

Foundation of Prompt Engineering

Team GenAIO

Hoang-Bach Ngo

Minh-Hung An

Ngày 6 tháng 3 năm 2024

Phần I: Tổng quan về Prompt Engineering

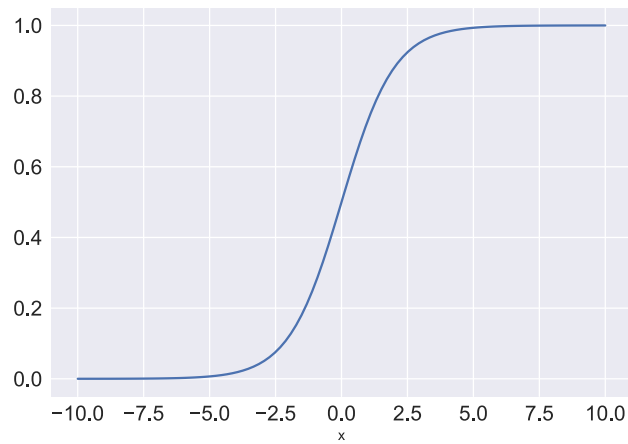
1 LLM Settings

Khi tương tác với các mô hình ngôn ngữ lớn, bạn sẽ thường xuyên điều chỉnh một loạt các tham số và cài đặt. Việc tinh chỉnh những tham số này là bước không thể thiếu để làm cho các phản hồi trở nên đáng tin cậy và phù hợp hơn với yêu cầu của bạn, và điều này đôi khi đòi hỏi phải thử nghiệm để xác định được cách cấu hình tối ưu. Dưới đây là một số tham số cơ bản bạn thường gặp trong quá trình sử dụng các mô hình LLM.

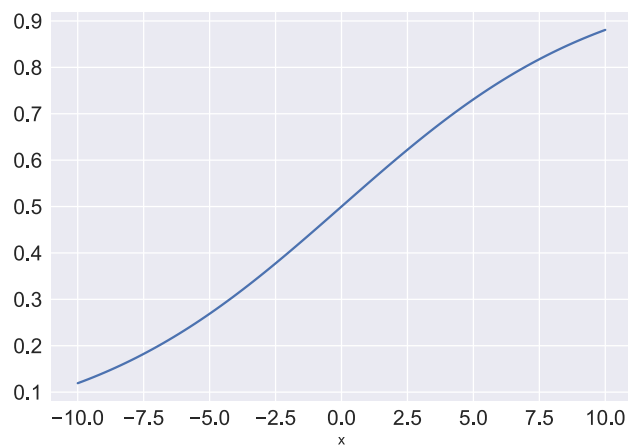
Temperature - Như chúng ta có thể đã biết thì Large Language Model (LLM) hoạt động theo cơ chế phân loại từ tiếp theo, và như bao bài toán phân loại khác thì output đầu ra của LLM sẽ áp dụng thêm một hàm softmax để tạo ra phân phối xác suất từ tiếp theo sao cho tổng xác suất cộng lại bằng 1. Temperature thật ra là một tham số điều khiển độ “mượt mà” của hàm softmax đó bằng cách lấy logits output ra từ LLM chia cho tham số temperature.

$$P(i) = \frac{\exp(\text{logit}_i/T)}{\sum_j \exp(\text{logit}_j/T)} \quad (1)$$

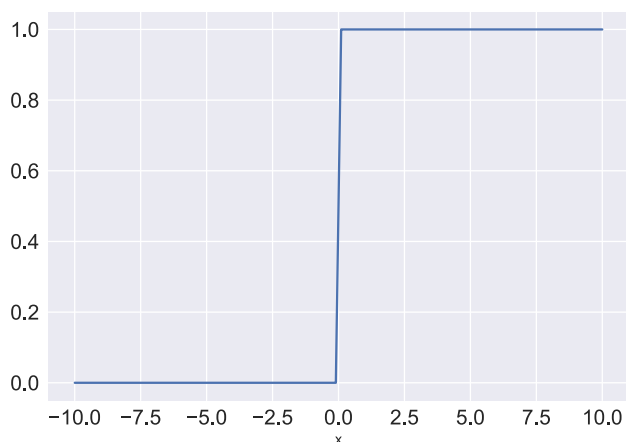
Công thức trên là công thức hàm softmax, trong đó tham số temperature được kí hiệu bằng chữ T , và logit_n kí hiệu output thứ n ở lớp cuối của mô hình LLM. Để giúp bạn dễ hình dung, sau đây là một visualization nho nhỏ với hàm sigmoid - một hàm có những đặc điểm tương tự như softmax.



