrence, cho phép huấn luyện song song trên dữ liệu lớn.

Results

Bài viết dạng tổng quan nên không cung cấp số liệu thực nghiệm cụ thể. Tuy nhiên, Transformer được ghi nhận cải thiện đáng kể hiệu năng trên các tác vụ NLP như dịch máy và hiểu ngôn ngữ. Các mô hình dựa trên nó như BERT và GPT đạt kết quả state-of-the-art trên nhiều benchmark.

Conclusion

Transformer trở thành nền tảng chính của AI ngôn ngữ hiện đại nhờ khả năng mở rộng và hiệu quả. Tuy nhiên, nó đòi hỏi tài nguyên tính toán lớn và không phải là một nghiên cứu thực nghiệm độc lập trong bài viết này.

3 câu hỏi nên tự đặt ra khi đọc bài:

1. Bài viết có cung cấp đủ bằng chứng thực nghiệm hay chỉ mang tính tổng quan?
2. Self-attention thực sự cải thiện ở đâu so với RNN/LSTM trong từng tác vụ cụ thể?
3. Chi phí tính toán và dữ liệu lớn ảnh hưởng thế nào đến tính khả thi khi áp dụng thực tế?

2.3. Bảng so sánh kết quả

Tiêu chí	Cơ bản	Cải tiến	Nâng cao
Cấu trúc đầu ra	Không có	Có 3 phần rõ	Có 5 phần chuẩn
Kiểm soát độ dài	Không	Có (150-200 từ)	Có (theo section)
Độ chính xác KT	Trung bình	Tốt	Rất tốt
Tư duy phân tích	Không có	Hạn chế	Có (3 câu hỏi phản biện)
Tính ứng dụng	Thấp	Trung bình	Cao

III. Tác vụ 2 — Giải thích Khái niệm Phức tạp

3.1. Bối cảnh

Chủ đề: Giải thích khái niệm Gradient Descent trong machine learning cho người mới học.
Mục tiêu: LLM truyền đạt được cơ chế hoạt động, ý nghĩa trực quan, và ứng dụng của thuật toán.

3.2. Ba phiên bản Prompt

Prompt Cơ bản

CƠ BẢN	`Gradient Descent là gì?`

Kết quả: Định nghĩa kỹ thuật khô khan, thiếu ví dụ trực quan. Không phù hợp với người mới học.

Gradient Descent: thuật toán tối ưu dùng để **tìm giá trị nhỏ nhất của hàm mất mát (loss)**.

Ý tưởng cốt lõi

- Tính gradient (đạo hàm) của loss theo tham số
- Cập nhật tham số theo hướng **ngược gradient** (đi xuống dốc)

$$\theta = \theta - \eta \nabla J(\theta)$$

Trong đó

- θ: tham số
- η: learning rate
- $\nabla J(\theta)$: gradient của loss

Ý nghĩa

→ Lặp lại cập nhật → mô hình dần học được tham số tốt nhất.

Prompt Cải tiến

CẢI TIẾN	`Giải thích thuật toán Gradient Descent cho một sinh viên năm nhất chưa học machine learning. Bao gồm: - Định nghĩa bằng ngôn ngữ đơn giản - Một ví dụ thực tế dễ hình dung - Tại sao nó quan trọng trong ML Độ dài: khoảng 250 từ.`

Kết quả: Dễ hiểu hơn đáng kể. Ví dụ thực tế xuất hiện (thường là ví dụ leo núi). Vẫn thiếu tính tương tác.

Định nghĩa (đơn giản)

Gradient Descent là cách để máy tính **tự điều chỉnh tham số từng bước** nhằm giảm sai số. Hãy tưởng tượng bạn đang cố tìm điểm thấp nhất của một "bề mặt" (hàm loss). Ở mỗi bước, bạn nhìn độ dốc xung quanh và **đi theo hướng xuống dốc nhất**.

$$\theta = \theta - \eta \nabla J(\theta)$$

Ví dụ dễ hiểu

Hãy tưởng tượng bạn đứng trên một ngọn đồi trong đêm tối và muốn xuống thung lũng thấp nhất. Bạn không nhìn xa được, chỉ cảm nhận độ dốc dưới chân.

