BÁO CÁO BÀI TẬP 2
VIẾT PROMPT HIỆU QUẢ CHO CÁC TÁC VỤ HỌC TẬP

Thiết kế, so sánh và đánh giá prompt cơ bản - cải tiến - nâng cao

Sinh viên: Nguyễn Minh Hiếu
Mã sinh viên: 25020152
Ngành: Công nghệ Thông tin
Kỳ học: Kỳ II - Năm nhất

1. Mục tiêu và phạm vi

Bài tập này nhằm rèn luyện kỹ năng viết prompt hiệu quả để sử dụng mô hình ngôn ngữ lớn trong học tập. Prompt tốt không chỉ là prompt dài hơn, mà là prompt có mục tiêu rõ ràng, có cấu trúc, có ngữ cảnh phù hợp và có tiêu chí kiểm soát đầu ra. Trong bối cảnh AI được dùng ngày càng nhiều trong học tập, lập trình và xử lý thông tin, kỹ năng thiết kế prompt giúp sinh viên giảm độ mơ hồ của yêu cầu, tăng chất lượng câu trả lời và tiết kiệm thời gian chỉnh sửa.

Báo cáo tập trung vào ba tác vụ học tập thường gặp của sinh viên Công nghệ thông tin: tóm tắt tài liệu học thuật, giải thích khái niệm kỹ thuật phức tạp và tạo bộ câu hỏi ôn tập. Với mỗi tác vụ, báo cáo trình bày ba mức prompt: cơ bản, cải tiến và nâng cao, sau đó so sánh lý do prompt nâng cao tạo ra kết quả tốt hơn.

2. Chọn 3 tác vụ học tập

2.1. Tác vụ 1: Tóm tắt một bài đọc/tài liệu học thuật

Chủ đề: Sự phát triển của LLM, AI coding agents và CLI. Đây là tác vụ đòi hỏi khả năng đọc hiểu, lọc ý chính, sắp xếp nội dung theo mạch phát triển công nghệ và giữ văn phong học thuật ngắn gọn.

2.2. Tác vụ 2: Giải thích một khái niệm phức tạp

Chủ đề: API trong coding. Đây là tác vụ cần biến một khái niệm kỹ thuật thành lời giải thích dễ hiểu, có ví dụ minh họa và phù hợp với người mới học lập trình.

2.3. Tác vụ 3: Tạo bộ câu hỏi ôn tập

Chủ đề: OOP - Object-Oriented Programming. Đây là tác vụ cần tạo câu hỏi đa dạng theo nhiều mức độ nhận thức, bao phủ khái niệm, so sánh, vận dụng và phân tích thiết kế lớp.

3. Tiêu chí thiết kế prompt hiệu quả

> **Nguyên tắc chung:** prompt hiệu quả cần mô tả rõ vai trò của mô hình, nhiệm vụ cần làm, đối tượng người đọc, cấu trúc đầu ra, phong cách viết, độ dài mong muốn và tiêu chí tự kiểm tra.

- Vai trò: quy định mô hình đóng vai ai, ví dụ trợ lý học thuật hoặc giảng viên CNTT.
- Mục tiêu: mô tả kết quả cuối cùng cần đạt được, không chỉ nêu tên nhiệm vụ.
- Đối tượng: xác định trình độ người đọc để mô hình chọn mức diễn giải phù hợp.
- Ràng buộc: quy định độ dài, phong cách, mức thuật ngữ, ví dụ và điều cần tránh.
- Định dạng đầu ra: yêu cầu bảng, mục lớn, danh sách, đáp án hoặc checklist.
- Tự kiểm tra: yêu cầu mô hình rà lại kết quả trước khi xuất câu trả lời cuối cùng.

4. Ba phiên bản prompt cho mỗi tác vụ

Tác vụ	Prompt cơ bản	Prompt cải tiến
Tóm tắt tài liệu học thuật	Tóm tắt bài đọc về sự phát triển của LLM, AI coding agents và CLI.	Hãy tóm tắt bài đọc về sự phát triển của LLM, AI coding agents và CLI theo các ý: ý chính từng phần, xu hướng quan trọng, tác động đối với lập trình và học tập; viết trong 150-200 từ.
Giải thích API trong coding	Giải thích API là gì trong coding.	Giải thích API trong coding cho sinh viên năm nhất CNTT: API là gì, hoạt động như thế nào, vì sao cần API, kèm một ví dụ thực tế dễ hiểu.
Tạo câu hỏi ôn tập OOP	Tạo câu hỏi ôn tập về OOP.	Tạo 10 câu hỏi ôn tập OOP cho sinh viên năm nhất CNTT, gồm 4 câu trắc nghiệm, 3 câu ngắn, 3 câu vận dụng; bao gồm class, object, encapsulation, inheritance, polymorphism.