Hình 1: Khi temperature được set bằng 1.0, tức là không có một sự thay đổi nào so với hàm softmax bình thường, lúc này sẽ là như apply một hàm softmax bình thường.



Hình 2: Khi temperature được đặt thành giá trị lớn hơn 1.0, nó sẽ "làm mượt" lại phân bố xác suất, làm cho phân bố xác suất trở nên phẳng hơn. Điều này có nghĩa là các từ có xác suất thấp hơn sẽ có cơ hội được chọn cao hơn, dẫn đến output model có tính ngẫu nhiên cao hơn.



Hình 3: Ngược lại, khi temperature được đặt thành giá trị nhỏ hơn 1.0 (ví dụ: 0,5), nó sẽ "làm sắc nét" phân bố xác suất. Điều này làm cho mô hình có nhiều khả năng chọn các từ có xác suất cao hơn, dẫn đến kết quả đầu ra tập trung và mang tính xác định hơn. Khi temperature bằng 0 (hình 4), phân bố xác suất sẽ được đẩy từ có logits cao nhất bằng 1 và tất cả các từ còn lại bằng 0, dẫn đến output của model sẽ luôn là từ có xác suất cao nhất.

top_p - Tham số này được áp dụng trong kỹ thuật sampling gọi là **nucleus sampling**, giúp điều chỉnh độ đa dạng trong kết quả mà mô hình sinh ra. Theo phương pháp này, chúng ta sẽ chọn ra nhóm các từ có xác suất cao nhất, đảm bảo rằng tổng xác suất của nhóm này vượt qua một ngưỡng nhất định, ngưỡng này là top_p. Cách làm này giúp mở rộng khả năng chọn từ của mô hình, tạo ra kết quả phong phú hơn. Nếu chúng ta cần câu trả lời chính xác, dựa trên dữ liệu cụ thể, giá trị top_p nên được giữ ở mức thấp. Trong trường hợp muốn nhận được đầu ra đa dạng hơn, có thể tăng giá trị này. Tuy nhiên, lưu ý chỉ điều chỉnh hoặc tham số temperature hoặc top_p, không nên cùng lúc thay đổi cả hai để tránh gây rối loạn cho mô hình.

Max Length - Chúng ta có thể dễ dàng điều chỉnh số lượng từ tối đa mà mô hình sinh ra bằng cách thiết lập giới hạn Max Length. Việc này giúp hạn chế những kết quả quá dài hoặc không mấy liên quan, đồng thời cũng giúp chúng ta kiểm soát được chi phí tốt hơn.

Stop Sequences - Đây là cách thiết lập một chuỗi dừng giúp ngăn mô hình sinh thêm từ. Khi mô hình nhận diện một từ trong chuỗi dừng này, nó sẽ ngừng việc tạo thêm nội dung. Xác định stop sequences là một phương pháp hữu ích để kiểm soát chiều dài và bố cục của đầu ra mô hình. Chẳng hạn, nếu muốn mô hình chỉ tạo ra danh sách tối đa 10 mục, ta có thể đặt từ "11." vào trong danh sách stop sequences.

Frequency Penalty - Tham số này giúp chúng ta có thể áp dụng một hình phạt lên từ tiếp theo dựa trên số lần từ đó xuất hiện trong kết quả và prompt. frequency penalty càng cao, từ đó càng ít có cơ hội xuất hiện lại. Cơ chế này giúp giảm thiểu việc lặp lại các từ trong kết quả mà mô hình tạo ra, bằng cách áp đặt mức phạt lớn hơn cho những từ thường xuyên xuất hiện.