- Nếu đất nghiêng sang trái → bạn bước sang trái
- Nếu nghiêng mạnh → bước dài hơn
- Nếu gần bằng phẳng → bước nhỏ lại

Sau nhiều bước, bạn sẽ dần xuống gần điểm thấp nhất. Gradient Descent hoạt động tương tự: "độ nghiêng" chính là gradient, và "bước chân" là learning rate.

Tại sao quan trọng trong Machine Learning

Hầu hết mô hình ML (linear regression, neural network...) đều cần **tìm tham số tốt nhất** để dự đoán chính xác. Nhưng không thể thử hết mọi giá trị.

Gradient Descent giúp:

- Tìm nghiệm tốt **nhanh và hiệu quả**
- Là nền tảng cho việc huấn luyện mô hình lớn (deep learning)
- Hoạt động được với dữ liệu rất lớn

→ Nói ngắn gọn: nếu không có Gradient Descent, phần lớn mô hình ML hiện đại **không thể học được**.

Prompt Nâng cao

NÂNG CAO	`Bạn là giáo viên dạy toán và lập trình cho học sinh cấp 3. Phong cách giảng dạy của bạn: đơn giản, dùng phép so sánh hàng ngày, hỏi ngược lại để kích thích tư duy. Hãy giải thích Gradient Descent theo cấu trúc sau: BƯỚC 1 — BẮT ĐẦU BẰNG CÂU HỎI: Đặt 1 câu hỏi khiến học sinh tự nhận ra vấn đề mà GD giải quyết. BƯỚC 2 — PHÉP SO SÁNH TRỰC QUAN: Dùng đúng 1 phép so sánh với cuộc sống hàng ngày. Không dùng "leo núi". BƯỚC 3 — CƠ CHẾ HOẠT ĐỘNG: Giải thích learning rate, gradient, và bước cập nhật bằng ngôn ngữ phi kỹ thuật. BƯỚC 4 — KẾT NỐI THỰC TẾ: Cho một ví dụ cụ thể: GD đang làm gì khi ta train model nhận diện ảnh mèo? BƯỚC 5 — CÂU HỎI KIỂM TRA: Đặt 2 câu hỏi để học sinh tự kiểm tra xem đã hiểu chưa.`

Kết quả: Xuất sắc. LLM hoàn toàn nhập vai giáo viên, tạo ra phép so sánh sáng tạo (không phải leo núi), chuỗi giải thích có chiều sâu sư phạm, và câu hỏi kiểm tra chất lượng cao.

BƯỚC 1 — BẮT ĐẦU BẰNG CÂU HỎI
Nếu một mô hình dự đoán sai, làm sao nó biết phải **sửa bao nhiêu và sửa theo hướng nào** để đúng hơn?

BƯỚC 2 — PHÉP SO SÁNH TRỰC QUAN
Giống như bạn đang **nêm canh bị mặn**:
bạn không đổ cả chai nước vào ngay, mà **nếm → thêm chút nước → nếm lại → điều chỉnh tiếp** cho đến khi vừa ăn.

BƯỚC 3 — CƠ CHẾ HOẠT ĐỘNG

- **Gradient**: giống như "vị đang lệch theo hướng nào" (quá mặn hay quá nhạt, và lệch nhiều hay ít).
- **Learning rate**: lượng bạn điều chỉnh mỗi lần (đổ nhiều hay ít nước).
- **Cập nhật**: mỗi lần bạn **sửa một chút theo hướng làm giảm sai**.

$$\theta = \theta - \eta \nabla J(\theta)$$

Hiểu đơn giản:

- Sai nhiều → chỉnh mạnh hơn
- Sai ít → chỉnh nhẹ lại
- Lặp lại nhiều lần → tiến gần "vị chuẩn"

BƯỚC 4 — KẾT NỐI THỰC TẾ
Khi train model nhận diện ảnh mèo:

- Model nhìn ảnh → đoán "mèo" hoặc "không mèo"
- So với đáp án thật → tính sai bao nhiêu
- Gradient chỉ ra: "nên tăng/giảm trọng số nào" để đoán đúng hơn
- Gradient Descent **điều chỉnh các trọng số từng chút một**

Lặp hàng nghìn lần → model dần nhận ra đặc trưng như tai nhọn, mắt, lông...