4.1. Prompt nâng cao cho tác vụ 1: Tóm tắt tài liệu học thuật

```
PROMPT NÂNG CAO CHO TÁC VỤ 1

Bạn hãy đóng vai một trợ lý học thuật có khả năng đọc hiểu tài liệu kỹ thuật, tóm tắt nội dung
phức tạp và trình bày theo văn phong ngắn gọn, logic, phù hợp với báo cáo sinh viên ngành Công
nghệ thông tin.

Nhiệm vụ của bạn là hỗ trợ tôi xử lý chủ đề:

“Sự phát triển của LLM, AI coding agents, và CLI.”

Mục tiêu:
Tạo ra một bản nội dung học thuật ngắn gọn, rõ ý, có cấu trúc hợp lý, giúp người đọc hiểu được
quá trình phát triển từ mô hình ngôn ngữ lớn, đến AI coding agents, và vai trò của giao diện
dòng lệnh CLI trong xu hướng này.

Yêu cầu xử lý:

1. Đọc và hiểu chủ đề theo hướng học thuật, không viết quá chung chung.
2. Lọc ra các ý chính quan trọng nhất, tránh liệt kê lan man.
3. Sắp xếp nội dung theo mạch logic:

   * LLM là gì và vì sao nó quan trọng.
   * LLM phát triển thành công cụ hỗ trợ lập trình như thế nào.
   * AI coding agents khác gì so với chatbot lập trình thông thường.
   * CLI đóng vai trò gì trong việc tương tác với AI coding agents.
   * Ý nghĩa của xu hướng này đối với lập trình viên và ngành phần mềm.
4. Giữ giọng văn học thuật, ngắn gọn, dễ hiểu.
5. Không sử dụng văn phong quảng cáo, không phóng đại quá mức.
6. Nếu có khái niệm kỹ thuật, hãy giải thích vừa đủ để người đọc phổ thông trong ngành CNTT
hiểu được.
7. Trước khi viết, hãy tự phân tích logic nội dung từng bước, nhưng không trình bày toàn bộ
quá trình suy luận nội bộ. Chỉ xuất ra kết quả cuối cùng đã được sắp xếp rõ ràng.

Yêu cầu đầu ra:
Hãy viết theo cấu trúc sau:

# Sự phát triển của LLM, AI coding agents, và CLI

## 1. Mở đầu

Giới thiệu ngắn gọn bối cảnh AI hiện nay và lý do chủ đề này đáng quan tâm.
```

```
## 2. Sự phát triển của LLM

Giải thích LLM là gì, khả năng chính của LLM, và vì sao LLM trở thành nền tảng cho nhiều ứng dụng AI hiện đại.

## 3. Từ LLM đến AI coding agents

Trình bày cách LLM được ứng dụng vào lập trình, từ việc gợi ý code, giải thích lỗi, đến việc hình thành các AI coding agents có khả năng thực hiện nhiều bước trong quy trình phát triển phần mềm.

## 4. Vai trò của CLI trong AI coding agents

Giải thích CLI là gì và vì sao CLI phù hợp với AI coding agents, đặc biệt trong các tác vụ như chạy lệnh, quản lý file, kiểm thử, sửa lỗi và tự động hóa quy trình lập trình.

## 5. Ý nghĩa và tác động

Phân tích ngắn gọn tác động của LLM, AI coding agents và CLI đối với lập trình viên, việc học lập trình, năng suất phát triển phần mềm và kỹ năng cần có trong tương lai.

## 6. Kết luận

Tóm tắt lại vai trò của ba yếu tố: LLM, AI coding agents và CLI trong xu hướng phát triển công nghệ phần mềm hiện đại.

Độ dài mong muốn:
Khoảng 500 từ.

Phong cách:
Học thuật, rõ ràng, mạch lạc, không quá dài dòng.

Ví dụ về phong cách viết mong muốn:
“LLM không chỉ là công cụ sinh văn bản, mà còn là nền tảng cho nhiều hệ thống AI có khả năng hỗ trợ tư duy, phân tích và tự động hóa tác vụ. Trong lĩnh vực lập trình, sự phát triển này thể hiện rõ qua các công cụ AI coding agents, nơi mô hình không chỉ trả lời câu hỏi mà còn có thể tham gia vào nhiều bước của quy trình phát triển phần mềm.”

Sau khi viết xong, hãy kiểm tra lại:

* Nội dung có đúng chủ đề không.
* Các phần có liên kết logic với nhau không.
* Có đoạn nào quá chung chung hoặc lặp ý không.
* Có giữ được giọng văn học thuật ngắn gọn không.

Nếu phát hiện phần nào chưa tốt, hãy tự chỉnh sửa trước khi đưa ra câu trả lời cuối cùng.
```