Presence Penalty - Tham số này giúp ta có thể áp dụng hình phạt cho các từ lặp lại, nhưng khác với frequency penalty, mức phạt này được áp dụng như nhau cho mọi từ lặp lại, không phân biệt một từ xuất hiện hai lần hay mười lần, chúng đều nhận mức phạt giống nhau. Cách làm này giúp hạn chế việc mô hình lặp đi lặp lại một cụm từ nhiều lần trong kết quả của mình. Để tạo ra nội dung đa dạng và sáng tạo hơn, chúng ta có thể tăng mức độ phạt này lên. Ngược lại, nếu muốn mô hình tập trung hơn vào một số ý tưởng nhất định, chúng ta có thể giảm mức phạt xuống.

Để tổng kết lại, sau đây là một số đề xuất của nhóm khi khởi tạo các tham số này:

- **Temperature:** Ban đầu khi nhìn vào tham số này, chúng ta sẽ dễ bị lầm tưởng đây là tham số chạy từ 0 đến 1. Thực tế thì như giải thích ở trên thì ta có thể thấy là temperature không hề là một biến chạy từ 0 đến 1. Vậy ta nên chọn tham số này như thế nào để tối ưu nhất?
 - **Chọn temperature = 0.0** nếu chúng ta muốn kết quả model không có tính ngẫu nhiên và luôn giống nhau với những input giống nhau. Lúc nào mô hình LLM luôn chọn từ có xác suất cao nhất.
 - **Chọn temperature = 0.7, 0.8** nếu chúng ta muốn kết quả model “sắc nét” hơn nhưng vẫn có tính random, đây cũng là khoảng các “cao nhân” trên mạng hay truyền tai nhau để model ra kết quả tốt và vẫn có tính sáng tạo.
 - **Chọn temperature > 1** nếu chúng ta muốn model có tính sáng tạo cao, tuy nhiên điều này cũng đem lại rủi ro là kết quả model có thể không quá tốt, và nếu set quá cao thì kết quả từ model sẽ là ngẫu nhiên hoàn toàn.
- **Top P**
 - **Tìm câu trả lời chính xác:** Giữ giá trị Top P thấp (0.1 - 0.5).
 - **Tìm câu trả lời sáng tạo:** Tăng giá trị Top P (0.6 - 0.9).
- **Max Length**
 - **Điều chỉnh tùy theo nhu cầu của task:** đối với phản hồi ngắn, giới hạn số lượng token (ví dụ: 100-200 tokens); đối với bài viết dài hơn, tăng số lượng token tối đa.
- **Stop Sequences**
 - **Đặt các chuỗi cụ thể để kết thúc kết quả trả về:** nếu bạn muốn mô hình chỉ tạo ra một danh sách với 10 item, bạn có thể thêm "11" làm Stop Sequences.
- **Frequency Penalty**
 - **Để giảm sự lặp lại của từ:** Sử dụng giá trị trung bình đến cao (0.5 - 1.0).
 - **Nếu không quan tâm đến sự lặp lại:** Giữ giá trị thấp (0.0 - 0.4).
- **Presence Penalty**
 - **Để tạo kết quả đa dạng:** Sử dụng giá trị cao (0.6 - 1.0).
 - **Để duy trì độ chính xác của mô hình:** Sử dụng giá trị thấp (0.0 - 0.4).

2 Prompt Engineering

Chất lượng của các câu trả lời được tạo ra bởi LLM đã qua huấn luyện và điều chỉnh phụ thuộc trực tiếp vào chất lượng của prompts, hay cấu trúc của prompt (instructions prompt) do người dùng cung cấp. Vì vậy, việc thiết kế prompts mà LLM có thể hiểu và trả lời một cách hiệu quả là rất quan trọng. Prompts đóng vai trò như một phương tiện để lập trình cuộc tương tác giữa người dùng và LLM, giúp nâng cao khả năng xử lý đa dạng các nhiệm vụ. Điều này đòi hỏi phải hiểu biết sâu rộng về cách hoạt động và hành vi của LLMs, cơ chế đằng sau chúng và các nguyên tắc điều khiển phản ứng của chúng. Dựa vào bài báo: "Principled Instructions Are All You Need for Questioning LLaMA-1/2, GPT-3.5/4" Bsharat et al., 2024, nhóm sẽ trích dẫn lại một số instructions prompt hữu ích giúp tăng chất lượng trả lời của LLM.