BƯỚC 5 — CÂU HỎI KIỂM TRA

1. Nếu learning rate quá lớn, chuyện gì xảy ra khi "điều chỉnh"?
2. Nếu gradient gần bằng 0, điều đó nói gì về trạng thái của mô hình?

3.3. Bảng so sánh kết quả

Tiêu chí	Cơ bản	Cải tiến	Nâng cao

Phù hợp đối tượng	Không rõ	Có xác định	Rõ ràng, nhất quán
Tính trực quan	Thấp	Trung bình	Cao
Phép so sánh	Không có	Chung chung	Sáng tạo, cụ thể
Tính tương tác	Không có	Không có	Có (câu hỏi 2 chiều)
Độ sâu sư phạm	Thấp	Trung bình	Cao

IV. Tác vụ 3 — Tạo Bộ Câu hỏi Ôn tập

4.1. Bối cảnh

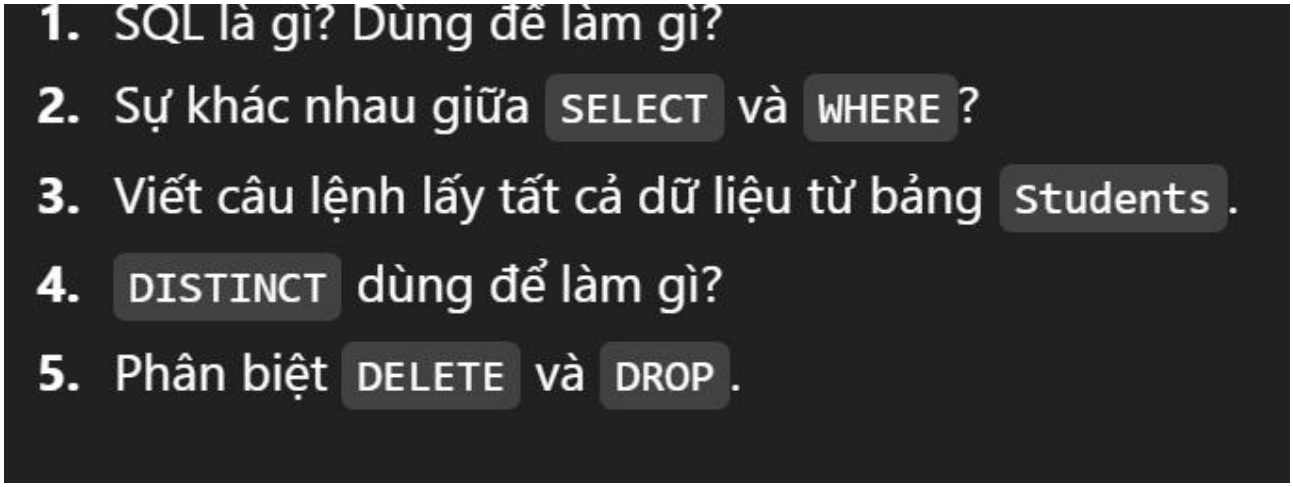
Chủ đề: Tạo bộ câu hỏi ôn tập môn Cơ sở Dữ liệu — phần SQL và thiết kế lược đồ quan hệ.
Mục tiêu: LLM tạo câu hỏi đa dạng cấp độ, có đáp án, đánh giá đúng mức độ tư duy cần thiết.

4.2. Ba phiên bản Prompt

Prompt Cơ bản

CƠ BẢN	`Tạo câu hỏi ôn tập về SQL.`

Kết quả: 5-7 câu hỏi lý thuyết cơ bản, lặp lại, không có đáp án, không phân cấp độ.



Prompt Cải tiến

CẢI TIẾN	`Tạo 10 câu hỏi ôn tập môn Cơ sở Dữ liệu về chủ đề: SQL và thiết kế lược đồ quan hệ. Yêu cầu: - 4 câu hỏi lý thuyết (nhận biết / hiểu) - 4 câu hỏi vận dụng (viết câu lệnh SQL) - 2 câu hỏi phân tích (thiết kế hoặc tối ưu) Mỗi câu hỏi kèm đáp án tham khảo.`

Kết quả: Tốt. Câu hỏi đa dạng, có đáp án. Nhưng câu SQL đôi khi thiếu context (không có schema).

I. Lý thuyết (4 câu)

Câu 1. SQL là gì?
Đáp án: Ngôn ngữ dùng để truy vấn và quản lý cơ sở dữ liệu quan hệ (CRUD, định nghĩa schema, phân quyền).

