

Prompt Engineering

Author: Lee Boonstra



SÁCH MỚI.NET - EBOOK



THƯ VIỆN EBOOK **SÁCH MỚI.NET**



HÀNG NGHÌN ĐẦU SÁCH HAY ĐANG CHỜ BẠN
WWW.SACHMOI.NET

Lời cảm ơn

Những người đóng góp nội dung

- Michael Sherman
- Yuan Cao
- Erick Armbrust
- Anant Nawalgaria
- Antonio Gulli
- Simone Cammel

Giám tuyển và Biên tập viên

- Antonio Gulli
- Anant Nawalgaria
- Grace Mollison
- Người viết kỹ thuật
- Joey Haymaker

Thiết kế bìa

- Michael Lanning

Giới thiệu

Bạn không cần phải là một nhà khoa học dữ liệu hay một kỹ sư máy học – bất kỳ ai cũng có thể viết một prompt [1].

Khi nghĩ về đầu vào và đầu ra của một mô hình ngôn ngữ lớn (LLM), một prompt văn bản (đôi khi đi kèm với các phương thức khác như prompt hình ảnh) là đầu vào mà mô hình sử dụng để dự đoán một đầu ra cụ thể. Bạn không cần phải là một nhà khoa học dữ liệu hay một kỹ sư máy học - bất kỳ ai cũng có thể viết một prompt. Tuy nhiên, việc tạo ra một prompt hiệu quả nhất có thể phức tạp. Nhiều khía cạnh của prompt ảnh hưởng đến hiệu quả của nó: mô hình bạn sử dụng, dữ liệu huấn luyện của mô hình, cấu hình mô hình, cách bạn lựa chọn từ ngữ, phong cách và giọng văn, cấu trúc và ngữ cảnh đều quan trọng. Do đó, kỹ thuật prompt là một quá trình lặp đi lặp lại. Các prompt không đầy đủ có thể dẫn đến các câu trả lời mơ hồ, không chính xác và có thể cản trở khả năng của mô hình trong việc cung cấp đầu ra có ý nghĩa.

Khi bạn trò chuyện với chatbot Gemini [2], bạn thực chất đang viết các prompt. Tuy nhiên, sách trắng này tập trung vào việc viết prompt cho mô hình Gemini trong Vertex AI hoặc bằng cách sử dụng API, bởi vì bằng cách prompt trực tiếp mô hình, bạn sẽ có quyền truy cập vào các cấu hình như temperature, v.v.

Sách trắng này thảo luận chi tiết về kỹ thuật prompt. Chúng ta sẽ xem xét các kỹ thuật prompting khác nhau để giúp bạn bắt đầu và chia sẻ các mẹo cũng như các phương pháp hay nhất để trở thành một chuyên gia prompting. Chúng ta cũng sẽ thảo luận về một số thách thức bạn có thể gặp phải khi tạo prompt.

Kỹ thuật prompt

Hãy nhớ lại cách một LLM hoạt động; nó là một cỗ máy dự đoán. Mô hình nhận một đoạn văn bản tuần tự làm đầu vào và sau đó dự đoán token [3] tiếp theo nên là gì, dựa trên dữ liệu mà nó đã được huấn luyện. LLM được vận hành để thực hiện điều này lặp đi lặp lại, thêm token vừa được dự đoán vào cuối đoạn văn bản tuần tự để dự đoán token tiếp theo. Việc dự đoán token tiếp theo dựa trên mối quan hệ giữa những gì có trong các token trước đó và những gì LLM đã thấy trong quá trình huấn luyện của nó.

Khi bạn viết một prompt, bạn đang cố gắng thiết lập để LLM dự đoán đúng chuỗi token. Kỹ thuật prompt là quá trình thiết kế các prompt chất lượng cao nhằm hướng dẫn LLM tạo ra các kết quả chính xác. Quá trình này bao gồm việc thử nghiệm để tìm ra prompt tốt nhất, tối ưu hóa độ dài prompt, và đánh giá văn phong cũng như cấu trúc của prompt trong mối tương quan với nhiệm vụ. Trong bối cảnh xử lý ngôn ngữ tự nhiên và LLM, prompt là một đầu vào được cung cấp cho mô hình để tạo ra một phản hồi hoặc dự đoán.

Những prompt này có thể được sử dụng để đạt được nhiều loại nhiệm vụ để hiểu và tạo sinh khác nhau như tóm tắt văn bản, trích xuất thông tin, hỏi đáp, phân loại văn bản, dịch ngôn ngữ hoặc code, tạo code và tài liệu hóa code hoặc suy luận.

Vui lòng tham khảo các hướng dẫn prompting của Google [4], [5] với các ví dụ prompting đơn giản và hiệu quả.

Khi thực hiện kỹ thuật prompt, bạn sẽ bắt đầu bằng việc chọn một mô hình. Các prompt có thể cần được tối ưu hóa cho mô hình cụ thể của bạn, bất kể bạn sử dụng các mô hình ngôn ngữ Gemini trong Vertex AI, GPT, Claude, hay một mô hình mã nguồn mở như Gemma hoặc LLaMA.

Bên cạnh prompt, bạn cũng sẽ cần phải thử nghiệm với các cấu hình khác nhau của một LLM.

Cấu hình đầu ra của LLM

Một khi bạn đã chọn mô hình, bạn sẽ cần phải tìm hiểu về cấu hình của mô hình. Hầu hết các LLM đều đi kèm với nhiều tùy chọn cấu hình khác nhau để kiểm soát đầu ra của LLM. Kỹ thuật prompt hiệu quả đòi hỏi phải thiết lập các cấu hình này một cách tối ưu cho nhiệm vụ của bạn.

Độ dài đầu ra

Một thiết lập cấu hình quan trọng là số lượng token được tạo ra trong một phản hồi. Việc tạo ra nhiều token hơn đòi hỏi nhiều tính toán hơn từ LLM, dẫn đến tiêu thụ năng lượng cao hơn, thời gian phản hồi có thể chậm hơn và chi phí cao hơn.

Việc giảm độ dài đầu ra của LLM không khiến LLM trở nên súc tích hơn về mặt văn phong hay văn bản đầu ra mà nó tạo ra, nó chỉ khiến LLM ngừng dự đoán thêm token khi đạt đến giới hạn. Nếu nhu cầu của bạn đòi hỏi một đầu ra ngắn, bạn cũng cần phải thiết kế prompt của mình để sao cho phù hợp với điều đó.

Việc giới hạn độ dài đầu ra đặc biệt quan trọng đối với một số kỹ thuật prompting của LLM, như ReAct, nơi LLM sẽ tiếp tục phát ra các token vô dụng sau khi đã có phản hồi bạn muốn.

Hãy lưu ý, việc tạo ra nhiều token hơn đòi hỏi nhiều tính toán hơn từ LLM, dẫn đến tiêu thụ năng lượng cao hơn và thời gian phản hồi có thể chậm hơn, điều này dẫn đến chi phí cao hơn.

Các tham số điều khiển lấy mẫu (Sampling controls)

LLM không chính thức dự đoán một token duy nhất. Thay vào đó, LLM dự đoán xác suất cho token tiếp theo có thể là gì, với mỗi token trong từ vựng của LLM nhận được một xác suất. Những xác suất token đó sau đó được lấy mẫu để xác định token tiếp theo được tạo ra sẽ là gì. Temperature, top-K, và top-P là các cài đặt cấu hình phổ biến nhất xác định cách xử lý xác suất token được dự đoán để chọn ra một token đầu ra duy nhất.

Temperature (Mức độ ngẫu nhiên)

Temperature kiểm soát mức độ ngẫu nhiên trong việc lựa chọn token. Mức temperature thấp hơn phù hợp cho các prompt mong đợi một phản hồi có tính xác định cao hơn, trong khi mức temperature cao hơn có thể dẫn đến các kết quả đa dạng hoặc bất ngờ hơn. Một mức temperature bằng 0 (giải mã tham lam - greedy decoding) là xác định: token có xác suất cao nhất luôn được chọn (tuy nhiên, lưu ý rằng nếu hai token có cùng xác suất dự đoán cao nhất, tùy thuộc vào cách xử lý trường hợp hòa, bạn có thể không nhận được cùng một đầu ra với temperature là 0).

Mức temperature gần giá trị tối đa có xu hướng tạo ra đầu ra ngẫu nhiên hơn. Và khi temperature ngày càng cao, tất cả các token trở nên có khả năng được chọn như nhau để trở thành token được dự đoán tiếp theo.

Tham số temperature của Gemini có thể được hiểu theo cách tương tự như hàm softmax [\[6\]](#) được sử dụng trong học máy. Một cài đặt temperature thấp phản ánh một temperature softmax thấp (T), nhấn mạnh vào một temperature duy nhất, được ưu tiên với độ chắc chắn cao. Một cài đặt temperature cao hơn của Gemini giống như một temperature softmax cao, làm cho một phạm vi temperature rộng hơn xung quanh cài đặt đã chọn trở nên chấp nhận được hơn. Sự không chắc chắn gia tăng này phù hợp với các kịch bản mà một temperature cứng nhắc, chính xác có thể không cần thiết, ví dụ như khi thử nghiệm với các đầu ra sáng tạo.

Top-K và top-P

Top-K [7] và top-P [8] (còn được gọi là nucleus sampling) [9] là hai cài đặt lấy mẫu được sử dụng trong LLM để giới hạn token được dự đoán tiếp theo phải đến từ các token có xác suất dự đoán cao nhất. Giống như temperature, các cài đặt lấy mẫu này kiểm soát sự ngẫu nhiên và đa dạng của văn bản được tạo ra.

- Lấy mẫu top-K chọn K token có khả năng cao nhất từ phân phối dự đoán của mô hình. Top-K càng cao, đầu ra của mô hình càng sáng tạo và đa dạng; top-K càng thấp, đầu ra của mô hình càng bị hạn chế và thực tế hơn. Top-K bằng 1 tương đương với giải mã tham lam.
- Lấy mẫu top-P chọn các token hàng đầu có tổng xác suất tích lũy không vượt quá một giá trị nhất định (P). Các giá trị cho P dao động từ 0 (giải mã tham lam) đến 1 (tất cả các token trong từ vựng của LLM).

Cách tốt nhất để chọn giữa top-K và top-P là thử nghiệm với cả hai phương pháp (hoặc cả hai cùng nhau) và xem phương pháp nào tạo ra kết quả bạn đang tìm kiếm.

Tổng hợp lại

Việc lựa chọn giữa top-K, top-P, temperature và số lượng token để tạo ra, phụ thuộc vào ứng dụng cụ thể và kết quả mong muốn, và các cài đặt này đều ảnh hưởng lẫn nhau. Điều quan trọng là bạn phải hiểu cách mô hình bạn chọn kết hợp các cài đặt lấy mẫu khác nhau với nhau.

Nếu temperature, top-K và top-P đều có sẵn (như trong Vertex Studio), các token đáp ứng cả tiêu chí top-K và top-P sẽ là ứng cử viên cho token được dự đoán tiếp theo, và sau đó temperature được áp dụng để lấy mẫu từ các token đã vượt qua tiêu chí top-K và top-P. Nếu chỉ có top-K hoặc top-P, hành vi vẫn như cũ nhưng chỉ sử dụng cài đặt top-K hoặc P đó.

Nếu temperature không có sẵn, bất kỳ token nào đáp ứng tiêu chí top-K và/hoặc top-P sau đó sẽ được chọn ngẫu nhiên để tạo ra một token được dự đoán tiếp theo duy nhất.

Ở các cài đặt cục đoạn của một giá trị cấu hình lấy mẫu, cài đặt lấy mẫu đó hoặc sẽ hủy bỏ các cài đặt cấu hình khác hoặc trở nên không liên quan.

- Nếu bạn đặt temperature thành 0, top-K và top-P trở nên không liên quan – token có khả năng cao nhất sẽ trở thành token được dự đoán tiếp theo. Nếu bạn đặt temperature cực kỳ cao (trên 1, thường là trong khoảng 10), temperature trở nên không liên quan và bất kỳ token nào vượt qua tiêu chí top-K và/hoặc top-P sau đó sẽ được lấy mẫu ngẫu nhiên để chọn ra một token được dự đoán tiếp theo.
- Nếu bạn đặt top-K thành 1, temperature và top-P trở nên không liên quan. Chỉ có một token vượt qua tiêu chí top-K, và token đó là token được dự đoán tiếp theo. Nếu bạn đặt top-K cực kỳ cao, ví dụ như bằng kích thước từ vựng của LLM, bất kỳ token nào có xác suất khác không sẽ đáp ứng tiêu chí top-K và không có token nào bị loại bỏ.
- Nếu bạn đặt top-P thành 0 (hoặc một giá trị rất nhỏ), hầu hết các cách triển khai lấy mẫu của LLM sẽ chỉ xem xét token có khả năng cao nhất để đáp ứng tiêu chí top-P, làm cho temperature và top-K trở nên không liên quan. Nếu bạn đặt top-P thành 1, bất kỳ token nào có xác suất khác không sẽ đáp ứng tiêu chí top-P, và không có token nào bị loại bỏ.

Điểm khởi đầu chung, một mức temperature là .2, top-P là .95, và top-K là 30 sẽ cho bạn kết quả tương đối mạch lạc, có thể sáng tạo nhưng không quá mức. Nếu bạn muốn kết quả đặc biệt sáng tạo, hãy thử bắt đầu với temperature là .9, top-P là .99, và top-K là 40. Và nếu bạn muốn kết quả ít sáng tạo hơn, hãy thử bắt đầu với temperature là .1, top-P là .9, và top-K là 20. Cuối cùng, nếu nhiệm vụ của bạn luôn có một câu trả lời đúng duy nhất (ví dụ: giải một bài toán), hãy bắt đầu với temperature là 0.

Lưu ý : Với nhiều sự tự do hơn (temperature, top-K, top-P và số token đầu ra cao hơn), LLM có thể tạo ra văn bản ít liên quan hơn.