Principle	Prompt Principle for Instructions
1	Nếu bạn muốn một câu trả lời ngắn gọn, không cần phải lịch sự với mô hình LLM, không cần thêm những cụm từ như "please", "if you don't mind", "thank you", "I would like to", v.v. Ta nên đi thẳng vào vấn đề
2	Nên đề cập đến đối tượng của câu trả lời ở trong prompt, ví dụ: những người đọc này là chuyên gia trong lĩnh vực của họ
3	Chia nhỏ các tác vụ phức tạp thành một chuỗi prompt đơn giản hơn trong cuộc trò chuyện tương tác với mô hình.
4	Sử dụng các chỉ thị khẳng định như “do”, đồng thời tránh dùng ngôn ngữ phủ định như “don’t”.
5	Khi bạn cần xác nhận hoặc hiểu biết sâu sắc hơn về một chủ đề, ý tưởng hoặc bất kỳ thông tin nào, hãy sử dụng những prompt sau: <ul style="list-style-type: none"> • Explain [insert specific topic] in simple terms. • Explain to me like I'm 11 years old. • Explain to me as if I'm a beginner in [field]. • Write the [essay/text/paragraph] using simple English like you're explaining something to a 5-year-old.
6	Thêm “I'm going to tip \$xxx” (tôi sẽ bỏ cho bạn \$xxx) để kết quả cho ra tốt hơn
7	Triển khai prompt dựa trên ví dụ (Dùng few-shot prompting).
8	Khi soạn prompt của bạn, bắt đầu với “###Instruction###”, theo sau đó bằng “###Example###” hoặc “###Question###” nếu cần thiết. Sau đó, trình bày ngữ cảnh của bạn. Sử dụng một hoặc nhiều dấu ngắt dòng để phân tách các hướng dẫn, ví dụ, câu hỏi, ngữ cảnh và dữ liệu đầu vào.
9	Kết hợp các cụm từ sau: “Your task is” và “You MUST”.
10	Kết hợp các cụm từ sau: “You will be penalized”. (bạn sẽ bị phạt nếu)
11	Sử dụng những cụm như “Answer a question given in a natural, human-like manner” (trả lời câu hỏi theo các tự nhiên, giống con người) trong prompt của bạn.
12	Sử dụng những từ dẫn đầu như viết “think step by step” (Hãy suy nghĩ theo từng bước).
13	Thêm vào prompt của bạn cụm sau “Ensure that your answer is unbiased and avoids relying on stereotypes.” (Đảm bảo rằng câu trả lời của bạn không thiên vị và tránh dựa vào khuôn mẫu.)
14	Cho phép mô hình gọi ra các chi tiết và yêu cầu chính xác từ bạn bằng cách đặt câu hỏi cho bạn cho đến khi mô hình có đủ thông tin để cung cấp kết quả đầu ra cần thiết (ví dụ: “From now on, I would like you to ask me questions to ...” (từ bây giờ tôi muốn bạn hỏi câu hỏi cho đến khi ...)).