Câu 2. Phân biệt `PRIMARY KEY` và `FOREIGN KEY`.
Đáp án:

- `PRIMARY KEY`: định danh duy nhất mỗi bản ghi, không null
- `FOREIGN KEY`: tham chiếu đến khóa chính của bảng khác

Câu 3. Chuẩn hóa (Normalization) là gì?
Đáp án: Quá trình tổ chức dữ liệu để giảm dư thừa và tránh bất nhất (thường đến 3NF).

Câu 4. `INNER JOIN` khác `LEFT JOIN` thế nào?
Đáp án:

- `INNER JOIN`: chỉ lấy bản ghi khớp cả 2 bảng
- `LEFT JOIN`: lấy tất cả bên trái, kể cả không khớp

II. Vận dụng (4 câu)

Câu 5. Lấy tất cả sinh viên có tuổi > 20.
Đáp án:

```sql
SELECT * FROM Students
WHERE age > 20;
```

Câu 6. Lấy tên sinh viên và điểm từ 2 bảng `Students`, `Scores`.
Đáp án:

```sql
SELECT s.name, sc.score
FROM Students s
INNER JOIN Scores sc ON s.id = sc.student_id;
```

Câu 7. Đếm số sinh viên theo từng lớp.
Đáp án:

```sql
SELECT class, COUNT(*)
FROM Students
GROUP BY class;
```

Câu 8. Lấy 5 sinh viên điểm cao nhất.
Đáp án:

III. Phân tích (2 câu)

Câu 9. Một bảng `Orders` chứa: order_id, customer_name, customer_address. Có vấn đề gì?
Đáp án: Dữ liệu khách hàng bị lặp → nên tách thành 2 bảng:

- `Customers(id, name, address)`
- `Orders(order_id, customer_id)`

Câu 10. Query join 3 bảng chạy chậm, nguyên nhân và cách tối ưu?
Đáp án:

- Nguyên nhân: thiếu index, dữ liệu lớn, join không đúng cột
- Tối ưu:
  - Tạo index cho cột join
  - Giảm dữ liệu bằng `WHERE` trước
  - Tránh `SELECT *`
  - Xem execution plan để tối ưu logic

Nếu cần, mình có thể nâng cấp bộ này thành **đề thi hoàn chỉnh 45–60 phút**.

Prompt Nâng cao

NÂNG CAO	`Bạn là giảng viên đại học thiết kế đề cương ôn thi cuối kỳ môn Cơ sở Dữ liệu. Dưới đây là schema cơ sở dữ liệu quản lý thư viện: - SACH(MaSach, TenSach, NamXB, MaTacGia, MaTheLoai) - DOCGIA(MaDocGia, HoTen, NgaySinh, DiaChi) - MUON(MaMuon, MaDocGia, MaSach, NgayMuon, NgayTra) Hãy tạo bộ câu hỏi ôn tập theo thang Bloom: LEVEL 1 — NHỚ (2 câu): Định nghĩa khái niệm SQL cơ bản LEVEL 2 — HIỂU (2 câu): Giải thích kết quả của câu lệnh SQL cho trước LEVEL 3 — ÁP DỤNG (3 câu): Viết câu lệnh SQL trên schema trên LEVEL 4 — PHÂN TÍCH (2 câu): Phát hiện lỗi / tối ưu câu truy vấn LEVEL 5 — ĐÁNH GIÁ (1 câu): So sánh 2 cách thiết kế lược đồ và lập luận Mỗi câu kèm: đáp án tham khảo + giải thích ngắn tại sao đáp án đúng. Ghi rõ mức độ Bloom của từng câu.`

Kết quả: Xuất sắc. Câu hỏi gắn với schema cụ thể, bao phủ toàn bộ thang Bloom, đáp án có giải thích — sát với chất lượng đề thi thực tế.

Dưới đây là bộ câu hỏi ôn tập theo thang Bloom:

LEVEL 1 — NHỚ (2 câu)

Câu 1. `PRIMARY KEY` là gì?

Đáp án: Khóa chính dùng để định danh duy nhất mỗi bản ghi, không được trùng và không null.

Giải thích: Đảm bảo mỗi dòng là duy nhất → nền tảng của quan hệ.