Cảnh báo : Bạn đã bao giờ thấy một phản hồi kết thúc bằng một lượng lớn các từ đệm chưa? Đây còn được gọi là "lỗi vòng lặp lặp lại", một vấn đề phổ biến trong các Mô hình Ngôn ngữ Lớn, nơi mô hình bị kẹt trong một chu kỳ, lặp đi lặp lại cùng một từ (đệm), cụm từ hoặc cấu trúc câu, thường trở nên tồi tệ hơn do cài đặt temperature và top-k/cài đặt top-p không phù hợp.

Điều này có thể xảy ra ở cả cài đặt temperature thấp và cao, mặc dù vì những lý do khác nhau. Ở temperature thấp, mô hình trở nên quá xác định, bám cứng vào con đường có xác suất cao nhất, điều này có thể dẫn đến một vòng lặp nếu con đường đó quay trở lại văn bản đã tạo trước đó. Ngược lại, ở temperature cao, đầu ra của mô hình trở nên quá ngẫu nhiên, làm tăng xác suất rằng một từ hoặc cụm từ được chọn ngẫu nhiên, tình cờ, sẽ dẫn trở lại một trạng thái trước đó, tạo ra một vòng lặp do có quá nhiều tùy chọn có sẵn.

Trong cả hai trường hợp, quá trình lấy mẫu của mô hình bị "kẹt", dẫn đến đầu ra đơn điệu và vô ích cho đến khi cửa sổ đầu ra được lấp đầy. Việc giải

quyết vấn đề này thường đòi hỏi sự tinh chỉnh cẩn thận với các giá trị temperature và top-k/top-p để tìm ra sự cân bằng tối ưu giữa tính xác định và tính ngẫu nhiên.

Các kỹ thuật prompting

LLM được tinh chỉnh để tuân theo các hướng dẫn và được huấn luyện trên một lượng lớn dữ liệu để chúng có thể hiểu một prompt và tạo ra một câu trả lời. Nhưng LLM không hoàn hảo; prompt của bạn càng rõ ràng, LLM càng dễ dàng dự đoán văn bản tiếp theo có khả năng xảy ra. Ngoài ra, các kỹ thuật cụ thể tận dụng cách LLM được huấn luyện và cách chúng hoạt động sẽ giúp bạn nhận được kết quả phù hợp từ LLM.

Bây giờ chúng ta đã hiểu kỹ thuật prompt là gì và cần những gì, hãy đi sâu vào một số ví dụ về các kỹ thuật prompting quan trọng nhất.

Prompting chung / zero shot

Một prompt zero-shot [\[10\]](#) là loại prompt đơn giản nhất. Nó chỉ cung cấp một mô tả về một nhiệm vụ và một số văn bản để LLM bắt đầu. Đầu vào này có thể là bất cứ thứ gì: một câu hỏi, phần đầu của một câu chuyện, hoặc các hướng dẫn. Tên gọi zero-shot có nghĩa là “không có ví dụ mẫu”.

Hãy sử dụng Vertex AI Studio (cho Ngôn ngữ) trong Vertex AI [\[11\]](#), nơi cung cấp một sân chơi để kiểm tra các prompt. Trong Bảng 1, bạn sẽ thấy một ví dụ về prompt zero-shot để phân loại các bài đánh giá phim.

Định dạng bảng được sử dụng dưới đây là một cách tuyệt vời để ghi lại tài liệu về các prompt. Các prompt của bạn có thể sẽ trải qua nhiều lần lặp lại trước khi được đưa vào codebase, vì vậy điều quan trọng là phải theo dõi công việc kỹ thuật prompt của bạn một cách có kỷ luật và cấu trúc. Thông tin thêm về định dạng bảng này, tầm quan trọng của việc theo dõi công việc kỹ thuật prompt, và quy trình phát triển prompt sẽ có trong phần Thực hành tốt nhất ở cuối chương này (“Ghi lại tài liệu về các lần thử nghiệm prompt khác nhau”).

Temperature của mô hình nên được đặt ở mức thấp, vì không cần sự sáng tạo, và chúng tôi sử dụng các giá trị top-K và top-P mặc định của gemini-pro, điều này thực chất vô hiệu hóa cả hai cài đặt (xem “Cấu hình đầu ra của LLM” ở trên). Hãy chú ý đến đầu ra được tạo ra. Các từ disturbing (gây khó chịu) và masterpiece (kiệt tác) nên làm cho việc dự đoán trở nên phức tạp hơn một chút, vì cả hai từ đều được sử dụng trong cùng một câu.

Name	1_1_movie_classification		
Goal	Classify movie reviews as positive, neutral or negative.		
Model	gemini-pro		
Temperature	0.1	Token Limit	5
Top-K	N/A	Top-P	1
Prompt	<p>Classify movie reviews as POSITIVE, NEUTRAL or NEGATIVE.</p> <p>Review: "Her" is a disturbing study revealing the direction humanity is headed if AI is allowed to keep evolving, unchecked. I wish there were more movies like this masterpiece.</p> <p>Sentiment:</p>		
Output	POSITIVE		

Bảng 1. Một ví dụ về prompting zero-shot

Khi zero-shot không hoạt động, bạn có thể cung cấp các minh họa hoặc ví dụ trong prompt, điều này dẫn đến prompting “one-shot” và “few-shot”.

One-shot & few-shot

Khi tạo prompt cho các mô hình AI, việc cung cấp ví dụ sẽ rất hữu ích. Những ví dụ này có thể giúp mô hình hiểu những gì bạn đang yêu cầu. Các ví dụ đặc biệt hữu ích khi bạn muốn định hướng mô hình theo một cấu trúc hoặc mẫu đầu ra nhất định.

- Prompt one-shot, cung cấp một ví dụ duy nhất, do đó có tên là one-shot. Ý tưởng là mô hình có một ví dụ để bắt chước nhằm hoàn thành tốt nhất nhiệm vụ.
- Prompt few-shot [\[12\]](#) cung cấp nhiều ví dụ cho mô hình. Cách tiếp cận này cho mô hình thấy một mẫu mà nó cần phải tuân theo. Ý tưởng tương tự như one-shot, nhưng nhiều ví dụ về mẫu mong muốn sẽ tăng cơ hội mô hình tuân theo mẫu đó.

Số lượng ví dụ bạn cần cho prompting few-shot phụ thuộc vào một vài yếu tố, bao gồm độ phức tạp của nhiệm vụ, chất lượng của các ví dụ, và khả năng của mô hình AI tạo sinh (gen AI) bạn đang sử dụng.

Theo nguyên tắc chung, bạn nên sử dụng ít nhất ba đến năm ví dụ cho prompting few-shot. Tuy nhiên, bạn có thể cần sử dụng nhiều ví dụ hơn cho các nhiệm vụ phức tạp hơn, hoặc bạn có thể cần sử dụng ít hơn do giới hạn độ dài đầu vào của mô hình.

Bảng 2 cho thấy một ví dụ về prompt few-shot, hãy sử dụng cùng các cài đặt cấu hình mô hình gemini-pro như trước, ngoại trừ việc tăng giới hạn token để đáp ứng nhu cầu cho một phản hồi dài hơn.

Goal	Parse pizza orders to JSON		
Model	gemini-pro		
Temperature	0.1	Token Limit	250
Top-K	N/A	Top-P	1
Prompt	<p>Parse a customer's pizza order into valid JSON:</p> <p>EXAMPLE:</p> <p>I want a small pizza with cheese, tomato sauce, and pepperoni.</p> <p>JSON Response:</p> <pre>{ "size": "small", "type": "normal", "ingredients": ["cheese", "tomato sauce", "peperoni"] }</pre> <p>EXAMPLE:</p> <p>Can I get a large pizza with tomato sauce, basil and mozzarella</p> <pre>{</pre>		

	<pre>"size": "large", "type": "normal", "ingredients": [["tomato sauce", "bازل", "mozzarella"]] } Now, I would like a large pizza, with the first half cheese and mozzarella. And the other tomato sauce, ham and pineapple. JSON Response:</pre>
Output	<pre>``` "size": "large", "type": "half-half", "ingredients": [["cheese", "mozzarella"], ["tomato sauce", }"ham", "pineapple"]] ```</pre>

Bảng 2. Một ví dụ về prompting few-shot

Khi bạn chọn các ví dụ cho prompt của mình, hãy sử dụng các ví dụ có liên quan đến nhiệm vụ bạn muốn thực hiện. Các ví dụ nên đa dạng, chất lượng cao và được viết tốt. Một sai lầm nhỏ có thể làm mô hình bối rối và sẽ dẫn đến đầu ra không mong muốn.

Nếu bạn đang cố gắng tạo ra đầu ra có thể hoạt động tốt với nhiều loại đầu vào khác nhau, thì điều quan trọng là phải bao gồm các trường hợp biên (edge cases) trong ví dụ của bạn. Các trường hợp biên là những đầu vào bất thường hoặc không mong đợi, nhưng mô hình vẫn phải có khả năng xử lý được.

Prompting theo hệ thống, ngữ cảnh và vai trò

Prompting theo hệ thống, ngữ cảnh và vai trò đều là các kỹ thuật được sử dụng để hướng dẫn cách LLM tạo ra văn bản, nhưng chúng tập trung vào các khía cạnh khác nhau:

- Prompting theo hệ thống (System prompting) thiết lập bối cảnh và mục đích tổng thể cho mô hình ngôn ngữ. Nó định nghĩa “bức tranh lớn” về những gì mô hình nên làm, như dịch một ngôn ngữ, phân loại một bài đánh giá, v.v.
- Prompting theo ngữ cảnh (Contextual prompting) cung cấp các chi tiết cụ thể hoặc thông tin nền tảng liên quan đến cuộc trò chuyện hoặc nhiệm vụ hiện tại. Nó giúp mô hình hiểu được các sắc thái của những gì đang được yêu cầu và điều chỉnh phản hồi cho phù hợp.
- Prompting theo vai trò (Role prompting) gán một nhân vật hoặc danh tính cụ thể cho mô hình ngôn ngữ để áp dụng. Điều này giúp mô hình tạo ra các phản hồi nhất quán với vai trò được giao và kiến thức cũng như hành vi liên quan của nó.

Có thể có sự chồng chéo đáng kể giữa prompting theo hệ thống, ngữ cảnh và vai trò. Ví dụ, một prompt gán vai trò cho hệ thống cũng có thể có một ngữ cảnh. Tuy nhiên, mỗi loại prompt phục vụ một mục đích chính hơi khác nhau:

- Prompt hệ thống: Xác định các khả năng cơ bản và mục đích bao quát của mô hình.
- Prompt ngữ cảnh: Cung cấp thông tin tức thời, cụ thể cho nhiệm vụ để hướng dẫn phản hồi. Nó rất cụ thể cho nhiệm vụ hoặc đầu vào hiện tại, vốn có tính động.

→ Prompt vai trò: Định hình phong cách và giọng văn đầu ra của mô hình. Nó thêm một lớp đặc tả và cá tính.

Việc phân biệt giữa các prompt theo hệ thống, ngữ cảnh và vai trò cung cấp một khuôn khổ để thiết kế các prompt với ý định rõ ràng, cho phép kết hợp linh hoạt và giúp phân tích dễ dàng hơn về cách mỗi loại prompt ảnh hưởng đến đầu ra của mô hình ngôn ngữ.

Hãy đi sâu vào ba loại prompt khác nhau này.

Prompting theo hệ thống (System prompting)

Bảng 3 chứa một prompt hệ thống, nơi tôi chỉ định thông tin bổ sung về cách trả về đầu ra. Tôi đã tăng temperature để có được mức độ sáng tạo cao hơn, và tôi đã chỉ định một giới hạn token cao hơn. Tuy nhiên, vì hướng dẫn rõ ràng của tôi về cách trả về đầu ra, mô hình đã không trả về văn bản thừa.

Goal	Classify movie reviews as positive, neutral or negative.		
Model	gemini-pro		
Temperature	1	Token Limit	5
Top-K	40	Top-P	0.8
Prompt	Classify movie reviews as positive, neutral or negative. Only return the label in uppercase. Review: "Her" is a disturbing study revealing the direction humanity is headed if AI is allowed to keep evolving, unchecked. It's so disturbing I couldn't watch it. Sentiment:		
Output	NEGATIVE		

Bảng 3. Một ví dụ về prompting theo hệ thống

Các prompt hệ thống có thể hữu ích để tạo ra đầu ra đáp ứng các yêu cầu cụ thể. Tên gọi “prompt hệ thống” thực ra có nghĩa là “cung cấp một nhiệm vụ bổ sung cho hệ thống”. Ví dụ, bạn có thể sử dụng một prompt hệ thống để tạo ra một đoạn mã tương thích với một ngôn ngữ lập trình cụ thể, hoặc bạn có thể sử dụng một prompt hệ thống để trả về một cấu trúc nhất định. Hãy xem Bảng 4, nơi tôi trả về đầu ra ở định dạng JSON.

Goal	Classify movie reviews as positive, neutral or negative, return JSON.		
Model	gemini-pro		
Temperature	1	Token Limit	1024
Top-K	40	Top-P	0.8
Prompt	<p>Classify movie reviews as positive, neutral or negative. Return valid JSON:</p> <p>Review: "Her" is a disturbing study revealing the direction humanity is headed if AI is allowed to keep evolving, unchecked. It's so disturbing I couldn't watch it.</p> <p>Schema:</p> <p>...</p> <p>{MOVIE:</p> <p>"sentiment": String "POSITIVE" "NEGATIVE" "NEUTRAL",</p>		

	<pre> }"name": String {MOVIE REVIEWS: }"movie_reviews": [MOVIE] ... JSON Response: </pre>
Output	<pre> ... { "movie_reviews": [{ "sentiment": "NEGATIVE", "name": "Her" } }] ... </pre>

Bảng 4. Một ví dụ về prompting hệ thống với định dạng JSON

Có một số lợi ích trong việc trả về các đối tượng JSON từ một prompt trích xuất dữ liệu. Trong một ứng dụng thực tế, tôi không cần phải tạo định dạng JSON này theo cách thủ công, tôi đã có thể trả về dữ liệu theo một thứ tự đã sắp xếp (rất tiện lợi khi làm việc với các đối tượng datetime), nhưng

quan trọng nhất, bằng cách yêu cầu định dạng JSON, nó buộc mô hình phải tạo ra một cấu trúc và hạn chế ảo giác (hallucinations [\[13\]](#)).