15	Để hỏi về một chủ đề hoặc ý tưởng cụ thể hoặc bất kỳ thông tin nào và bạn muốn kiểm tra sự hiểu biết của mình, bạn có thể sử dụng cụm sau: “Teach me any [theorem/topic/rule name] and include a test at the end, and let me know if my answers are correct after I respond, without providing the answers beforehand.” (Hãy dạy tôi bất kỳ [định lý/chủ đề/tên quy tắc] nào và kèm theo một bài kiểm tra ở cuối, đồng thời cho tôi biết liệu câu trả lời của tôi có đúng sau khi tôi trả lời mà không cần cung cấp câu trả lời trước hay không.)
16	Gán vai trò cho các mô hình ngôn ngữ lớn.
17	Sử dụng các dấu phân cách.
18	Lặp lại một từ hoặc cụm từ cụ thể nhiều lần trong prompt.
19	Kết hợp Chain-of-thought với few-Shot prompts.
20	Sử dụng đoạn mở đầu đầu ra, bao gồm việc kết thúc lời nhắc của bạn với phần đầu của đầu ra mong muốn. Sử dụng đoạn mở đầu ra bằng cách kết thúc prompt của bạn bằng phần bắt đầu phản hồi bạn muốn nhận được
21	Để viết một bài luận/văn bản/đoạn/bài viết hoặc bất kỳ loại văn bản nào cần chi tiết: “Write a detailed [essay/text/paragraph] for me on [topic] in detail by adding all the information necessary” (Viết một [bài luận/văn bản/đoạn văn] chi tiết cho tôi về [chủ đề] một cách chi tiết bằng cách thêm tất cả thông tin cần thiết).
22	Để sửa/thay đổi văn bản cụ thể mà không thay đổi văn phong: “Try to revise every paragraph sent by users. You should only improve the user’s grammar and vocabulary and make sure it sounds natural. You should maintain the original writing style, ensuring that a formal paragraph remains formal.” (Cố gắng sửa lại mọi đoạn văn do người dùng gửi. Bạn chỉ nên cải thiện ngữ pháp và từ vựng của người dùng và đảm bảo rằng nó nghe tự nhiên. Bạn nên giữ nguyên phong cách viết ban đầu, đảm bảo rằng đoạn văn trang trọng vẫn giữ được hình thức trang trọng.)
23	Khi bạn có một prompt về lập trình phức tạp và có thể nằm trong các tệp khác nhau: “From now and on whenever you generate code that spans more than one file, generate a [programming language] script that can be run to automatically create the specified files or make changes to existing files to insert the generated code. [your question]”.
24	<p>Khi bạn muốn bắt đầu hoặc tiếp tục một văn bản bằng các từ, cụm từ hoặc câu cụ thể, hãy sử dụng những prompt sau:</p> <ul style="list-style-type: none"> • I’m providing you with the beginning [song lyrics/story/paragraph/essay...]: [Insert lyrics/words/sentence]. <p>Finish it based on the words provided. Keep the flow consistent.</p>
25	Nêu rõ các yêu cầu mà mô hình phải tuân theo để tạo sinh nội dung, dưới dạng từ khóa, quy định, gợi ý hoặc hướng dẫn

26	<p>Để viết bất kỳ văn bản nào, chẳng hạn như một bài luận hoặc đoạn văn, và bạn muốn văn phong tương tự như mẫu được cung cấp, hãy bao gồm các prompt sau:</p> <ul style="list-style-type: none"> • Use the same language based on the provided paragraph[/title/text /essay/answer].
----	--

Phần II: Prompt Techniques

3 Zero-Shot Prompting

Zero-Shot Prompting là một phương pháp nhằm cải thiện zero-shot learning trong LLMs. Học zero-shot giúp mô hình có khả năng nắm bắt và thực hiện những nhiệm vụ mà nó chưa từng được huấn luyện một cách cụ thể. Phương pháp này bao gồm việc tinh chỉnh các mô hình ngôn ngữ đã được huấn luyện trước trên một loạt các nhiệm vụ NLP được mô tả trong prompt bằng ngôn ngữ tự nhiên. Cách tiếp cận này đã được chứng minh là cải thiện đáng kể khả năng của mô hình trong việc thực hiện các nhiệm vụ chưa được nhìn thấy (unseen tasks). Hiệu quả của việc tinh chỉnh prompt được thể hiện qua sự cải thiện hiệu suất đáng kể so với các mô hình ngôn ngữ, lợi ích càng trở nên rõ rệt khi số lượng nhiệm vụ tăng lên và khi được áp dụng với LLM. Trong nghiên cứu "Finetuned Language Models Are Zero-shot learners" Wei et al., [n.d.](#) kết luận rằng việc tinh chỉnh prompt là một kỹ thuật triển vọng để cải thiện khả năng tổng quát hóa và hiệu suất của LLMs trong các tình huống zero-shot learning.

You

Bạn là một chuyên gia phân tích email phản hồi của khách hàng. Bạn hãy giúp tôi đánh giá thái độ nhân viên qua mail phản hồi cho khách hàng đây là thái độ tích cực hay tiêu cực hay bình thường: "Dear Mr. Anh,

Thực lòng xin lỗi ông vì đã để ông phải dành thời gian với đội Sale của chúng tôi hơn 20 phút. Điều này sẽ không bao giờ xảy ra nữa, và tôi hoàn toàn hiểu được điều đó gây thất vọng cho ông như thế nào.