4.2. Prompt nâng cao cho tác vụ 2: Giải thích API trong coding

```
PROMPT NÂNG CAO CHO TÁC VỤ 2

2.2. Prompt nâng cao – Giải thích khái niệm phức tạp: API trong coding

Bạn hãy đóng vai một giảng viên Công nghệ thông tin có khả năng giải thích khái niệm kỹ thuật phức tạp bằng ngôn ngữ dễ hiểu, phù hợp với sinh viên mới học lập trình.

Nhiệm vụ của bạn là giải thích chủ đề:

“API trong coding.”

Mục tiêu:
```

```
Giúp người học hiểu API là gì, API dùng để làm gì trong lập trình, vì sao API quan trọng, và cách API xuất hiện trong các ứng dụng thực tế.

Đối tượng người học:
Sinh viên hoặc người mới học lập trình, đã biết khái niệm cơ bản về chương trình, hàm, dữ liệu, nhưng chưa hiểu rõ API là gì.

Yêu cầu giải thích:

1. Giải thích API bằng ngôn ngữ đơn giản, tránh dùng quá nhiều thuật ngữ khó.
2. Nếu bắt buộc dùng thuật ngữ kỹ thuật, hãy giải thích ngay sau đó.
3. Dùng ít nhất một ví dụ đời thường hoặc ẩn dụ để người học dễ hình dung.
4. Dùng ít nhất một ví dụ trong lập trình, ví dụ như ứng dụng thời tiết, đăng nhập Google, thanh toán online, hoặc lấy dữ liệu từ server.
5. Phân biệt rõ:

   * API là gì.
   * API khác gì với giao diện người dùng thông thường.
   * API giúp các phần mềm giao tiếp với nhau như thế nào.
6. Không giải thích quá dài dòng, nhưng phải đủ ý.
7. Trước khi trả lời, hãy tự xác định các ý chính cần giải thích, nhưng không trình bày toàn bộ quá trình suy luận. Chỉ đưa ra câu trả lời cuối cùng đã được sắp xếp logic.

Yêu cầu đầu ra:
Hãy trình bày theo cấu trúc sau:

# API trong coding là gì?

## 1. Giải thích ngắn gọn

Đưa ra định nghĩa API bằng cách dễ hiểu nhất.

## 2. Ví dụ đời thường

Dùng một ví dụ hoặc ẩn dụ, chẳng hạn như gọi món trong nhà hàng, ổ cắm điện, hoặc nhân viên lễ tân, để giải thích vai trò trung gian của API.

## 3. API hoạt động như thế nào trong lập trình?

Giải thích cách một chương trình gửi yêu cầu đến API và nhận dữ liệu hoặc kết quả trả về.

## 4. Ví dụ thực tế trong coding

Đưa ra một ví dụ cụ thể, chẳng hạn:

* Ứng dụng thời tiết gọi API để lấy nhiệt độ.
* Website dùng API đăng nhập bằng Google.
* Ứng dụng bán hàng dùng API thanh toán.

## 5. Vì sao API quan trọng?

Trình bày ngắn gọn các lợi ích:

* Kết nối các hệ thống phần mềm.
* Tái sử dụng chức năng có sẵn.
* Giúp lập trình viên không phải tự xây dựng mọi thứ từ đầu.
* Hỗ trợ phát triển ứng dụng nhanh hơn.

## 6. Kết luận ngắn

Tóm tắt lại API bằng 2–3 câu dễ nhớ.

Độ dài mong muốn:
```

```
Khoảng 500 từ.

Phong cách:
Dễ hiểu, có tính giảng giải, phù hợp với người mới học lập trình.

Ví dụ về phong cách viết mong muốn:
“Có thể hiểu API giống như một người phục vụ trong nhà hàng. Người dùng không cần vào bếp để nấu món ăn, mà chỉ cần gọi món thông qua người phục vụ. Tương tự, một chương trình không cần biết toàn bộ hệ thống bên trong hoạt động ra sao, mà chỉ cần gửi yêu cầu đến API và nhận kết quả trả về.”

Sau khi viết xong, hãy tự kiểm tra:

* Người mới học lập trình có thể hiểu được không.
* Ví dụ có rõ ràng không.
* Có giải thích được API dùng để làm gì không.
* Có tránh giải thích quá kỹ thuật hoặc quá trừu tượng không.
```