Câu 2. `JOIN` dùng để làm gì?

Đáp án: Kết hợp dữ liệu từ nhiều bảng dựa trên điều kiện liên kết.

Giải thích: Vì dữ liệu được tách ra nhiều bảng, cần JOIN để truy vấn đầy đủ.

LEVEL 2 — HIỂU (2 câu)

Câu 3.

```sql
SELECT TenSach
FROM SACH
WHERE NamXB > 2020;
```

Hỏi: Kết quả là gì?

Đáp án: Danh sách tên sách xuất bản sau năm 2020.

Giải thích: `WHERE` lọc theo điều kiện năm.

Câu 4.

```sql
SELECT MaDocGia, COUNT(*)
FROM MUON
GROUP BY MaDocGia;
```

Hỏi: Kết quả là gì?
Đáp án: Số lần mượn của từng độc giả.
Giải thích: `GROUP BY` gom theo độc giả, `COUNT` đếm số dòng.

LEVEL 3 — ÁP DỤNG (3 câu)

Câu 5. Liệt kê tên sách và năm xuất bản.
Đáp án:

```sql
SELECT TenSach, NamXB
FROM SACH;
```

Giải thích: Truy vấn cột cơ bản từ bảng.

Câu 6. Liệt kê tên độc giả và ngày mượn sách.
Đáp án:

```sql
SELECT d.HoTen, m.NgayMuon
FROM DOCGIA d
JOIN MUON m ON d.MaDocGia = m.MaDocGia;
```

Giải thích: JOIN để kết nối thông tin người mượn và giao dịch.

Câu 7. Tìm các sách chưa từng được mượn.
Đáp án:

```sql
SELECT TenSach
FROM SACH s
LEFT JOIN MUON m ON s.MaSach = m.MaSach
WHERE m.MaSach IS NULL;
```

Giải thích: `LEFT JOIN` + `IS NULL` để tìm phần không khớp.

LEVEL 4 — PHÂN TÍCH (2 câu)

Câu 8.

```sql
SELECT *
FROM MUON
WHERE YEAR(NgayMuon) = 2023;
```

Hỏi: Vấn đề và cách tối ưu?

Đáp án:

- Vấn đề: dùng hàm `YEAR()` làm mất index
- Tối ưu:

```sql
WHERE NgayMuon >= '2023-01-01' AND NgayMuon < '2024-01-01';
```

Giải thích: So sánh trực tiếp giúp dùng index hiệu quả.

Câu 9.

```sql
SELECT TenSach
FROM SACH, MUON
WHERE SACH.MaSach = MUON.MaSach;
```

Hỏi: Vấn đề?

Đáp án:

- Dùng join kiểu cũ, khó đọc
- Không kiểm soát rõ loại JOIN

Tối ưu:

```sql
SELECT TenSach
FROM SACH s
INNER JOIN MUON m ON s.MaSach = m.MaSach;
```

Giải thích: Rõ ràng, dễ bảo trì.

LEVEL 5 — ĐÁNH GIÁ (1 câu)

Câu 10. So sánh 2 thiết kế:

- **Cách A:** Gộp thông tin sách + tác giả trong 1 bảng
- **Cách B:** Tách bảng `SACH` và `TACGIA`

Đáp án:

- Nên chọn **Cách B**

Giải thích:

- Tránh lặp dữ liệu tác giả
- Dễ cập nhật (1 tác giả → nhiều sách)
- Tuân theo chuẩn hóa (ít nhất 3NF)

→ Đổi lại: cần JOIN khi truy vấn, nhưng lợi ích lớn hơn chi phí.



4.3. Bảng so sánh kết quả

Tiêu chí	Cơ bản	Cải tiến	Nâng cao
Số lượng câu hỏi	Ngẫu nhiên	10 (có kiểm soát)	11 (theo cấp độ)
Phân cấp độ	Không có	Có (3 loại)	Thang Bloom đầy đủ
Context schema	Không có	Không có	Có schema cụ thể
Chất lượng đáp án	Không có	Có nhưng ngắn	Có + giải thích
Ứng dụng ôn thi	Thấp	Trung bình	Cao (sát đề thi thực)

V. Phân tích — Tại sao Prompt Nâng cao Hiệu quả hơn?

5.1. Yếu tố quyết định chất lượng prompt

1. Role Prompting — Thiết lập vai trò chuyên gia

Khi được gán vai trò cụ thể ("nhà nghiên cứu AI", "giáo viên", "giảng viên"), LLM kích hoạt phân phối xác suất token phù hợp với văn phong và kiến thức của chuyên gia đó. Thay vì phản hồi trung tính, mô hình áp dụng heuristic của domain — tương đương với việc 'thu hẹp không gian tìm kiếm' của mô hình về miền phù hợp.