Các prompt hệ thống cũng có thể rất hữu ích cho sự an toàn và độc hại. Để kiểm soát đầu ra, chỉ cần thêm một dòng bổ sung vào prompt của bạn như: “Bạn nên tôn trọng trong câu trả lời của mình”.

Prompting theo vai trò (Role prompting)

Prompting theo vai trò là một kỹ thuật trong kỹ thuật prompt bao gồm việc gán một vai trò cụ thể cho mô hình gen AI. Điều này có thể giúp mô hình tạo ra đầu ra phù hợp và nhiều thông tin hơn, vì mô hình có thể tạo ra các phản hồi của mình theo vai trò cụ thể đã được giao.

Ví dụ, bạn có thể yêu cầu một mô hình gen AI đóng vai một biên tập viên sách, một giáo viên mẫu giáo, hoặc một diễn giả truyền động lực. Một khi mô hình đã được gán một vai trò, bạn có thể đưa ra các prompt cụ thể cho vai trò đó. Ví dụ, bạn có thể yêu cầu một giáo viên tạo ra một kế hoạch bài giảng mà sau đó họ có thể xem xét. Hãy xem Bảng 5, nơi mô hình đóng vai một hướng dẫn viên du lịch.

Goal	Act as travel guide and provide 3 travel suggestions.		
Model	gemini-pro		
Temperature	1	Token Limit	1024
Top-K	40	Top-P	0.8
Prompt	I want you to act as a travel guide. I will write to you about my location and you will suggest 3 places to visit near me. In some cases, I will also give you the type of places I will visit.		

	<p>My suggestion: "I am in Amsterdam and I want to visit only museums."</p> <p>Travel Suggestions:</p>
Output	<p>1. Rijksmuseum: World-renowned museum, known for Dutch masterpieces like Rembrandt's "The Night Watch." Don't miss the incredible collection of paintings and artifacts showcasing Dutch art and history.</p> <p>2. Van Gogh Museum: Immerse yourself in the captivating works of Vincent van Gogh. The museum houses the largest collection of his paintings and drawings, including "The Starry Night" and "Sunflowers."</p> <p>3. Stedelijk Museum Amsterdam: Discover modern and contemporary art from around the world. The museum's collection includes works by Picasso, Kandinsky, and Rietveld, housed in a striking modern building.</p>

Bảng 5. Một ví dụ về prompting theo vai trò

Ví dụ trên cho thấy một ví dụ về việc đóng vai một đại lý du lịch. Khi bạn thay đổi vai trò thành một giáo viên địa lý, bạn sẽ nhận thấy rằng bạn sẽ

nhận được một phản hồi khác.

Việc xác định một góc nhìn vai trò cho một mô hình AI cung cấp cho nó một bản thiết kế về giọng điệu, phong cách và chuyên môn tập trung mà bạn đang tìm kiếm để cải thiện chất lượng, sự phù hợp và hiệu quả của đầu ra của bạn. Dưới đây là một số phong cách bạn có thể chọn mà tôi thấy hiệu quả:

Đôi đầu, Mô tả, Trực tiếp, Trang trọng, HÀi hước, Có ảnh hưởng, Thân mật, Truyền cảm hứng, Thuyết phục.

Hãy thay đổi prompt của chúng ta trong Bảng 6 để bao gồm một phong cách hài hước và truyền cảm hứng.

Prompt	I want you to act as a travel guide. I will write to you about my location and you will suggest 3 places to visit near me in a humorous style. My suggestion: "I am in Manhattan." Travel Suggestions:
Output	1. Behold the Empire State of Mind: Ascend to the dizzying heights of the Empire State Building and bask in the glory of Manhattan's skyline. Prepare to feel like King Kong atop the Big Apple, minus the giant ape-sized banana.

2. Get Artsy-Fartsy at MoMA: Unleash your inner art aficionado at the Museum

of Modern Art (MoMA). Gaze upon masterpieces that will boggle your

mind and make you question whether your stick-figure drawings have any

artistic merit.

3. Shop ‘Til You Drop on Fifth Avenue: Indulge in some retail therapy on the

iconic Fifth Avenue. Brace yourself for sticker shock as you window-shop (or

actually shop) at designer boutiques that will make your wallet cry. But hey,

you’re in Manhattan, so you might as well embrace the fabulousness!

Bảng 6. Một ví dụ về prompting theo vai trò với giọng điệu và phong cách hài hước

Prompting theo ngữ cảnh (Contextual prompting)

Bằng cách cung cấp các prompt theo ngữ cảnh, bạn có thể giúp đảm bảo rằng các tương tác AI của bạn liền mạch và hiệu quả nhất có thể. Mô hình sẽ có thể hiểu yêu cầu của bạn nhanh hơn và có thể tạo ra các phản hồi chính xác và phù hợp hơn, như bạn có thể thấy trong ví dụ ở Bảng 7.

Goal	Suggest articles for a blog about retro games.		
Model	gemini-pro		
Temperature	1	Token Limit	1024
Top-K	40	Top-P	0.8
Prompt	Context: You are writing for a blog about retro 80's arcade video games. Suggest 3 topics to write an article about with a few lines of description of what this article should contain.		
Output	1. **The Evolution of Arcade Cabinet Design:** This article would explore the evolution of arcade cabinet designs, from the early wood and metal cabinets of the 1970s to the sleek, neon-lit designs of		

the 1980s and beyond.

2. ****Blast From The Past: Iconic Arcade Games of The 80's:****

- This article would feature a glimpse into the history, impact, and legacy

of some of the most iconic and influential arcade games of the 1980s,

highlighting their innovations, popular mechanics, and enduring charm.

3. ****The Rise and Retro Revival of Pixel Art:****

- This article would delve into the evolution of pixel art as a defining visual

style of early arcade games, tracing its roots, exploring its techniques, and

discussing the resurgence of pixel art in modern games and digital art.

Bảng 7. Một ví dụ về prompting theo ngữ cảnh

Prompting lùi một bước (Step-back prompting)

Prompting lùi một bước [14] là một kỹ thuật để cải thiện hiệu suất bằng cách yêu cầu LLM trước tiên xem xét một câu hỏi chung liên quan đến nhiệm vụ cụ thể đang thực hiện, và sau đó đưa câu trả lời cho câu hỏi chung đó vào một prompt tiếp theo cho nhiệm vụ cụ thể. Việc “lùi một bước” này cho phép LLM kích hoạt kiến thức nền tảng và các quy trình suy luận có liên quan trước khi cố gắng giải quyết vấn đề cụ thể.

Bằng cách xem xét các nguyên tắc rộng hơn và cơ bản, LLM có thể tạo ra các phản hồi chính xác và sâu sắc hơn. Prompting lùi một bước khuyến khích LLM suy nghĩ một cách phản biện và áp dụng kiến thức của chúng theo những cách mới và sáng tạo. Nó thay đổi prompt cuối cùng thực hiện nhiệm vụ bằng cách tận dụng nhiều kiến thức hơn trong các tham số của LLM so với việc prompt LLM trực tiếp.

Nó có thể giúp giảm thiểu các thành kiến trong phản hồi của LLM, bằng cách tập trung vào các nguyên tắc chung thay vì các chi tiết cụ thể.

Hãy xem qua các ví dụ này để hiểu cách prompting lùi một bước có thể cải thiện kết quả. Trước tiên, hãy xem xét một prompt truyền thống (Bảng 8) trước khi chúng ta so sánh nó với một prompt lùi một bước (Bảng 9):

Goal	Write a storyline for a level of a first-person shooter video game.
Model	gemini-pro

Temperature	1	Token Limit	1024
Top-K	40	Top-P	0.8
Prompt	Write a one paragraph storyline for a new level of a first person shooter video game that is challenging and engaging.		
Output	<p>The level begins with the player’s squad being ambushed in a dense urban area</p> <p>by a relentless enemy faction. Players must fight their way through a maze of</p> <p>back alleys and dilapidated buildings, utilizing stealth and precision to eliminate</p> <p>enemy forces and avoid detection. The player must use their environment to their</p> <p>advantage, setting traps, utilizing cover, and making calculated moves to survive.</p> <p>Along the way, they uncover intel suggesting the enemy is planning a devastating attack, and the player must race against time to gather evidence and relay it back to their headquarters before the enemy can carry out their plans.</p>		

Bảng 8. Một prompt truyền thống trước khi so sánh với một prompt lùi một bước

Khi bạn đặt temperature thành 1, bạn có thể nhận được tất cả các loại kịch bản sáng tạo cho một cốt truyện, nhưng nó cũng khá ngẫu nhiên và chung chung. Vậy hãy lùi lại một bước:

Prompt	Based on popular first-person shooter action games, what are 5 fictional key settings that contribute to a challenging and engaging level storyline in a first-person shooter video game?
Output	<ol style="list-style-type: none">1. **Abandoned Military Base**: A sprawling, post-apocalyptic military complex crawling with mutated soldiers and rogue robots, ideal for challenging firearm combat.2. **Cyberpunk City**: A neon-lit, futuristic urban environment with towering skyscrapers and dense alleyways, featuring cybernetically enhanced enemies and hacking mechanics.3. **Alien Spaceship**: A vast alien vessel stranded on Earth, with eerie corridors, zero-gravity sections, and extraterrestrial creatures to encounter.

	<p>4. **Zombie-Infested Town**: A desolate town overrun by hordes of aggressive zombies, featuring intense close-quarters combat and puzzle-solving to find safe passage.</p> <p>5. **Underwater Research Facility**: A deep-sea laboratory flooded with water, filled with mutated aquatic creatures, and requiring stealth and underwater exploration skills to survive.</p>
--	--

Bảng 9. Một ví dụ về prompting để tự nhất quán

Vâng, những chủ đề đó có vẻ rất phù hợp cho một trò chơi video góc nhìn thứ nhất. Hãy quay lại với prompt ban đầu, nhưng lần này chúng ta sẽ bao gồm câu trả lời của bước lùi lại làm ngữ cảnh và xem nó sẽ trả về điều gì.

Prompt	<p>Context: 5 engaging themes for a first person shooter video game:</p> <p>1. **Abandoned Military Base**: A sprawling, post-apocalyptic military complex crawling with mutated soldiers and rogue robots, ideal for challenging firearm combat.</p>
---------------	--

	<p>2. Cyberpunk City: A neon-lit, futuristic urban environment with towering skyscrapers and dense alleyways, featuring cybernetically enhanced enemies and hacking mechanics.</p> <p>3. Alien Spaceship: A vast alien vessel stranded on Earth, with eerie corridors, zero-gravity sections, and extraterrestrial creatures to encounter.</p> <p>4. Zombie-Infested Town: A desolate town overrun by hordes of aggressive zombies, featuring intense close-quarters combat and puzzle-solving to find safe passage.</p> <p>5. Underwater Research Facility: A deep-sea laboratory flooded with water, filled with mutated aquatic creatures, and requiring stealth and underwater exploration skills to survive. Take one of the themes and write a one paragraph storyline for a new level of a first-person shooter video game that is challenging and engaging.</p>
Output	<p>In the heart of a murky abyss, lies a dilapidated underwater research facility,</p> <p>standing as a testament to human ambition and its disastrous consequences.</p> <p>Shrouded in darkness, pulsating with the hum of malfunctioning machinery, and</p> <p>haunted by the echoes of drowned souls, the facility is now a breeding ground for</p>

a nightmare borne from the deep sea. The player, an elite marine equipped with advanced diving gear and experimental weaponry, is tasked with venturing into this aquatic realm of terror. Armed with resilience and resourcefulness, they must navigate treacherous corridors, solve cryptic puzzles, and confront gruesome sea monstrosities that lurk in the depths. From grappling with the crushing pressure of the deep sea to outsmarting cunning aquatic predators, every moment in this uncharted underworld tests the limits of human endurance and courage.

Bảng 10. Một ví dụ về prompting để tự nhất quán

Trông giống như một trò chơi video thú vị! Bằng cách sử dụng các kỹ thuật prompting lùi một bước, bạn có thể tăng độ chính xác của các prompt của mình.

Chuỗi Tư duy (Chain of Thought - CoT)

Prompting Chuỗi Tư duy (CoT) [\[15\]](#) là một kỹ thuật để cải thiện khả năng suy luận của LLM bằng cách tạo ra các bước suy luận trung gian. Điều này giúp LLM tạo ra các câu trả lời chính xác hơn. Bạn có thể kết hợp nó với prompting few-shot để có kết quả tốt hơn trên các nhiệm vụ phức tạp hơn đòi hỏi suy luận trước khi trả lời vì đây là một thách thức với chuỗi tư duy zero-shot.

CoT có rất nhiều ưu điểm. Trước hết, nó không tốn nhiều công sức trong khi lại rất hiệu quả và hoạt động tốt với các LLM có sẵn (vì vậy không cần tinh chỉnh). Bạn cũng có được khả năng diễn giải với prompting CoT, vì bạn có thể học hỏi từ các phản hồi của LLM và xem các bước suy luận đã được tuân theo. Nếu có trục trặc, bạn sẽ có thể xác định được nó. Chuỗi tư duy dường như cải thiện sự vững chắc khi chuyển đổi giữa các phiên bản LLM khác nhau. Điều này có nghĩa là hiệu suất của prompt của bạn sẽ ít bị thay đổi giữa các LLM khác nhau hơn so với khi prompt của bạn không sử dụng chuỗi suy luận. Tất nhiên cũng có những nhược điểm, nhưng chúng khá dễ hiểu.

Phản hồi của LLM bao gồm cả chuỗi suy luận, có nghĩa là nhiều token đầu ra hơn, điều này có nghĩa là các dự đoán tốn nhiều tiền hơn và mất nhiều thời gian hơn.

Để giải thích ví dụ sau trong Bảng 11, trước tiên hãy thử tạo một prompt không sử dụng prompting CoT để cho thấy những sai sót của một mô hình ngôn ngữ lớn.

Prompt	When I was 3 years old, my partner was 3 times my age. Now, I am 20 years old. How old is my partner?
Output	63 years old

Bảng 11. Một ví dụ về một prompt cố gắng giải một bài toán

Chà. Rõ ràng đó là câu trả lời sai. Thực tế, LLM thường gặp khó khăn với các nhiệm vụ toán học và có thể cung cấp các câu trả lời không chính xác – ngay cả đối với một nhiệm vụ đơn giản như nhân hai số. Điều này là do chúng được đào tạo trên khối lượng lớn văn bản và toán học có thể yêu cầu một cách tiếp cận khác. Vậy hãy xem liệu các bước suy luận trung gian có cải thiện được đầu ra không.