4.3. Prompt nâng cao cho tác vụ 3: Tạo bộ câu hỏi ôn tập OOP

```
2.3. Prompt nâng cao – Tạo bộ câu hỏi ôn tập: OOP

Bạn hãy đóng vai một giảng viên môn Lập trình hướng đối tượng, có kinh nghiệm thiết kế câu hỏi ôn tập cho sinh viên Công nghệ thông tin.

Nhiệm vụ của bạn là tạo một bộ câu hỏi ôn tập về chủ đề:

“OOP – Object-Oriented Programming.”

Mục tiêu:
Tạo bộ câu hỏi giúp sinh viên ôn tập các kiến thức quan trọng của lập trình hướng đối tượng, bao gồm khái niệm cơ bản, khả năng phân biệt, áp dụng vào ví dụ, và phân tích tình huống code.

Đối tượng người học:
Sinh viên đang học môn Lập trình hướng đối tượng ở mức cơ bản đến trung bình.

Yêu cầu nội dung:
Bộ câu hỏi cần bao phủ các chủ đề sau:

1. Class và Object.
2. Encapsulation – đóng gói.
3. Inheritance – kế thừa.
4. Polymorphism – đa hình.
5. Abstraction – trừu tượng hóa.
6. Abstract class và Interface.
7. Constructor.
8. Overloading và Overriding.
9. Quan hệ giữa các lớp: association, aggregation, composition, dependency.
10. Một số nguyên lý thiết kế cơ bản như SOLID nếu phù hợp.

Yêu cầu về dạng câu hỏi:
Tạo tổng cộng 20 câu hỏi, chia thành các nhóm sau:

## Nhóm 1: Câu hỏi nhận biết

Tạo 5 câu hỏi kiểm tra định nghĩa và khái niệm cơ bản.

## Nhóm 2: Câu hỏi thông hiểu

Tạo 5 câu hỏi yêu cầu giải thích, so sánh hoặc phân biệt các khái niệm.
```

```
## Nhóm 3: Câu hỏi vận dụng

Tạo 5 câu hỏi yêu cầu áp dụng OOP vào tình huống cụ thể hoặc ví dụ lập trình.

## Nhóm 4: Câu hỏi phân tích

Tạo 5 câu hỏi yêu cầu đọc tình huống, phân tích thiết kế lớp, phát hiện lỗi thiết kế hoặc giải thích quan hệ giữa các lớp.

Yêu cầu về đáp án:
Sau mỗi câu hỏi, hãy cung cấp:

* Đáp án ngắn gọn hoặc gợi ý trả lời.
* Với câu hỏi phân tích, hãy nêu các ý chính cần có trong câu trả lời.
* Nếu là câu hỏi có nhiều cách trả lời đúng, hãy ghi rõ tiêu chí chấm điểm thay vì chỉ đưa một đáp án duy nhất.

Yêu cầu về độ khó:

* 30% câu hỏi dễ.
* 50% câu hỏi trung bình.
* 20% câu hỏi khó.
  Hãy ghi nhãn độ khó cho từng câu: [Dễ], [Trung bình], hoặc [Khó].

Yêu cầu trình bày:
Hãy trình bày theo cấu trúc rõ ràng:

# Bộ câu hỏi ôn tập OOP

## Nhóm 1: Nhận biết

Câu 1...
Đáp án/Gợi ý...

## Nhóm 2: Thông hiểu

Câu 6...
Đáp án/Gợi ý...

## Nhóm 3: Vận dụng

Câu 11...
Đáp án/Gợi ý...

## Nhóm 4: Phân tích

Câu 16...
Đáp án/Gợi ý...

Phong cách:
Rõ ràng, học thuật vừa phải, phù hợp để dùng làm tài liệu ôn tập trước kiểm tra hoặc thi cuối kỳ.

Lưu ý:
Không chỉ tạo câu hỏi lý thuyết đơn giản. Cần có sự đa dạng giữa định nghĩa, so sánh, ví dụ thực tế, tình huống code, và phân tích thiết kế.

Ví dụ về dạng câu hỏi mong muốn:
“[Trung bình] Phân biệt overloading và overriding trong OOP. Cho một ví dụ ngắn minh họa cho mỗi trường hợp.”

Ví dụ về cách trả lời mong muốn:
“Overloading là việc nhiều phương thức có cùng tên nhưng khác tham số trong cùng một lớp.
```

```
Overriding là việc lớp con định nghĩa lại phương thức đã có ở lớp cha. Câu trả lời tốt cần nêu
được sự khác nhau về vị trí xuất hiện, tham số, mục đích sử dụng và thời điểm quyết định
phương thức được gọi.”

Trước khi đưa ra kết quả cuối cùng, hãy tự kiểm tra:

* Có đủ 20 câu hỏi không.
* Có đủ 4 nhóm mức độ nhận thức không.
* Có bao phủ các chủ đề OOP quan trọng không.
* Câu hỏi có đa dạng không.
* Đáp án hoặc gợi ý có đủ để người học tự ôn tập không.
```