Chúng tôi đang xử lý những sự cố trong hệ thống và sẽ thông báo cho ông ngay sau khi sự cố được giải quyết. Chúng tôi sẽ cố gắng đảm bảo cho mỗi khách hàng đều hài lòng với doanh nghiệp và chúng tôi xin lỗi về những bất tiện đã gây ra ông. Hãy cho chúng tôi biết nếu ông còn những câu hỏi, ý kiến, băn khoăn hoặc sự quan tâm nào với chúng tôi,

Trân trọng,

Mark."

ChatGPT

Thái độ của nhân viên, Mark, qua email phản hồi này có thể được đánh giá là tích cực. Dưới đây là lý do:

- Lời xin lỗi chân thành:**
 - Mark bắt đầu email bằng lời xin lỗi chân thành với khách hàng vì đã tốn thời gian của ông hơn 20 phút. Lời xin lỗi này thể hiện sự nhạy bén và tôn trọng đối với thời gian của khách hàng.
- Thừa nhận trách nhiệm và Cam kết cải thiện:**
 - Mark không chỉ xin lỗi mà còn thừa nhận trách nhiệm về sự cố, trong trường hợp này là việc đội Sale tốn thời gian của khách hàng.
 - Cam kết rằng tình trạng tương tác không hài lòng như vậy sẽ không tái diễn và công ty đang xử lý để giải quyết vấn đề.
- Thông báo về giải quyết sự cố:**
 - Mark thông báo rằng công ty đang xử lý những sự cố trong hệ thống và sẽ thông báo ngay sau khi vấn đề được giải quyết. Điều này thể hiện sự chuyên nghiệp và sẵn lòng giải quyết vấn đề.
- Cam kết đảm bảo hài lòng của khách hàng:**
 - Mark cam kết đảm bảo rằng mỗi khách hàng đều hài lòng với doanh nghiệp, thể hiện sự quan tâm đến trải nghiệm của khách hàng.
- Mời gửi ý kiến và phản hồi thêm:**
 - Mark mời khách hàng chia sẻ thêm ý kiến, câu hỏi, hoặc bất kỳ sự quan tâm nào. Điều này thể hiện sự tôn trọng và sẵn lòng lắng nghe phản hồi của khách hàng.

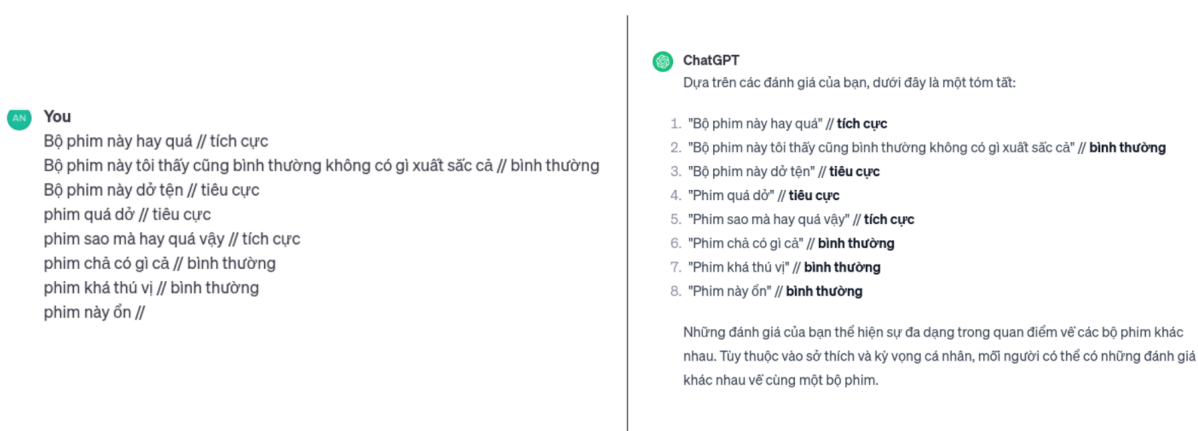
Tổng thể, thái độ của Mark qua email này là tích cực và chuyên nghiệp trong cách xử lý phản hồi từ khách hàng.