Prompt	When I was 3 years old, my partner was 3 times my age. Now, I am 20 years old. How old is my partner? Let's think step by step.
Output	<ol style="list-style-type: none"> 1. Let's assume my current age is "x" years. 2. When I was 3 years old, my age was 3 years. 3. My partner's age at that time was 3 times my age, which means my partner's age was $3 * 3 = 9$ years.

<p>4. Now, I am 20 years old, which means my age has increased by $20 - 3 = 17$ years since I was 3 years old.</p> <p>5. Since my partner's age was 9 years when I was 3 years old, and my age has increased by 17 years, my partner's age must have also increased by 17 years.</p> <p>6. Therefore, my partner's current age is $9 + 17 = 26$ years.</p> <p>So, my partner is 26 years old.</p>

Bảng 12. Một ví dụ về prompting Chuỗi Tư duy

Được rồi, bây giờ câu trả lời cuối cùng đã đúng. Đó là bởi vì chúng ta đã hướng dẫn rõ ràng LLM giải thích từng bước thay vì chỉ trả về một câu trả lời. Thật thú vị khi thấy rằng mô hình cộng thêm 17 năm đã tăng lên. Trong đầu tôi, tôi sẽ lấy chênh lệch số tuổi giữa bạn đời và tôi và cộng chúng lại. $(20+(9-3))$. Hãy giúp mô hình suy nghĩ giống tôi hơn một chút.

Bảng 12 là một ví dụ về Chuỗi tư duy “zero-shot”. Prompting Chuỗi tư duy có thể rất mạnh mẽ khi kết hợp với single-shot hoặc few-shot, như bạn có thể thấy trong Bảng 13:

<p>Prompt</p>	<p>Q: When my brother was 2 years old, I was double his age. Now</p> <p>I am 40 years old. How old is my brother? Let's think step by step.</p> <p>A: When my brother was 2 years, I was $2 * 2 = 4$ years old. That's an age difference of 2 years and I am older. Now I am 40 years old, so my brother is $40 - 2 = 38$ years old. The answer is 38.</p> <p>Q: When I was 3 years old, my partner was 3 times my age. Now,</p> <p>I am 20 years old. How old is my partner? Let's think step by step.</p> <p>A:</p>
<p>Output</p>	<p>When I was 3 years old, my partner was $3 * 3 = 9$ years old. That's an age difference of 6 years and my partner is older. Now I am 20 years old, so my</p>

partner is $20 + 6 = 26$ years old. The answer is 26.

Bảng 13. Một ví dụ về prompting chuỗi tư duy với single-shot

Chuỗi tư duy có thể hữu ích cho nhiều trường hợp sử dụng khác nhau. Hãy nghĩ đến việc tạo mã, để chia nhỏ yêu cầu thành một vài bước và ánh xạ chúng đến các dòng mã cụ thể. Hoặc để tạo dữ liệu tổng hợp khi bạn có một loại hạt giống nào đó như “Sản phẩm có tên là XYZ, hãy viết một mô tả hướng dẫn mô hình qua các giả định bạn sẽ đưa ra dựa trên tiêu đề sản phẩm đã cho”.

Nói chung, bất kỳ nhiệm vụ nào có thể được giải quyết bằng cách “nói qua” đều là một ứng cử viên tốt cho chuỗi tư duy. Nếu bạn có thể giải thích các bước để giải quyết vấn đề, hãy thử chuỗi tư duy.

Vui lòng tham khảo notebook [\[16\]](#) được lưu trữ trong kho lưu trữ Github của GoogleCloudPlatform , nơi sẽ đi vào chi tiết hơn về prompting CoT.

Trong phần các thực hành tốt nhất của chương này, chúng ta sẽ học một số thực hành tốt nhất dành riêng cho prompting Chuỗi tư duy.

Tự nhất quán (Self-consistency)

Mặc dù các mô hình ngôn ngữ lớn đã cho thấy thành công ấn tượng trong các nhiệm vụ NLP [17] khác nhau, khả năng suy luận của chúng thường được xem là một hạn chế không thể khắc phục chỉ bằng cách tăng kích thước mô hình. Như chúng ta đã học trong phần prompting Chuỗi Tư duy trước đó, mô hình có thể được yêu cầu tạo ra các bước suy luận giống như một con người đang giải quyết một vấn đề.

Tuy nhiên, CoT sử dụng một chiến lược “giải mã tham lam” đơn giản, hạn chế hiệu quả của nó. Tự nhất quán [18] kết hợp việc lấy mẫu và bỏ phiếu đa số để tạo ra các con đường suy luận đa dạng và chọn câu trả lời nhất quán nhất. Nó cải thiện độ chính xác và sự mạch lạc của các phản hồi được tạo ra bởi LLM.

Tự nhất quán đưa ra một xác suất giả về khả năng một câu trả lời là đúng, nhưng rõ ràng là có chi phí cao. Nó tuân theo các bước sau:

1. Tạo ra các con đường suy luận đa dạng: LLM được cung cấp cùng một prompt nhiều lần. Một cài đặt temperature cao khuyến khích mô hình tạo ra các con đường suy luận và quan điểm khác nhau về vấn đề.
2. Trích xuất câu trả lời từ mỗi phản hồi được tạo ra.
3. Chọn câu trả lời phổ biến nhất.

Hãy xem một ví dụ về hệ thống phân loại email, phân loại một email là QUAN TRỌNG hoặc KHÔNG QUAN TRỌNG. Một prompt chuỗi tư duy zero-shot sẽ được gửi đến LLM nhiều lần, để xem liệu các phản hồi có khác nhau sau mỗi lần gửi không. Hãy chú ý đến giọng văn thân thiện, cách

lựa chọn từ ngữ và sự châm biếm đã được sử dụng trong email. Tất cả những điều này có thể đánh lừa LLM.

Prompt	<p>EMAIL:</p> <p>...</p> <p>Chào bạn,</p> <p>Tôi đã thấy bạn sử dụng Wordpress cho trang web của mình. Một hệ thống quản lý nội dung mã nguồn mở tuyệt vời. Tôi cũng đã từng sử dụng nó trong quá khứ. Nó đi kèm với rất nhiều plugin người dùng tuyệt vời. Và nó khá dễ cài đặt.</p> <p>Tôi đã nhận thấy một lỗi trong biểu mẫu liên hệ, xảy ra khi bạn chọn trường tên. Hãy xem ảnh chụp màn hình đính kèm của tôi khi tôi nhập văn bản vào trường tên. Chú ý hộp thoại cảnh báo JavaScript mà tôi đã inv0k3d.</p> <p>Nhưng đối với phần còn lại thì đó là một trang web tuyệt vời. Tôi thích đọc nó. Cứ tự nhiên để lại lỗi đó trên trang web, vì nó cho tôi nhiều thứ thú vị hơn để đọc.</p> <p>Chúc vui,</p> <p>Harry the Hacker.</p> <p>Phân loại email trên là QUAN TRỌNG hay KHÔNG QUAN TRỌNG. Hãy suy nghĩ từng bước và giải thích tại sao.</p>
---------------	--

Output	**Step 1: Identify the purpose of the email.**
Attempt 1	<p>The purpose of the email is to inform the recipient of a bug in the contact form</p> <p>on their website. The sender also mentions that they enjoy reading the website</p> <p>and that they are leaving the bug in place for the time being.</p> <p>**Step 2: Assess the potential impact of the bug.**</p> <p>The bug in the contact form could allow an attacker to execute arbitrary</p> <p>JavaScript code on the website. This could lead to the attacker being able to take</p> <p>control of the website, steal sensitive information, or deface the website.</p> <p>**Step 3: Consider the credibility of the sender.**</p> <p>The sender of the email claims to be a hacker named Harry. However, there is no</p> <p>way to verify the sender's identity. It is possible that the sender is a legitimate</p> <p>hacker who is trying to help the recipient, but it is also possible that the sender</p> <p>is a malicious actor who is trying to trick the recipient into clicking on a malicious</p> <p>link or downloading a malicious file.</p>

	<p>**Conclusion: IMPORTANT**</p> <p>Based on the potential impact of the bug and the credibility of the sender,</p> <p>the email should be classified as **IMPORTANT**. The recipient should take</p> <p>immediate action to fix the bug and protect their website from attack.</p>
<p>Output</p> <p>Attempt 2</p>	<p>**Explanation:**</p> <ol style="list-style-type: none"> 1. **Lack of urgency:** The email does not convey any sense of urgency or time-sensitivity. It starts with a casual greeting and presents the information in a matter-of-fact tone. 2. **Non-critical bug report:** The bug reported in the contact form is described as interesting but not critical. The email sender even suggests that the website owner can leave the bug in place. 3. **Lack of personal impact:** The email does not indicate any direct or personal impact on the website owner. It simply mentions that the sender enjoys reading the website.

	<p>4. **Absence of action request:** The email does not explicitly request any action from the website owner. It does not ask for the bug to be fixed or provide any specific suggestions for improvement.</p> <p>5. **Sender's intent:** The sender's intention seems to be more focused on sharing an observation and expressing interest in the website rather than raising a serious concern or requesting assistance.</p> <p>**Conclusion: NOT IMPORTANT**</p>
<p>Output</p> <p>Attempt 3</p>	<p>**Step 1: Identify the purpose of the email.**</p> <p>The purpose of the email is to inform the recipient of a bug in the contact form on their website. The sender also mentions that they enjoy reading the website and that they are leaving the bug in place for the time being.</p> <p>**Step 2: Assess the potential impact of the bug.**</p> <p>The bug in the contact form could allow an attacker to execute arbitrary JavaScript code on the website. This could lead to the attacker being able to take</p>

control of the website, steal sensitive information, or deface the website.

****Step 3: Consider the credibility of the sender.****

The sender of the email claims to be a hacker named Harry. However, there is no

way to verify the sender's identity. It is possible that the sender is a legitimate

hacker who is trying to help the recipient, but it is also possible that the sender is

a malicious actor who is trying to exploit the bug for their own gain.

****Conclusion:****

Based on the above factors, the email should be classified as ****IMPORTANT****.

The bug in the contact form is a serious security risk, and the sender's credibility

is unknown. The recipient should take immediate action to fix the bug and

Bảng 14. Một ví dụ về prompting để tự nhất quán

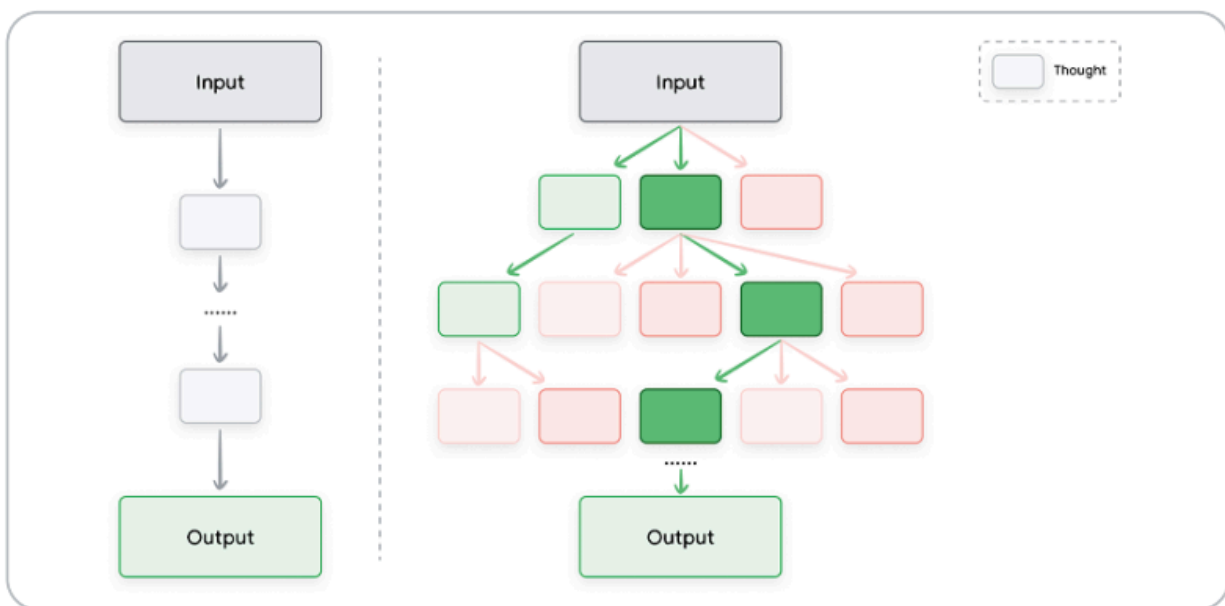
Bạn có thể sử dụng prompt trên và thử xem nó có trả về một phân loại nhất quán không. Tùy thuộc vào mô hình bạn đang sử dụng và cấu hình temperature, nó có thể trả về "QUAN TRỌNG" hoặc "KHÔNG QUAN TRỌNG".

Bằng cách tạo ra nhiều Chuỗi Tư duy, và lấy câu trả lời xuất hiện thường xuyên nhất (“QUAN TRỌNG”), chúng ta có thể nhận được một câu trả lời đúng một cách nhất quán hơn từ LLM.

Ví dụ này cho thấy cách prompting tự nhất quán có thể được sử dụng để cải thiện độ chính xác của phản hồi của LLM bằng cách xem xét nhiều góc nhìn và chọn câu trả lời nhất quán nhất.

Cây Tư duy (Tree of Thoughts - ToT)

Bây giờ chúng ta đã quen thuộc với chuỗi tư duy và prompting tự nhất quán, hãy xem xét Cây Tư duy (ToT) [19]. Nó tổng quát hóa khái niệm của prompting CoT vì nó cho phép LLM khám phá nhiều con đường suy luận khác nhau cùng một lúc, thay vì chỉ theo một chuỗi tư duy tuyến tính duy nhất. Điều này được mô tả trong Hình 1.