5. So sánh kết quả và phân tích

Tiêu chí	Prompt cơ bản	Prompt cải tiến	Prompt nâng cao
Độ rõ mục tiêu	Thấp: chỉ nêu nhiệm vụ chung.	Trung bình: có thêm ý cần xử lý.	Cao: có vai trò, mục tiêu, đối tượng, phạm vi và tiêu chí kiểm tra.
Kiểm soát đầu ra	Yếu: mô hình tự đoán độ dài và định dạng.	Tốt hơn: có yêu cầu số ý hoặc độ dài.	Rất tốt: có cấu trúc đầu ra, phong cách, độ dài và checklist tự kiểm tra.
Mức phù hợp với người học	Không xác định rõ người đọc.	Có định hướng cho sinh viên năm nhất.	Rõ ràng: chỉ ra trình độ người học, mức thuật ngữ, ví dụ và mục đích sử dụng.
Khả năng dùng lại	Khó dùng lại cho bài tương tự.	Có thể dùng lại nhưng cần chỉnh.	Dễ dùng lại vì prompt đóng vai trò như một bản đặc tả nhiệm vụ hoàn chỉnh.

5.1. Nhận xét tổng quát

Prompt cơ bản thường chỉ nêu nhiệm vụ chung nên mô hình phải tự đoán nhiều yếu tố như đối tượng người đọc, độ dài, cấu trúc và mức độ chi tiết. Vì vậy, đầu ra dễ ngắn, chung chung hoặc lệch trọng tâm. Prompt cải tiến đã tốt hơn vì có thêm danh sách yêu cầu và giới hạn cụ thể, nhưng vẫn chưa mô tả đầy đủ vai trò, mục tiêu học tập và tiêu chí tự kiểm tra.

Prompt nâng cao là phiên bản hiệu quả nhất vì hoạt động như một bản đặc tả nhiệm vụ. Ba prompt nâng cao trong báo cáo đều chỉ rõ vai trò, mục tiêu, đối tượng, cấu trúc đầu ra, phong cách, độ dài mong muốn và yêu cầu tự rà soát. Điều này giúp mô hình tạo kết quả có tính ổn định hơn, ít mơ hồ hơn và phù hợp hơn với mục đích học tập.