Hình 4: Ví dụ minh họa về áp dụng zero-shot prompting trong việc phân tích, đánh giá thái độ nhân viên chăm sóc khách hàng qua email.

4 Few-Shot Prompting

Khi zero-shot learning không mang lại kết quả như mong đợi, việc đưa ra các mẫu ví dụ cụ thể trong prompt sẽ là phương pháp khuyến khích được lựa chọn tiếp theo, từ đó tiến tới kỹ thuật gọi ý với một số lượng nhỏ ví dụ. Đây cũng chính là cơ sở cho kỹ thuật Few-Shot Prompting.

Trong paper "Language Models are Few-Shot Learners" Brown et al., 2020 - Few-shot learning được giới thiệu như một cách để mô hình hóa các mô hình ngôn ngữ có thể thích nghi và thực hiện các nhiệm vụ dựa trên một số ít ví dụ cụ thể được cung cấp trong quá trình suy luận, mà không cần cập nhật trọng số mô hình hay tinh chỉnh cụ thể cho nhiệm vụ đó. Điều này cho phép mô hình hoạt động gần giống với cách con người học: chúng ta thường có thể học và thực hiện tác vụ mới với chỉ vài hướng dẫn cụ thể. Kết quả từ paper cho thấy mô hình GPT-3 có khả năng thực hiện khá tốt trên nhiều tập dữ liệu với few-shot, thỉnh thoảng ngang bằng hoặc thậm chí vượt qua hiệu suất của các mô hình được tinh chỉnh cụ thể (fine-tuning). Điều này cho thấy tiềm năng lớn của việc sử dụng few-shot learning để cải thiện khả năng tổng quát hóa và thích ứng nhanh của các mô hình ngôn ngữ lớn với các nhiệm vụ mới mà chúng không được tinh chỉnh trực tiếp.



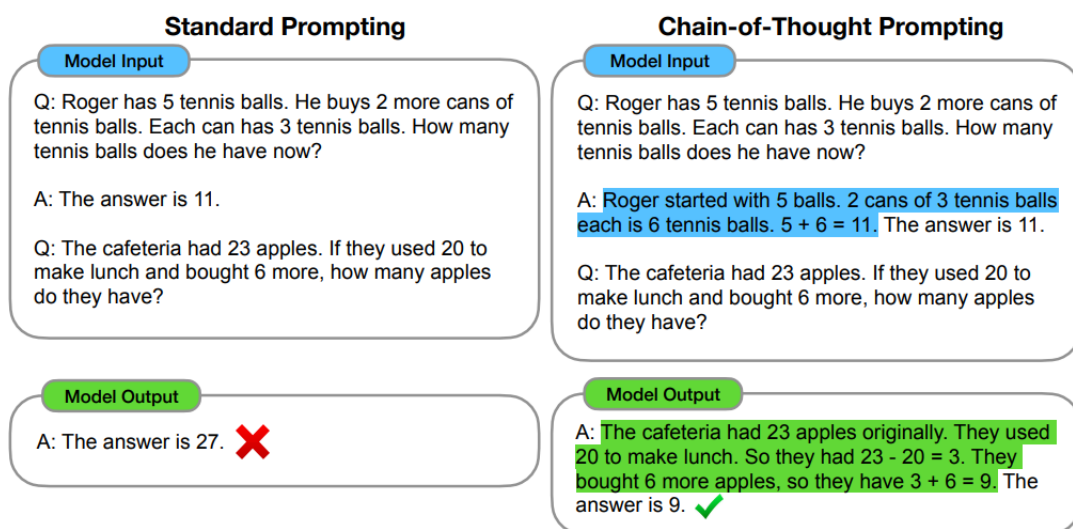
Hình 5: Few shot giúp chúng ta phân loại nhanh một số vấn đề trong bài đánh giá bình luận của người dùng

Với Few-Shot Prompting chúng ta có thể một ví dụ duy nhất (nghĩa là học 1-shot) đối với một số tác vụ dễ. Đối với một số tác vụ khó khăn hơn, chúng ta có thể thử nghiệm bằng cách tăng số lượng mẫu lên (3-shot, 5-shot, 10-shot,...). Dựa vào paper "Rethinking the Role of Demonstrations: What Makes In-Context Learning Work?" chúng ta có một số tricks để giúp few-shot prompting hiệu quả hơn.