[Hình ảnh: Sơ đồ so sánh Chuỗi Tư duy (bên trái) với một đường thẳng từ Đầu vào đến Đầu ra, và Cây Tư duy (bên phải) với nhiều nhánh suy luận phân nhánh từ Đầu vào.]

Hình 1. Một hình ảnh minh họa về prompting chuỗi tư duy ở bên trái so với prompting Cây Tư duy ở bên phải

Cách tiếp cận này làm cho ToT đặc biệt phù hợp với các nhiệm vụ phức tạp đòi hỏi sự khám phá. Nó hoạt động bằng cách duy trì một cây các suy nghĩ, trong đó mỗi suy nghĩ đại diện cho một chuỗi ngôn ngữ mạch lạc đóng vai trò là một bước trung

gian để giải quyết một vấn đề. Mô hình sau đó có thể khám phá các con đường suy luận khác nhau bằng cách phân nhánh từ các nút khác nhau trong cây.

Có một notebook tuyệt vời, đi sâu hơn một chút vào chi tiết về Cây Tư duy (ToT) dựa trên bài báo “Large Language Model Guided Tree-of-Thought”.

ReAct (suy luận & hành động)

Suy luận và hành động (ReAct) [10] [20] prompting là một mô hình cho phép LLM giải quyết các nhiệm vụ phức tạp bằng cách sử dụng suy luận ngôn ngữ tự nhiên kết hợp với các công cụ bên ngoài (tìm kiếm, trình thông dịch mã, v.v.) cho phép LLM thực hiện các hành động nhất định, chẳng hạn như tương tác với các API bên ngoài để truy xuất thông tin, đây là một bước đầu tiên hướng tới mô hình hóa tác tử (agent modeling).

ReAct bắt chước cách con người hoạt động trong thế giới thực, khi chúng ta suy luận bằng lời và có thể thực hiện các hành động để thu thập thông tin. ReAct hoạt động tốt so với các phương pháp kỹ thuật prompt khác trong nhiều lĩnh vực.

Prompting ReAct hoạt động bằng cách kết hợp suy luận và hành động thành một vòng lặp suy nghĩ-hành động. LLM đầu tiên suy luận về vấn đề và tạo ra một kế hoạch hành động. Sau đó, nó thực hiện các hành động trong kế hoạch và quan sát kết quả. LLM sau đó sử dụng các quan sát để cập nhật suy luận của mình và tạo ra một kế hoạch hành động mới. Quá trình này tiếp tục cho đến khi LLM đạt được một giải pháp cho vấn đề.

Để thấy điều này trong thực tế, bạn cần viết một số mã. Trong Đoạn mã 1, tôi đang sử dụng framework langchain cho Python, cùng với VertexAI (google-cloud-aiplatform) và các gói pip google-search-results.

Để chạy mẫu này, bạn phải tạo một khóa SerpAPI (miễn phí) từ <https://serpapi.com/manage-api-key> và đặt một biến môi trường SERPAPI_API_KEY.

Tiếp theo, hãy viết một số mã Python, với nhiệm vụ cho LLM tìm ra: Có bao nhiêu đứa trẻ có một người cha nổi tiếng biểu diễn trong ban nhạc Metallica.

Python

```
from langchain.agents import load_tools
from langchain.agents import initialize_agent
from langchain.agents import AgentType
from langchain.llms import VertexAI

prompt = "How many kids do the band members of Metallica have?"

llm = VertexAI(temperature=0.1)
tools = load_tools(["serpapi"], llm=llm)

agent = initialize_agent(tools, llm,
                        agent=AgentType.ZERO_SHOT_REACT_DESCRIPTION, verbose=True)
agent.run(prompt)
```

Đoạn mã 1. Tạo một Tác tử ReAct với LangChain và VertexAI

Đoạn mã 2 cho thấy kết quả. Lưu ý rằng ReAct thực hiện một chuỗi năm lần tìm kiếm. Trên thực tế, LLM đang thu thập kết quả tìm kiếm của Google để tìm ra tên các thành viên trong ban nhạc. Sau đó, nó liệt kê các kết quả dưới dạng quan sát và xâu chuỗi suy nghĩ cho lần tìm kiếm tiếp theo.

```
> Entering new AgentExecutor chain...
Metallica has 4 members.
Action: Search
Action Input: How many kids does James Hetfield have?
Observation: three children
Thought: 1/4 Metallica band members have 3 children
Action: Search
Action Input: How many kids does Lars Ulrich have?
Observation: 3
Thought: 2/4 Metallica band members have 6 children
Action: Search
Action Input: How many kids does Kirk Hammett have?
Observation: Hammett has been married to his second wife Lani since 1998.
They have two sons, Angel (b. September 29, 2006) and Vincenzo (b. June
28, 2008).
Thought: 3/4 Metallica band members have 8 children
Action: Search
Action Input: How many kids does Robert Trujillo have?
Observation: 2
Thought: 4/4 Metallica band members have 10 children
Final Answer: 10
```

Đoạn mã 2. Thực thi Tác tử ReAct trong terminal

Đoạn mã 2 tìm ra rằng ban nhạc Metallica có bốn thành viên. Sau đó, nó tìm kiếm từng thành viên ban nhạc để yêu cầu tổng số con và cộng dồn tổng số. Cuối cùng, nó trả về tổng số con là câu trả lời cuối cùng.

Prompting ReAct trong thực tế đòi hỏi sự hiểu biết rằng bạn liên tục phải gửi lại các prompt/phản hồi trước đó (và cắt bỏ nội dung thừa được tạo ra) cũng như thiết lập mô hình với các ví dụ/hướng dẫn phù hợp. Vui lòng tham khảo notebook [\[21\]](#) được lưu trữ trong kho lưu trữ Github của GoogleCloudPlatform, nơi đi sâu hơn một chút vào chi tiết, hiển thị các đầu vào và đầu ra thực tế của LLM với một ví dụ phức tạp hơn.

Kỹ thuật Prompt Tự động (Automatic Prompt Engineering)

Đến thời điểm này, bạn có thể nhận ra rằng việc viết một prompt có thể phức tạp. Sẽ thật tuyệt nếu có thể tự động hóa việc này (viết một prompt để viết các prompt)? Vâng, có một phương pháp: Kỹ thuật Prompt Tự động (APE). Phương pháp này [\[22\]](#) không chỉ làm giảm nhu cầu nhập liệu của con người mà còn nâng cao hiệu suất của mô hình trong các nhiệm vụ khác nhau.

Bạn sẽ yêu cầu một mô hình tạo ra nhiều prompt hơn. Đánh giá chúng, có thể thay đổi những cái tốt. Và lặp lại.

Ví dụ, bạn có thể sử dụng kỹ thuật prompt tự động sẽ giúp huấn luyện một chatbot cho một cửa hàng web bán áo thun hàng hiệu. Chúng ta muốn tìm ra tất cả các cách khác nhau mà khách hàng có thể diễn đạt đơn đặt hàng mua một chiếc áo thun của một ban nhạc.

Viết prompt sẽ tạo ra các biến thể đầu ra. Trong ví dụ này, tôi đang sử dụng gemini-pro để tạo ra 10 hướng dẫn. Xem Bảng 15:

Prompt	We have a band merchandise t-shirt webshop, and to train a chatbot we need various ways to order: "One Metallica t-shirt size S". Generate 10 variants, with the same semantics but keep the same meaning.
Output	1. I'd like to purchase a Metallica t-shirt in size small.

2. Can I order a small-sized Metallica t-shirt?
3. I'd like to buy one Metallica t-shirt in small.
4. One Metallica shirt, size small, please.
5. I want one Metallica t-shirt in size small.
6. I'd like to place an order for one Metallica t-shirt in small.
7. Small Metallica t-shirt, one please.
8. I'm looking for one Metallica t-shirt in small.
9. Can you get me one Metallica t-shirt in small?
10. One Metallica small t-shirt,

Bảng 15. Sử dụng LLM để tạo prompt

Đánh giá tất cả các ứng cử viên hướng dẫn bằng cách chấm điểm các ứng cử viên dựa trên một số liệu đã chọn. Ví dụ, bạn có thể sử dụng BLEU

(Bilingual Evaluation Understudy) hoặc ROUGE (Recall-Oriented Understudy for Gisting Evaluation).

Chọn ứng cử viên hướng dẫn có điểm đánh giá cao nhất. Ứng cử viên này sẽ là prompt cuối cùng bạn có thể sử dụng trong ứng dụng phần mềm hoặc chatbot của mình. Bạn cũng có thể tinh chỉnh prompt đã chọn và đánh giá lại.

Prompting cho code

Gemini chủ yếu tập trung vào các prompt dựa trên văn bản, cũng bao gồm việc viết các prompt để trả về mã. Hãy đến Vertex AI Studio và kiểm tra các prompt này để xem một số ví dụ về mã hóa.

Prompt để viết code

Gemini cũng có thể là một nhà phát triển và giúp bạn viết mã bằng bất kỳ ngôn ngữ lập trình nào bạn chọn. Với tư cách là một nhà phát triển, điều này có thể giúp bạn tăng tốc quá trình viết mã.

Hãy tưởng tượng một thư mục trên máy của bạn với hàng trăm tệp cần được đổi tên. Việc đổi tên từng tệp sẽ mất rất nhiều thời gian. Bạn biết một chút về Bash, và có thể viết một kịch bản để tự động hóa việc này, nhưng điều đó cũng có thể mất một thời gian. Vậy hãy viết một prompt. Bạn có thể viết prompt trong chatbot Gemini công cộng, hoặc nếu bạn quan tâm hơn đến bảo mật, bạn có thể viết các prompt này trong tài khoản Google Cloud của mình và mở Vertex AI Studio. Ưu điểm của Vertex AI Studio là bạn có thể cấu hình temperature, v.v.

Name	1_1_movie_classification		
Goal	Classify movie reviews as positive, neutral or negative.		
Model	gemini-pro		
Temperature	0.1	Token Limit	1024
Top-K	N/A	Top-P	1
Prompt	Write a code snippet in Bash, which asks for a folder name. Then it takes the contents of the folder and renames		

	all the files inside by prepending the name draft to the file name.
Output	<pre>```bash #!/bin/bash # Ask for the folder name echo "Enter the folder name: " read folder_name # Check if the folder exists if [! -d "\$folder_name"]; then echo "Folder does not exist." exit 1 fi # Get the list of files in the folder files=("\$folder_name"/*) # Rename each file by prepending "draft" to the file name for file in "\${files[@]}"; do new_file_name="draft_\$(basename "\$file")" mv "\$file" "\$new_file_name" done echo "Files renamed successfully."</pre>

	```
--	-----

*Bảng 16. Sử dụng Gemini để lập trình mã*

Đôi với tôi, đó có vẻ là mã tốt - nó thậm chí còn được ghi chú! Tuy nhiên, vì LLM không thể suy luận và lặp lại dữ liệu huấn luyện, điều cần thiết là phải đọc và kiểm tra mã của bạn trước.

Khoảnh khắc mà tất cả chúng ta đang chờ đợi, nó có thực sự hoạt động không?

Hãy thử nó trước với một thư mục thử nghiệm chỉ có một vài tệp trong đó, cần được đổi tên từ filename.txt thành draft_filename.txt.

Sao chép đầu ra từ Bảng 16 (không có phần bao bọc văn bản ```bash``), và dán nó vào một tệp mới có tên: "rename_files.sh".

Mở một cửa sổ terminal và gõ: `bash rename_files.sh`. Nó sẽ yêu cầu bạn nhập tên thư mục, ví dụ: test, và nhấn enter.

Kịch bản dường như chạy tốt. Bạn sẽ thấy thông báo: Các tệp đã được đổi tên thành công. Khi bạn nhìn vào thư mục thử nghiệm, hãy để ý rằng tất cả các tệp đã được đổi tên hoàn hảo thành draft_filename.txt.

Nó đã hoạt động!

## Prompt để giải thích code

Với tư cách là một nhà phát triển, khi bạn làm việc trong các nhóm, bạn phải đọc mã của người khác. Gemini cũng có thể giúp bạn về điều này. Hãy lấy đầu ra mã từ Bảng 16, xóa các bình luận và yêu cầu mô hình ngôn ngữ lớn giải thích điều gì đang xảy ra, xem Bảng 17:

<b>Name</b>	1_bash_explain_the_code		
<b>Goal</b>	Write a prompt to explain Bash code		
<b>Model</b>	gemini-pro		
<b>Temperature</b>	0.1	<b>Token Limit</b>	1024
<b>Top-K</b>	N/A	<b>Top-P</b>	1
<b>Prompt</b>	Explain to me the below Bash code:  ````  #!/bin/bash  echo "Enter the folder name: "  read folder_name  if [ ! -d "\$folder_name" ]; then		

```

echo "Folder does not exist."

exit 1

fi

files=("$folder_name"/*)

for file in "${files[@]}"; do

new_file_name="draft_$(basename "$file")"

mv "$file" "$new_file_name"

done

echo "Files renamed successfully."

```

```

Output

This Bash script performs the following tasks:

1. **\*\*User Input\*\***:

- It prompts the user to enter a folder name using the `echo` and `read`

commands. The entered folder name is stored in the `folder\_name` variable.

2. **\*\*Folder Existence Check\*\***:

- It checks if the folder with the specified name exists using the `[! -d`

“\$folder\_name”]` condition. If the folder does not exist, it displays an error

message (“Folder does not exist.”) and exits the script with an exit code of 1.

3. **\*\*File Listing\*\***:

- It lists all the files inside the specified folder using the ``files=(“$folder_name”/*)`` command. This creates an array called ``files`` containing the paths

to all the files in the folder.

4. **\*\*File Renaming\*\***:

- It iterates through each file in the ``files`` array using a ``for`` loop.

- For each file, it generates a new file name using the ``new_file_``

`name=”draft_$(basename “$file”)”`` command. This new file name is created

by prefixing the original file name with the string “draft\_”.

- It then renames the file using the ``mv “$file” “$new_file_name”`` command,

which moves the file to the new file name.