5.2. Vì sao prompt nâng cao phù hợp với từng tác vụ?

- Với tác vụ tóm tắt, prompt nâng cao không chỉ yêu cầu tóm tắt mà còn quy định mạch logic từ LLM đến AI coding agents và CLI. Vì vậy, nội dung đầu ra dễ có cấu trúc học thuật hơn.
- Với tác vụ giải thích API, prompt nâng cao xác định rõ trình độ người học và bắt buộc có ví dụ đời thường lẫn ví dụ coding. Điều này giúp giảm tính trừu tượng của khái niệm.
- Với tác vụ tạo câu hỏi OOP, prompt nâng cao quy định số lượng, nhóm nhận thức, độ khó, chủ đề bao phủ và đáp án/gợi ý. Vì vậy, bộ câu hỏi có thể dùng trực tiếp để ôn tập hoặc kiểm tra.

6. Kiểm soát dữ liệu và các điểm đã hoàn thiện

So với bản trước, báo cáo này được chỉnh để hoàn chỉnh hơn ở ba điểm. Thứ nhất, các prompt nâng cao đã được mở rộng đúng theo yêu cầu mới, không còn quá ngắn so với bản chất tác vụ. Thứ hai, các mã trích dẫn dạng placeholder như [page:1], [page:2] hoặc [web:41] được loại bỏ khỏi thân bài để tránh làm báo cáo giống bản nháp. Thứ ba, phần phân tích được cập nhật để khớp với nội dung prompt nâng cao mới, thay vì chỉ nhận xét chung.

Bản báo cáo cũng bổ sung phần tiêu chí thiết kế prompt hiệu quả, giúp người đọc hiểu vì sao các prompt nâng cao được xem là tốt hơn. Nhờ đó, bài không chỉ liệt kê prompt mà còn thể hiện được năng lực phân tích, so sánh và rút ra nguyên tắc thiết kế prompt.

7. Nguyên tắc và mẹo viết prompt hiệu quả

7.1. Nguyên tắc rút ra

- Nêu rõ mục tiêu của tác vụ ngay từ đầu.
- Xác định rõ đối tượng người đọc hoặc người học.
- Chia yêu cầu thành các phần nhỏ, có thứ tự logic.
- Chỉ rõ định dạng đầu ra mong muốn.
- Thêm ràng buộc về độ dài, phong cách và mức độ chi tiết.
- Dùng ví dụ khi muốn mô hình bắt chước phong cách trả lời.
- Yêu cầu mô hình tự kiểm tra kết quả trước khi trả lời cuối cùng.

7.2. Mẹo thực tế

- Với tác vụ tóm tắt, hãy yêu cầu số lượng ý chính, mạch logic và độ dài cụ thể.
- Với tác vụ giải thích, hãy yêu cầu ví dụ, ẩn dụ và tránh thuật ngữ chưa giải thích.
- Với tác vụ tạo câu hỏi, hãy quy định số lượng, mức độ khó, kiểu câu hỏi và đáp án hoặc gợi ý chấm điểm.
- Nếu kết quả quá chung chung, hãy thêm vai trò, bối cảnh và cấu trúc đầu ra.
- Nếu kết quả quá dài hoặc lan man, hãy giới hạn độ dài và yêu cầu chia thành mục rõ ràng.

8. Kết luận

Qua ba tác vụ học tập, có thể thấy prompt hiệu quả không đơn thuần là prompt dài hơn mà là prompt rõ hơn, có cấu trúc hơn và kiểm soát đầu ra tốt hơn. Prompt nâng cao giúp mô hình hiểu rõ nhiệm vụ, người đọc, phạm vi xử lý và tiêu chí chất lượng. Đối với sinh viên Công nghệ thông tin, kỹ năng viết prompt có ý nghĩa thực tế vì có thể hỗ trợ đọc tài liệu kỹ thuật, học khái niệm mới, ôn tập lập trình và làm việc với các công cụ AI trong tương lai.

9. Tài liệu tham khảo

[1] Anthropic (2025) *Effective context engineering for AI agents*. Available at: https://www.anthropic.com/engineering/effective-context-engineering-for-ai-agents (Accessed: 4 June 2026).

[2] Prompting Guide (2026) *Few-shot prompting*. Available at: https://www.promptingguide.ai/techniques/fewshot (Accessed: 4 June 2026).

[3] Chen, B., Zhang, Z., Langrené, N. and Zhu, S. (2025) 'Unleashing the potential of prompt engineering for large language models', *Patterns*, 6(6), 101260. doi: 10.1016/j.patter.2025.101260.