2. Structural Constraint — Ràng buộc cấu trúc đầu ra

Chỉ định định dạng đầu ra (số câu, số phần, nhãn section) hoạt động như một bộ lọc hard constraint. LLM không thể "đoán mò" cấu trúc mà phải tuân thủ blueprint cho trước. Điều này loại bỏ variance không cần thiết trong output và tăng tính nhất quán.

3. Chain-of-Thought — Hướng dẫn chuỗi tư duy

Chia nhỏ tác vụ thành các bước (BƯỚC 1, BƯỚC 2...) bắt buộc LLM phải xử lý tuần tự thay vì tạo ra toàn bộ đầu ra một lần. Điều này giảm thiểu lỗi suy luận và tăng độ chính xác với các tác vụ đòi hỏi reasoning nhiều bước.

4. Context Grounding — Cung cấp ngữ cảnh cụ thể

Cung cấp schema, ví dụ cụ thể, hoặc tham chiếu tài liệu loại bỏ sự mơ hồ về miền bài toán. LLM không cần "tự đoán" context mà tập trung 100% vào reasoning trong không gian đã xác định.

5. Audience Specification — Xác định đối tượng người đọc

Nêu rõ đối tượng ("sinh viên năm nhất", "học sinh cấp 3") điều chỉnh register ngôn ngữ, độ phức tạp từ vựng, và mức độ abstraction của giải thích. Đây là yếu tố thường bị bỏ qua nhưng tác động lớn đến tính ứng dụng của output.

VI. Tổng hợp Nguyên tắc Viết Prompt Hiệu quả

Dựa trên kết quả thực nghiệm, 7 nguyên tắc sau được rút ra:

#	Nguyên tắc	Mô tả & Ví dụ
P1	**Gán vai trò chuyên gia**	"Bạn là..." + domain cụ thể. Ví dụ: "Bạn là giảng viên CSDL 10 năm kinh nghiệm"
P2	**Xác định đối tượng**	Nêu rõ người đọc. "Giải thích cho sinh viên năm nhất chưa học ML"
P3	**Ràng buộc đầu ra**	Chỉ định định dạng, số lượng, độ dài. "10 câu, mỗi câu kèm đáp án"
P4	**Cung cấp context**	Dán schema, đoạn văn, dữ liệu mẫu trực tiếp vào prompt
P5	**Chain-of-Thought**	Chia tác vụ thành các bước đánh số. Tránh prompt một khối dài
P6	**Few-shot Examples**	Cho 1-2 ví dụ mẫu về đầu ra mong muốn trước khi đặt câu hỏi thật

P7	**Iterate & Refine**	Prompt là quá trình lặp. Test → quan sát thiếu sót → bổ sung constraint

VII. Kết luận

Thực nghiệm xác nhận rằng chất lượng đầu ra của LLM phụ thuộc rất lớn vào chất lượng của prompt đầu vào — một quan hệ gần như tuyến tính. Prompt cơ bản (ngắn, mơ hồ) cho kết quả chung chung và thiếu cấu trúc. Prompt cải tiến cải thiện đáng kể bằng cách thêm ràng buộc định dạng và context. Prompt nâng cao, với sự kết hợp của role prompting, chain-of-thought, và structural constraint, tạo ra đầu ra ở cấp độ chuyên nghiệp.

Về mặt kỹ thuật, prompt engineering không chỉ là nghệ thuật diễn đạt — nó là quá trình thu hẹp không gian đầu ra của mô hình về đúng miền cần thiết, loại bỏ variance không mong muốn, và khai thác biểu diễn tri thức ẩn mà mô hình đã học trong quá trình pre-training.

Kỹ năng viết prompt hiệu quả là năng lực chiến lược trong kỷ nguyên AI — không thay thế tư duy phân tích, mà là công cụ khuếch đại nó.