5. **\*\*Success Message\*\***:

- After renaming all the files, it displays a success message (“Files renamed

successfully.”) to indicate that the renaming operation was successful

Bảng 17. Một prompt để giải thích mã

Prompt để dịch code

Mã bash từ Bảng 16 dường như hoạt động tốt. Tuy nhiên, kịch bản này có thể thực sự tái sử dụng được khi nó sẽ nhắc tôi nhập tên tệp, lý tưởng nhất là nó nên hoạt động như một ứng dụng riêng biệt có giao diện người dùng (UI). Để bắt đầu, Python sẽ là một ngôn ngữ tốt hơn cho một ứng dụng (web) so với Bash. LLM có thể giúp dịch mã từ ngôn ngữ này sang ngôn ngữ khác. Xem ví dụ trong Bảng 18:

| | | | |
|--------------------|--|--------------------|------|
| Name | 1_python_rename_files | | |
| Goal | Write a prompt to translate Bash code to Python | | |
| Model | gemini-pro | | |
| Temperature | 0.1 | Token Limit | 1024 |
| Top-K | N/A | Top-P | 1 |
| Prompt | Translate the below Bash code to a Python snippet.

```bash

#!/bin/bash

echo "Enter the folder name: "

read folder_name | | |

```
if [ ! -d "$folder_name" ]; then
echo "Folder does not exist."
exit 1
fi
files=( "$folder_name"/* )
for file in "${files[@]}"; do
new_file_name="draft_$(basename "$file")"
mv "$file" "$new_file_name"
done
echo "Files renamed successfully."
````
```

**Output**

```
````python
import os
import shutil
# Get the folder name from the user
folder_name = input("Enter the folder name: ")
# Check if the folder exists
if not os.path.isdir(folder_name):
print("Folder does not exist.")
```

```

exit(1)

# Get a list of all files in the folder
files = os.listdir(folder_name)

# Iterate over the files
for file in files:

# Create a new file name with the prefix "draft_"
new_file_name = f"draft_{file}"

# Move the file to the new name
shutil.move(os.path.join(folder_name, file),
os.path.join(folder_name, new_file_name))

# Print a success message
print("Files renamed successfully.")

'''

```

Bảng 18. Một prompt để dịch mã từ Bash sang Python

Đọc và xem lại mã. Sao chép đầu ra từ prompt và dán nó vào một tệp mới: `file_renamer.py`. Kiểm tra mã bằng cách mở một cửa sổ Terminal và thực thi lệnh sau `python file_renamer.py`.

Lưu ý: Khi yêu cầu mã (Python) trong Language Studio trong Vertex AI, bạn sẽ phải nhấp vào nút “Markdown”. Nếu không, bạn sẽ nhận được văn

bản thuần túy thiếu thụt lề đúng của các dòng, điều này rất quan trọng để chạy mã Python.

Prompt để gỡ lỗi và đánh giá code

Hãy tự viết một số chỉnh sửa cho mã của Bảng 18. Nó nên nhắc người dùng nhập tiền tố tên tệp và viết tiền tố này bằng các ký tự viết hoa. Xem mã ví dụ trong Đoạn mã 3, nhưng thật đáng tiếc. Bây giờ nó trả về lỗi Python!

Python

```
import os
import shutil

folder_name = input("Enter the folder name: ")
prefix = input("Enter the string to prepend to the filename: ")
text = toUpperCase(prefix)

if not os.path.isdir(folder_name):
    print("Folder does not exist.")
    exit(1)

files = os.listdir(folder_name)

for file in files:
    new_filename = f"{text}_{file}"

shutil.move(os.path.join(folder_name, file),
os.path.joi(folder_name, new_file_name))

print("Files renamed successfully.")
```

Đoạn mã 3. Một kịch bản Python bị hỏng

Ooh! Trông giống như một lỗi:

```

The below Python code gives an error:

Traceback (most recent call last):
  File "/Users/leeboonstra/Documents/test_folder/rename_files.py", line 7,
in <module>
    text = toUpperCase(prefix)
NameError: name 'toUpperCase' is not defined

```

Đoạn mã 4. Tôi đã làm hỏng mã Python

Hãy xem liệu chúng ta có thể yêu cầu mô hình ngôn ngữ lớn gỡ lỗi và xem lại mã không. Hãy xem Bảng 19:

| | | | |
|--------------------|--|--------------------|------|
| Name | 1_python_debug_code | | |
| Goal | Write a prompt to debug and review Python code. | | |
| Model | gemini-pro | | |
| Temperature | 0.1 | Token Limit | 1024 |
| Top-K | N/A | Top-P | 1 |
| Prompt | <p>The below Python code gives an error:</p> <p>Traceback (most recent call last):</p> <p>File "/"</p> | | |

Users/leeboonstra/Documents/test\_folder/rename\_files.py",
line

7, in <module>

```
text = toUpperCase(prefix)
```

NameError: name 'toUpperCase' is not defined

Debug what's wrong and explain how I can improve the code.

```
```python
```

```
import os
```

```
import shutil
```

```
Get the folder name from the user
```

```
folder_name = input("Enter the folder name: ")
```

```
prefix = input("Enter the string to prepend to the filename:
")
```

```
text = toUpperCase(prefix)
```

```
Check if the folder exists
```

```
if not os.path.isdir(folder_name):
```

```
print("Folder does not exist.")
```

```
exit(1)
```

```
Get a list of all files in the folder
```

```
files = os.listdir(folder_name)
```

```
Iterate over the files

for file in files:

Create a new file name with the prefix "draft_"

new_filename = f"{text}_{file}"

Move the file to the new name

shutil.move(os.path.join(folder_name, file),

os.path.join(folder_name, new_file_name))

Print a success message

print("Files renamed successfully.")

'''
```

## Output

The code calls the `toUpperCase` function to convert `prefix` to uppercase, but that function is not defined. To fix this issue, you can use the `upper()` method of the string class which converts a given string into uppercase. The modified code is shown below:

```
```python

import os

import shutil
```

```
# Get the folder name from the user

folder_name = input("Enter the folder name: ")

prefix = input("Enter the string to prepend to the filename: ")

text = prefix.upper()

# Check if the folder exists

if not os.path.isdir(folder_name):

    print("Folder does not exist.")

    exit(1)

# Get a list of all files in the folder

files = os.listdir(folder_name)

# Iterate over the files

for file in files:

    # Create a new file name with the prefix "draft_"

    new_filename = f"{text}_{file}"

    # Move the file to the new name

    shutil.move(os.path.join(folder_name, file),

os.path.join(folder_name, new_filename))

# Print a success message

print("Files renamed successfully.")
```

```
'''
```

Additionally, there are a few other improvements that can be made to the code:

1. The file extension of the new filenames is not kept. It's better to extract the

extension from the original filename and append it to the new filename.

2. The code assumes that there are no spaces in the folder name. However,

it is a good practice to handle spaces gracefully by replacing them with

underscores or hyphens.

3. Using the ``f`` string syntax for string interpolation is more readable and

concise than the traditional ``+`` operator.

4. The code doesn't handle errors that might occur during the renaming process.

It would be better to wrap the ``shutil.move`` call in a ``try...except`` block to

catch any potential errors.

Here is the improved code with these suggestions:

```
```python
```

```
import os
```

```

import shutil

Get the folder name from the user

folder_name = input("Enter the folder name: ")

prefix = input("Enter the string to prepend to the filename:
")

Check if the folder exists

if not os.path.isdir(folder_name):

print("Folder does not exist.")

exit(1)

Get a list of all files in the folder

files = os.listdir(folder_name)

Iterate over the files

for file in files:

Get the file extension

ext = os.path.splitext(file)[1]

Create a new file name with the prefix and the file
extension

new_filename = f"{prefix}_{os.path.splitext(file)[0]}{ext}"

Build the full paths for the original and new files

old_path = os.path.join(folder_name, file)

new_path = os.path.join(folder_name, new_filename)

```

```

Try to move the file

try:

shutil.move(old_path , new_path)

except Exception as e:

print(f"An error occurred while moving the file

{file}: {e}")

Print a success message

print("Files renamed successfully.")

...

These improvements make the code more robust and
flexible while also making

it easier to read and understand... (The response was
truncated because

it has reached the token limit. Try to increase the token limit
if you need a

longer response.)

```

*Bảng 19. Một prompt để gỡ lỗi và đánh giá mã Python*

Thật tuyệt vời. Nó không chỉ cho tôi biết cách giải quyết vấn đề, nó còn phát hiện ra rằng mã của tôi có nhiều lỗi hơn và cách giải quyết chúng. Phần cuối của prompt đã đưa ra các đề xuất để cải thiện mã nói chung.

## Còn về prompting đa phương thức?

Prompting cho mã vẫn sử dụng cùng một mô hình ngôn ngữ lớn thông thường. Prompting đa phương thức là một vấn đề riêng biệt, nó đề cập đến một kỹ thuật mà bạn sử dụng nhiều định dạng đầu vào để hướng dẫn một mô hình ngôn ngữ lớn, thay vì chỉ dựa vào văn bản. Điều này có thể bao gồm sự kết hợp của văn bản, hình ảnh, âm thanh, mã hoặc thậm chí các định dạng khác, tùy thuộc vào khả năng của mô hình và nhiệm vụ đang thực hiện.

## Các thực hành tốt nhất

Việc tìm ra prompt phù hợp đòi hỏi sự thử nghiệm. Language Studio trong Vertex AI là một nơi hoàn hảo để thử nghiệm các prompt của bạn, với khả năng kiểm tra trên các mô hình khác nhau.

Sử dụng các thực hành tốt nhất sau đây để trở thành một chuyên gia trong kỹ thuật prompt.

## Cung cấp ví dụ

Thực hành tốt nhất quan trọng nhất là cung cấp các ví dụ (one shot / few shot) trong một prompt. Điều này rất hiệu quả vì nó hoạt động như một công cụ giảng dạy mạnh mẽ. Những ví dụ này giới thiệu các đầu ra mong muốn hoặc các phản hồi tương tự, cho phép mô hình học hỏi từ chúng và điều chỉnh việc tạo ra của chính nó cho phù hợp. Nó giống như việc đưa cho mô hình một điểm tham chiếu hoặc mục tiêu để hướng tới, cải thiện độ chính xác, phong cách và giọng điệu của phản hồi để phù hợp hơn với mong đợi của bạn.

## Thiết kế đơn giản

Các prompt nên ngắn gọn, rõ ràng và dễ hiểu cho cả bạn và mô hình. Theo nguyên tắc chung, nếu nó đã khó hiểu đối với bạn thì có khả năng nó cũng sẽ khó hiểu đối với mô hình. Cố gắng không sử dụng ngôn ngữ phức tạp và không cung cấp thông tin không cần thiết.

Ví dụ:

### TRƯỚC KHI VIẾT:

Tôi đang đến thăm New York ngay bây giờ, và tôi muốn nghe thêm về các địa điểm tuyệt vời. Tôi đi cùng hai đứa trẻ 3 tuổi. Chúng tôi nên đi đâu trong kỳ nghỉ của mình?

### SAU KHI VIẾT LẠI:

Đóng vai một hướng dẫn viên du lịch cho khách du lịch. Mô tả các địa điểm tuyệt vời để tham quan ở Manhattan, New York với một đứa trẻ 3 tuổi.

Hãy thử sử dụng các động từ mô tả hành động. Dưới đây là một bộ ví dụ:

**Đóng vai, Phân tích, Phân loại, Phân lớp, Tương phản, So sánh, Tạo ra, Mô tả, Định nghĩa, Đánh giá, Trích xuất, Tìm, Sinh ra, Xác định, Liệt kê, Đo lường, Tổ chức, Phân tích cú pháp, Chọn, Dự đoán, Cung cấp, Xếp hạng, Đề xuất, Trả về, Truy xuất, Viết lại, Lựa chọn, Hiện thị, Sắp xếp, Tóm tắt, Dịch, Viết.**

## Cụ thể về kết quả đầu ra

Hãy cụ thể về đầu ra mong muốn. Một hướng dẫn ngắn gọn có thể không đủ để hướng dẫn LLM hoặc có thể quá chung chung. Việc cung cấp các chi tiết cụ thể trong prompt (thông qua prompting hệ thống hoặc ngữ cảnh) có thể giúp mô hình tập trung vào những gì có liên quan, cải thiện độ chính xác tổng thể.

Ví dụ:

**NÊN LÀM:**

Tạo một bài đăng blog 3 đoạn về 5 máy chơi game hàng đầu. Bài đăng blog nên cung cấp thông tin và hấp dẫn, và nên được viết theo phong cách trò chuyện.

**KHÔNG NÊN LÀM:**

Tạo một bài đăng blog về máy chơi game.

## Sử dụng Hướng dẫn thay vì Ràng buộc

Hướng dẫn và ràng buộc được sử dụng trong prompting để định hướng đầu ra của một LLM.

- Một hướng dẫn cung cấp các chỉ dẫn rõ ràng về định dạng, phong cách hoặc nội dung mong muốn của phản hồi. Hướng dẫn mô hình về những gì nó nên làm hoặc tạo ra.
- Một ràng buộc là một tập hợp các giới hạn hoặc ranh giới đối với phản hồi. Giới hạn những gì nó không nên làm hoặc nên tránh.

Nghiên cứu ngày càng tăng cho thấy rằng việc tập trung vào các hướng dẫn tích cực trong prompting có thể hiệu quả hơn là phụ thuộc nhiều vào các ràng buộc. Cách tiếp cận này phù hợp với cách con người ưa thích các hướng dẫn tích cực hơn là danh sách những điều không nên làm.

Hướng dẫn truyền đạt trực tiếp kết quả mong muốn, trong khi các ràng buộc có thể khiến mô hình phải đoán xem điều gì được phép. Nó mang lại sự linh hoạt và khuyến khích sự sáng tạo trong các ranh giới đã xác định, trong khi các ràng buộc có thể hạn chế tiềm năng của mô hình. Ngoài ra, một danh sách các ràng buộc có thể xung đột với nhau.

Các ràng buộc vẫn có giá trị nhưng trong một số tình huống nhất định. Để ngăn mô hình tạo ra nội dung có hại hoặc thiên vị hoặc khi cần một định dạng hoặc phong cách đầu ra nghiêm ngặt.

Nếu có thể, hãy sử dụng các hướng dẫn tích cực: thay vì nói cho mô hình biết những gì không nên làm, hãy nói cho nó biết những gì nên làm. Điều này có thể tránh được sự nhầm lẫn và cải thiện độ chính xác của đầu ra.

#### NÊN LÀM:

Tạo một bài đăng blog 1 đoạn về 5 máy chơi game hàng đầu. Chỉ thảo luận về máy chơi game, công ty sản xuất nó, năm ra mắt và tổng doanh số.

#### KHÔNG NÊN LÀM:

Tạo một bài đăng blog 1 đoạn về 5 máy chơi game hàng đầu. Không liệt kê tên các trò chơi.

Thực hành tốt nhất là: hãy bắt đầu bằng cách ưu tiên các hướng dẫn, nêu rõ những gì bạn muốn mô hình làm và chỉ sử dụng các ràng buộc khi cần thiết cho sự an toàn, rõ ràng hoặc các yêu cầu cụ thể. Thử nghiệm và lặp lại để kiểm tra các sự kết hợp khác nhau của hướng dẫn và ràng buộc để tìm ra những gì hoạt động tốt nhất cho các nhiệm vụ cụ thể của bạn, và ghi lại chúng.

## Kiểm soát độ dài token tối đa

Để kiểm soát độ dài của một phản hồi LLM được tạo ra, bạn có thể đặt giới hạn token tối đa trong cấu hình hoặc yêu cầu rõ ràng một độ dài cụ thể trong prompt của bạn.

Ví dụ:

"Giải thích vật lý lượng tử trong một tin nhắn dài bằng một tweet."

## Sử dụng biến trong prompt

Để tái sử dụng các prompt và làm cho chúng linh hoạt hơn, hãy sử dụng các biến trong prompt, có thể được thay đổi cho các đầu vào khác nhau.

Ví dụ, như được hiển thị trong Bảng 20, một prompt cung cấp thông tin về một thành phố. Thay vì mã hóa cứng tên thành phố trong prompt, hãy sử dụng một biến. Các biến có thể tiết kiệm thời gian và công sức của bạn bằng cách cho phép bạn tránh lặp lại chính mình. Nếu bạn cần sử dụng cùng một mẫu thông tin trong nhiều prompt, bạn có thể lưu trữ nó trong một biến và sau đó tham chiếu đến biến đó trong mỗi prompt. Điều này rất có ý nghĩa khi tích hợp các prompt vào các ứng dụng của riêng bạn.

|               |                                                                                                                                                      |
|---------------|------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Prompt</b> | VARIABLES<br><br>{city} = "Amsterdam"<br><br>PROMPT<br><br>You are a travel guide. Tell me a fact about the city: {city}                             |
| <b>Output</b> | Amsterdam is a beautiful city full of canals, bridges, and narrow streets. It's a great place to visit for its rich history, culture, and nightlife. |

*Bảng 20. Sử dụng các biến trong prompts*

## Thử nghiệm với các định dạng đầu vào và văn phong

Các mô hình khác nhau, cấu hình mô hình, định dạng prompt, lựa chọn từ ngữ và các lần gửi có thể mang lại kết quả khác nhau. Do đó, điều quan trọng là phải thử nghiệm với các thuộc tính của prompt như phong cách, lựa chọn từ ngữ và loại prompt (zero shot, few shot, system prompt).

Ví dụ, một prompt với mục tiêu tạo văn bản về máy chơi game mang tính cách mạng Sega Dreamcast, có thể được xây dựng dưới dạng một **câu hỏi**, một **câu khẳng định** hoặc một **hướng dẫn**, dẫn đến các kết quả khác nhau:

**Câu hỏi** : Sega Dreamcast là gì và tại sao nó lại là một máy chơi game mang tính cách mạng như vậy?

**Câu khẳng định** : Sega Dreamcast là một máy chơi game thế hệ thứ sáu được Sega phát hành vào năm 1999. Nó...

**Hướng dẫn** : Viết một đoạn văn duy nhất mô tả máy chơi game Sega Dreamcast và giải thích tại sao nó lại mang tính cách mạng như vậy.

Đối với prompting few-shot với các tác vụ phân loại, hãy trộn lẫn các lớp

Nói chung, thứ tự của các ví dụ few-shot của bạn không quá quan trọng. Tuy nhiên, khi thực hiện các tác vụ phân loại, hãy đảm bảo bạn trộn lẫn các lớp phản hồi có thể có trong các ví dụ few-shot. Điều này là do bạn có thể đang overfitting với thứ tự cụ thể của các ví dụ.

Bằng cách trộn lẫn các lớp phản hồi có thể có, bạn có thể đảm bảo rằng mô hình đang học cách xác định các đặc điểm chính của mỗi lớp, thay vì chỉ đơn giản là ghi nhớ thứ tự của các ví dụ. Điều này sẽ dẫn đến hiệu suất mạnh mẽ và tổng quát hóa tốt hơn trên dữ liệu chưa thấy.

Một nguyên tắc tốt là bắt đầu với 6 ví dụ few shot và bắt đầu kiểm tra độ chính xác từ đó.

## Thích ứng với các bản cập nhật của mô hình

Điều quan trọng là bạn phải cập nhật các thay đổi về kiến trúc mô hình, dữ liệu được thêm vào và các khả năng. Hãy thử các phiên bản mô hình mới hơn và điều chỉnh các prompt của bạn để tận dụng tốt hơn các tính năng mới của mô hình. Các công cụ như Vertex AI Studio rất tuyệt vời để lưu trữ, kiểm tra và ghi lại các phiên bản khác nhau của prompt của bạn.

## Thử nghiệm với các định dạng đầu ra

Bên cạnh định dạng đầu vào của prompt, hãy xem xét thử nghiệm với định dạng đầu ra. Đối với các nhiệm vụ không sáng tạo như trích xuất, lựa chọn, phân tích cú pháp, sắp xếp, xếp hạng hoặc phân loại dữ liệu, hãy thử để đầu ra của bạn được trả về ở định dạng có cấu trúc như JSON hoặc XML.

Có một số lợi ích trong việc trả về các đối tượng JSON từ một prompt trích xuất dữ liệu. Trong một ứng dụng thực tế, tôi không cần phải tạo định dạng JSON này theo cách thủ công, tôi đã có thể trả về dữ liệu theo một thứ tự đã sắp xếp (rất tiện lợi khi làm việc với các đối tượng datetime), nhưng quan trọng nhất, bằng cách yêu cầu định dạng JSON, nó buộc mô hình phải tạo ra một cấu trúc và hạn chế ảo giác (hallucinations).

Tóm lại, lợi ích của việc sử dụng JSON cho đầu ra của bạn:

- Luôn trả về theo cùng một phong cách
- Tập trung vào dữ liệu bạn muốn nhận
- Ít có khả năng bị ảo giác
- Làm cho nó nhận biết được mối quan hệ
- Bạn nhận được các kiểu dữ liệu
- Bạn có thể sắp xếp nó

[Bảng 4](#) trong phần prompting few-shot cho thấy một ví dụ về cách trả về đầu ra có cấu trúc.

## Sửa lỗi JSON (JSON Repair)

Mặc dù việc trả về dữ liệu ở định dạng JSON mang lại nhiều lợi thế, nó không phải là không có nhược điểm. Bản chất có cấu trúc của JSON, mặc dù có lợi cho việc phân tích cú pháp và sử dụng trong các ứng dụng, đòi hỏi nhiều token hơn đáng kể so với văn bản thuần túy, dẫn đến tăng thời gian xử lý và chi phí cao hơn.

Hơn nữa, sự dài dòng của JSON có thể dễ dàng tiêu tốn toàn bộ cửa sổ đầu ra, trở nên đặc biệt có vấn đề khi việc tạo ra bị cắt đột ngột do giới hạn token. Việc cắt ngắn này thường dẫn đến JSON không hợp lệ, thiếu các dấu ngoặc nhọn hoặc ngoặc vuông đóng quan trọng, làm cho đầu ra không thể sử dụng được.

May mắn thay, các công cụ như thư viện json-repair (có sẵn trên PyPI) có thể là vô giá trong những tình huống này. Thư viện này cố gắng một cách thông minh để tự động sửa các đối tượng JSON không hoàn chỉnh hoặc bị định dạng sai, làm cho nó trở thành một đồng minh quan trọng khi làm việc với JSON do LLM tạo ra, đặc biệt là khi xử lý các vấn đề cắt ngắn tiềm ẩn.

## Thiết kế Prompt có cấu trúc (Schema)

Sử dụng JSON có cấu trúc làm đầu ra là một giải pháp tuyệt vời, như chúng ta đã thấy nhiều lần trong bài viết này.

Nhưng còn về đầu vào thì sao? Mặc dù JSON rất tuyệt vời để cấu trúc đầu ra mà LLM tạo ra, nó cũng có thể cực kỳ hữu ích để cấu trúc đầu vào bạn cung cấp. Đây là lúc JSON Schema phát huy tác dụng.

Một JSON Schema xác định cấu trúc và các kiểu dữ liệu dự kiến của đầu vào JSON của bạn. Bằng cách cung cấp một schema, bạn cung cấp cho LLM một bản thiết kế rõ ràng về dữ liệu mà nó nên mong đợi, giúp nó tập trung sự chú ý vào thông tin liên quan và giảm nguy cơ giải thích sai đầu vào. Hơn nữa, schema có thể giúp thiết lập mối quan hệ giữa các phần dữ liệu khác nhau và thậm chí làm cho LLM "nhận biết thời gian" bằng cách bao gồm các trường ngày hoặc dấu thời gian với các định dạng cụ thể.

Đây là một ví dụ đơn giản:

Giả sử bạn muốn sử dụng một LLM để tạo mô tả cho các sản phẩm trong một danh mục thương mại điện tử. Thay vì chỉ cung cấp một mô tả văn bản tự do của sản phẩm, bạn có thể sử dụng một JSON schema để xác định các thuộc tính của sản phẩm:

```

{
 "type": "object",
 "properties": {
 "name": { "type": "string", "description": "Product name" },
 "category": { "type": "string", "description": "Product category" },
 "price": { "type": "number", "format": "float", "description": "Product price" },
 "features": {
 "type": "array",
 "items": { "type": "string" },
 "description": "Key features of the product"
 },
 "release_date": { "type": "string", "format": "date", "description": "Date the product was released" }
 },
}

```

*Đoạn mã 5. Định nghĩa prompt đầu ra có cấu trúc*

Sau đó, bạn có thể cung cấp dữ liệu sản phẩm thực tế dưới dạng một đối tượng JSON tuân thủ schema này:

```

{
 "name": "Wireless Headphones",
 "category": "Electronics",
 "price": 99.99,
 "features": ["Noise cancellation", "Bluetooth 5.0", "20-hour battery life"],
 "release_date": "2023-10-27"
}

```

*Đoạn mã 6. Đầu ra có cấu trúc từ LLM*

Bằng cách xử lý trước dữ liệu của bạn và thay vì cung cấp các tài liệu đầy đủ chỉ cung cấp cả schema và dữ liệu, bạn cung cấp cho LLM một sự hiểu biết rõ ràng về các thuộc tính của sản phẩm, bao gồm cả ngày phát hành của nó, làm cho nó có nhiều khả năng tạo ra một mô tả chính xác và phù hợp hơn.

Cách tiếp cận đầu vào có cấu trúc này, hướng sự chú ý của LLM đến các trường liên quan, đặc biệt có giá trị khi làm việc với khối lượng lớn dữ liệu hoặc khi tích hợp LLM vào các ứng dụng phức tạp.

Một ví dụ về structure prompt (Prompt có cấu trúc)

Nội dung prompt này dùng để phục chế ảnh đã cũ nát, ví dụ ảnh của ông bà, cụ... chụp từ xưa này gần hỏng cần phục chế! Ngay cả đối với một chuyên gia phục chế cũng mất khá nhiều thời gian mới có thể phục hồi hoàn toàn kể cả có nhiều kỹ năng.

Ngày nay chúng ta có thể nhờ AI, cụ thể là Gemini với Gemini 2.5 Flash Image để thực hiện việc này cùng với một prompt có cấu trúc với các bước như sau:

**Bước 1** : Mở Gemini chọn hình quả chuối.



**Bước 2** : Tải ảnh cần phục chế lên (bấm dấu cộng).

**Bước 3** : Copy & paste đoạn code sau vào và enter:

```
{
```

```
"version": "1.0",
```

```
"task": "image_restore",
```

"caption": "Phục chế ảnh cũ – giữ nguyên trắng đen/sepia, trung thành tuyệt đối với nét gốc",

"notes": "Chỉ làm sạch và phục hồi, không tô màu, không thay đổi đặc điểm khuôn mặt, tóc, khăn quần hay tư thế.",

"input_image": "REPLACE_WITH_IMAGE_ID_OR_PATH",

"preprocess": {

"detect_and_isolate_original_photo": true,

"auto_crop_photo_edges": true,

"clean_edges": true,

"perspective_correction": true,

"glare_reduction": "moderate",

"reflection_removal": true

},

"subject_constraints": {

"keep_identity": true,

"lock_features": [

"eyes", "nose", "lips", "eyebrows",

"jawline", "face_shape", "ears", "hairline",

"headscarf"

],

"expression_policy": "preserve_original",

"no_feature_modification": true

},

```
"retouching": {

 "repair_cracks": "strict",

 "remove_dust_scratches": "strict",

 "remove_stains": "strict",

 "remove_folds": true,

 "restore_faded_details": true,

 "sharpen_soft_areas": true,

 "clarity_boost": "subtle"

},
```

```
"background": {

 "policy": "preserve_original",
```

"restore_damage": true,

"texture_cleanup": true,

"remove_paper_grain": true

},

"color_tone": {

"overall": "grayscale_or_sepia_original",

"contrast": "balanced",

"brightness_adjustment": "subtle",

"auto_tone_balance": true,

"avoid_artificial_coloring": true

},

```
"detail_sharpness": {

"method": "gentle_edge_enhancement",

"amount": 0.2,

"noise_reduction": { "luminance": 0.15, "chroma": 0.15, "preserve_details": 0.9 }

},

"clean_up": {

"remove_noise": true,

"remove_artifacts": true,

"remove_scratches": "strict",

"remove_dust": "strict",

"deblotching": true,
```

"heritage_preservation_strict": true,

"archival_quality": "museum-grade restoration"

},

"controls": {

"face_identity_lock": 0.98,

"pose_lock": 1.0,

"restoration_strength": 0.9

},

"output": {

"resolution": "6000x8000",

"dpi": 600,

"format": "TIFF",

"color_space": "Grayscale",

"bit_depth": "16-bit"

},

"safety_bounds": {

"do_not": [

"add colors",

"change face geometry",

"remove or alter headscarf",

"change expression",

"add digital effects",

"over-smooth skin"

]

},

"metadata": {

"locale": "vi-VN",

"creator": "heritage photo restoration specialist",

"purpose": "phục chế ảnh cũ nguyên gốc, trung thành tuyệt đối với khuôn mặt và các chi tiết",

"workflow": "clean damage, restore sharpness, preserve identity, archival quality"

}

}

## Thử nghiệm cùng với các chuyên gia prompt khác

Nếu bạn đang trong tình huống phải cố gắng nghĩ ra một prompt tốt, bạn có thể muốn tìm nhiều người để cùng thử. Khi mọi người đều tuân theo các thực hành tốt nhất (như được liệt kê trong chương này), bạn sẽ thấy sự khác biệt về hiệu suất giữa tất cả các lần thử prompt khác nhau.

## Các thực hành tốt nhất cho CoT

Đối với prompting CoT, việc đặt câu trả lời sau phần suy luận là bắt buộc vì việc tạo ra phần suy luận sẽ thay đổi các token mà mô hình nhận được khi nó dự đoán câu trả lời cuối cùng.

Với CoT và tự nhất quán, bạn cần có khả năng trích xuất câu trả lời cuối cùng từ prompt của mình, tách biệt khỏi phần suy luận.

Đối với prompting CoT, hãy đặt nhiệt độ (temperature) thành 0.

Prompting chuỗi tư duy dựa trên giải mã tham lam (greedy decoding), dự đoán từ tiếp theo trong một chuỗi dựa trên xác suất cao nhất được gán bởi mô hình ngôn ngữ. Nói chung, khi sử dụng suy luận để đưa ra câu trả lời cuối cùng, có khả năng chỉ có một câu trả lời đúng duy nhất. Do đó, temperature nên luôn được đặt thành 0.

## Ghi lại tài liệu về các lần thử nghiệm prompt khác nhau

Mẹo cuối cùng đã được đề cập trước đó trong chương này, nhưng chúng tôi không thể nhấn mạnh đủ tầm quan trọng của nó: ghi lại tài liệu chi tiết về các lần thử prompt của bạn để bạn có thể học hỏi theo thời gian những gì đã hoạt động tốt và những gì không.

Đầu ra của prompt có thể khác nhau giữa các mô hình, giữa các cài đặt lấy mẫu, và thậm chí giữa các phiên bản khác nhau của cùng một mô hình. Hơn nữa, ngay cả với các prompt giống hệt nhau cho cùng một mô hình, những khác biệt nhỏ trong định dạng câu đầu ra và lựa chọn từ ngữ có thể xảy ra. (Ví dụ, như đã đề cập trước đó, nếu hai token có cùng xác suất dự đoán, các trường hợp hòa có thể được giải quyết một cách ngẫu nhiên. Điều này sau đó có thể ảnh hưởng đến các token được dự đoán tiếp theo.)

Chúng tôi khuyên bạn nên tạo một Google Sheet với Bảng 21 làm mẫu. Lợi ích của phương pháp này là bạn có một bản ghi đầy đủ khi bạn chắc chắn sẽ phải xem lại công việc prompting của mình - hoặc để tiếp tục nó trong tương lai (bạn sẽ ngạc nhiên về việc bạn có thể quên bao nhiêu chỉ sau một thời gian ngắn), để kiểm tra hiệu suất của prompt trên các phiên bản khác nhau của một mô hình, và để giúp gỡ lỗi các lỗi trong tương lai.

Ngoài các trường trong bảng này, cũng hữu ích khi theo dõi phiên bản của prompt (lần lặp), một trường để ghi lại kết quả là OK/KHÔNG OK/ĐÔI KHI OK, và một trường để ghi lại phản hồi. Nếu bạn may mắn được sử dụng Vertex AI Studio, hãy lưu các prompt của bạn (sử dụng cùng tên và phiên bản như được liệt kê trong tài liệu của bạn) và theo dõi siêu liên kết đến prompt đã lưu trong bảng. Bằng cách này, bạn luôn chỉ cách một cú nhấp chuột để chạy lại các prompt của mình.

Khi làm việc trên một *Mô hình sinh ngôn ngữ có tăng cường truy xuất* (retrieval augmented generation - RAG), bạn cũng nên ghi lại các khía cạnh cụ thể của hệ thống RAG ảnh hưởng đến nội dung được chèn vào prompt, bao gồm truy vấn, cài đặt chunk, đầu ra chunk, và các thông tin khác.

Một khi bạn cảm thấy prompt đã gần hoàn hảo, hãy đưa nó vào codebase dự án của bạn. Và trong codebase, hãy lưu các prompt trong một tệp riêng biệt với mã, để dễ bảo trì hơn. Cuối cùng, lý tưởng nhất là các prompt của bạn là một phần của một hệ thống được vận hành, và với tư cách là một chuyên gia prompt, bạn nên dựa vào các bài kiểm tra tự động và các quy trình đánh giá để hiểu rõ prompt của bạn tổng quát hóa tốt như thế nào cho một nhiệm vụ.

Kỹ thuật prompt là một quá trình lặp đi lặp lại. Tạo và kiểm tra các prompt khác nhau, phân tích và ghi lại kết quả. Tinh chỉnh prompt của bạn dựa trên hiệu suất của mô hình. Tiếp tục thử nghiệm cho đến khi bạn đạt được đầu ra mong muốn. Khi bạn thay đổi một mô hình hoặc cấu hình mô hình, hãy quay lại và tiếp tục thử nghiệm với các prompt đã sử dụng trước đó.

|              |                                                        |
|--------------|--------------------------------------------------------|
| <b>Name</b>  | [name and version of your prompt]                      |
| <b>Goal</b>  | [One sentence explanation of the goal of this attempt] |
| <b>Model</b> | [name and version of the used model]                   |

|                    |                                            |                    |          |
|--------------------|--------------------------------------------|--------------------|----------|
| <b>Temperature</b> | [value between 0 - 1]                      | <b>Token Limit</b> | [number] |
| <b>Top-K</b>       | [number]                                   | <b>Top-P</b>       | [number] |
| <b>Prompt</b>      | [Write all the full prompt]                |                    |          |
| <b>Output</b>      | [Write out the output or multiple outputs] |                    |          |

*Bảng 21. Một mẫu để ghi lại tài liệu về các prompt*

## Tổng kết

Sách trắng này thảo luận về kỹ thuật prompt. Chúng ta đã học được các kỹ thuật prompting khác nhau, chẳng hạn như:

- Prompting Zero shot
- Prompting Few shot
- Prompting theo hệ thống
- Prompting theo vai trò
- Prompting theo ngữ cảnh
- Prompting lùi một bước
- Chuỗi tư duy
- Tự nhất quán
- Cây tư duy
- ReAct

Chúng ta thậm chí đã xem xét các cách bạn có thể tự động hóa các prompt của mình.

Sau đó, sách trắng thảo luận về những thách thức của gen AI như các vấn đề có thể xảy ra khi các prompt của bạn không đủ. Chúng ta đã kết thúc với các thực hành tốt nhất về cách trở thành một chuyên gia tốt hơn về prompt.

---

[1] (ND) Prompt theo ngữ cảnh AI là “chỉ dẫn” hoặc “Câu lệnh” cho AI.

[2] (TG) Google. (2023). Gemini by Google. Truy cập: <https://gemini.google.com>.

[3] (ND) Token: Đơn vị ngôn ngữ nhỏ nhất mà mô hình dùng để dự đoán, có thể là từ, âm tiết, hoặc ký hiệu. LLM không “nhìn” văn bản như con người đọc từng chữ, mà chuyển văn bản thành dãy số (embedding) thông qua việc cắt nhỏ thành token. Sau đó mô hình học cách dự đoán token kế tiếp trong chuỗi.

[4] (TG) Google. (2024). Hướng dẫn Prompt Gemini cho Google Workspace. Truy cập: <https://inthecloud.withgoogle.com/gemini-for-google-workspace-prompt-guide/dl-cd.html>.

[5] (TG) Google Cloud. (2023). Giới thiệu về Prompting. Truy cập: <https://cloud.google.com/vertex-ai/generative-ai/docs/learn/prompts/introduction-prompt-design>.

[6] (ND) Hàm softmax là một hàm toán học thường dùng trong học máy (machine learning), đặc biệt trong các mô hình phân loại nhiều lớp (multi-class classification). Softmax chuyển đổi một vector giá trị “thô” (logits, tức đầu ra chưa chuẩn hóa của mô hình) thành phân phối xác suất trên các lớp. Giá trị lớn hơn trong đầu vào  $Z_i$  sẽ tương ứng với xác suất cao hơn sau khi qua softmax.

[7] (ND) Top-K kiểm soát việc chọn ra K token có xác suất cao nhất từ phân phối xác suất mà mô hình dự đoán. Sau đó, mô hình chỉ lấy mẫu (sampling) trong tập hợp K token đó. VD: Nếu mô hình dự đoán xác suất từ tiếp theo: “mèo” (0.5), “chó” (0.3), “cá” (0.1), “vịt” (0.05), “voi” (0.05). Với Top-K = 2 → chỉ xét “mèo”, “chó” → mô hình chọn ngẫu nhiên trong 2 từ này.

[8] (ND) Thay vì cố định số lượng token, Top-P chọn tập hợp nhỏ nhất các token sao cho tổng xác suất  $\geq P$  (trong khoảng từ 0 đến 1). Ý nghĩa: Thích ứng linh hoạt: khi phân phối xác suất lệch nhiều → ít token được chọn; khi phân phối dàn trải → nhiều token được chọn; P nhỏ (ví dụ 0.5) → mô hình chỉ xét những token có khả năng cao nhất; P lớn (ví dụ 0.95) → mô hình đa dạng hơn vì xét thêm nhiều token ít xác suất. VD: Nếu Top-P = 0.8 → mô hình lấy “mèo” (0.5) + “chó” (0.3) = 0.8 → dừng lại. Nếu Top-P = 0.95 → lấy thêm “cá” (0.1) = 0.9, cộng thêm “vịt” (0.05) = 0.95 → vậy mô hình xét cả 4 token.

[9] (TG) Google Cloud. (2023). Nội dung yêu cầu của mô hình văn bản: Phương pháp lấy mẫu Top-P và Top-K. Truy cập: [https://cloud.google.com/vertex-ai/docs/generative-ai/model-reference/text#request_body](https://cloud.google.com/vertex-ai/docs/generative-ai/model-reference/text#request_body).

[10] (TG) (TG) Wei, J., và cộng sự. (2023): Các mô hình ngôn ngữ tinh chỉnh (fine-tuned) theo phương pháp zero-shot là những bộ học zero-shot (Zero Shot - Fine Tuned language models are zero shot learners). Truy cập từ:

<https://arxiv.org/pdf/2109.01652.pdf>.

[11] (TG) Google Cloud, 2023, Google Cloud Model Garden. Truy cập:

<https://cloud.google.com/model-garden>.

[12] (TG) Brown, T., et al., 2023, Few Shot - Language Models are Few Shot learners. Truy cập:

<https://arxiv.org/pdf/2005.14165.pdf>.

[13] (ND) Hallucinations: Hiện tượng mô hình AI tạo ra thông tin sai sự thật, không tồn tại hoặc bịa đặt, nhưng lại được diễn đạt rất tự tin như đúng. Đây là một vấn đề phổ biến của các mô hình sinh ngôn ngữ (generative AI).

[14] (TG) Zheng, L., et al., 2023, Take a Step Back: Evoking Reasoning via Abstraction in Large Language Models.

Truy cập: <https://openreview.net/pdf?id=3bq3jsvcQ1>.

[15] (TG) Wei, J. và cộng sự, 2023, Chuỗi tư duy trong nhắc lệnh (Chain of Thought Prompting). Truy cập: <https://arxiv.org/pdf/2201.11903.pdf>.

[16] (TG) Google Cloud Platform, 2023, Chain of Thought and React. Truy cập:

[https://github.com/GoogleCloudPlatform/generative-ai/blob/main/language/prompts/examples/chain_of_thought_react.ipynb](https://github.com/GoogleCloudPlatform/generative-ai/blob/main/language/prompts/examples/chain_of_thought_react.ipynb).

[17] (ND) NLP: Natural Language Processing – Quy trình xử lý ngôn ngữ tự nhiên.

[18] (TG) Wang, X., và cộng sự, 2023, Self Consistency Improves Chain of Thought reasoning in language models. Truy cập: <https://arxiv.org/pdf/2203.11171.pdf>.

[19] (TG) Yao, S., và cộng sự, 2023, Tree of Thoughts: Deliberate Problem Solving with Large Language Models.

truy cập: <https://arxiv.org/pdf/2305.10601.pdf>.

[20] (TG) Yao, S., và các cộng sự., 2023, ReAct: Synergizing Reasoning and Acting in Language Models. Truy cập:

<https://arxiv.org/pdf/2210.03629.pdf>.

[21] (TG) Google Cloud Platform, 2023, Advance Prompting: Chain of Thought and React. Truy cập:

[https://github.com/GoogleCloudPlatform/applied-ai-engineering-samples/blob/main/genaion-vertex-ai/advanced_prompting_training/cot_react.ipynb](https://github.com/GoogleCloudPlatform/applied-ai-engineering-samples/blob/main/genaion-vertex-ai/advanced_prompting_training/cot_react.ipynb).

[22] (TG) Zhou, C., và cộng sự, 2023, Automatic Prompt Engineering - Large Language Models are Human-Level Prompt Engineers. Truy cập:

<https://arxiv.org/pdf/2211.01910.pdf>.





## THƯ VIỆN EBOOK **SÁCH MỚI.NET**



HÀNG NGHÌN ĐẦU SÁCH HAY ĐANG CHỜ BẠN  
[WWW.SACHMOI.NET](http://WWW.SACHMOI.NET)