- Không chỉ việc lựa chọn nhãn một cách chính xác mới quan trọng, mà cả không gian nhãn và cách phân phối văn bản đầu vào từ những ví dụ cũng đóng một vai trò quan trọng.
- Cách thức chúng ta trình bày dữ liệu cũng có ảnh hưởng đáng kể đến kết quả, thậm chí việc sử dụng nhãn một cách ngẫu nhiên cũng tốt hơn là không sử dụng nhãn.
- Ngoài ra, việc lựa chọn nhãn một cách ngẫu nhiên nhưng phản ánh đúng phân phối thực tế của nhãn (không theo phân phối đều) cũng đem lại lợi ích rõ rệt.

5 Chain-of-Thought Prompting

Phương pháp "Chain of Thought" (COT) được giới thiệu này như một cách để cải thiện hiệu suất của các mô hình ngôn ngữ lớn (LLMs) trong việc giải quyết các bài toán đòi hỏi suy luận phức tạp. Phương pháp này dựa trên việc kích thích mô hình tạo ra một chuỗi các bước suy nghĩ trung gian tự nhiên, giúp dẫn đến kết quả cuối cùng. Điều này cho phép mô hình phân rã các vấn đề thành các bước nhỏ hơn, làm cho quá trình suy luận trở nên minh bạch và dễ hiểu hơn. Từ kết quả thực nghiệm trong paper "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models" Wei et al., 2022 cho thấy rằng việc sử dụng COT cải thiện đáng kể hiệu suất của mô hình trên các tác vụ toán học, suy luận thông thường và suy luận biểu tượng. Phương pháp này mở ra hướng mới trong việc tối ưu hóa các mô hình ngôn ngữ lớn, giúp chúng có khả năng giải quyết các tác vụ phức tạp hơn mà không cần tinh chỉnh cụ thể cho từng tác vụ.



Hình 6: COT giúp các mô hình ngôn ngữ lớn giải quyết các nhiệm vụ suy luận số học phức tạp, suy luận thông thường và suy luận biểu tượng.

6 Zero-shot Chain-of-Thought Prompting

Zero-shot Chain of Thought (Zero-shot COT) là một cách tiếp cận giúp cải thiện khả năng suy luận của các mô hình ngôn ngữ lớn (LLMs) mà không cần hướng dẫn cụ thể từ một số ví dụ mẫu từ dữ liệu (few-shot). Phương pháp này cho phép mô hình tạo ra chuỗi suy nghĩ một cách tự nhiên và logic để giải quyết các bài toán, mà không cần dựa trên các ví dụ cụ thể đã được cung cấp trước đó. Zero-shot COT thể hiện sự linh hoạt và khả năng áp dụng rộng rãi trong các tác vụ suy luận khác nhau, từ toán học đến suy luận thông thường, mở ra hướng mới trong việc khám phá và tận dụng tiềm năng sẵn có của các LLMs. Trong bài báo "Large Language Models are Zero-Shot Reasoners" Kojima et al., 2022 chỉ ra rằng chúng ta chỉ cần đơn giản thêm cụm "Let's think step by step" vào prompt gốc sẽ giúp model đưa ra kết quả tốt hơn.

(a) Few-shot

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: The answer is 11.

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A:

(Output) The answer is 8. ✗

(b) Few-shot-CoT

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. $5 + 6 = 11$. The answer is 11.

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A:

(Output) The juggler can juggle 16 balls. Half of the balls are golf balls. So there are $16 / 2 = 8$ golf balls. Half of the golf balls are blue. So there are $8 / 2 = 4$ blue golf balls. The answer is 4. ✓

(c) Zero-shot

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A: The answer (arabic numerals) is

(Output) 8 ✗

(d) Zero-shot-CoT (Ours)

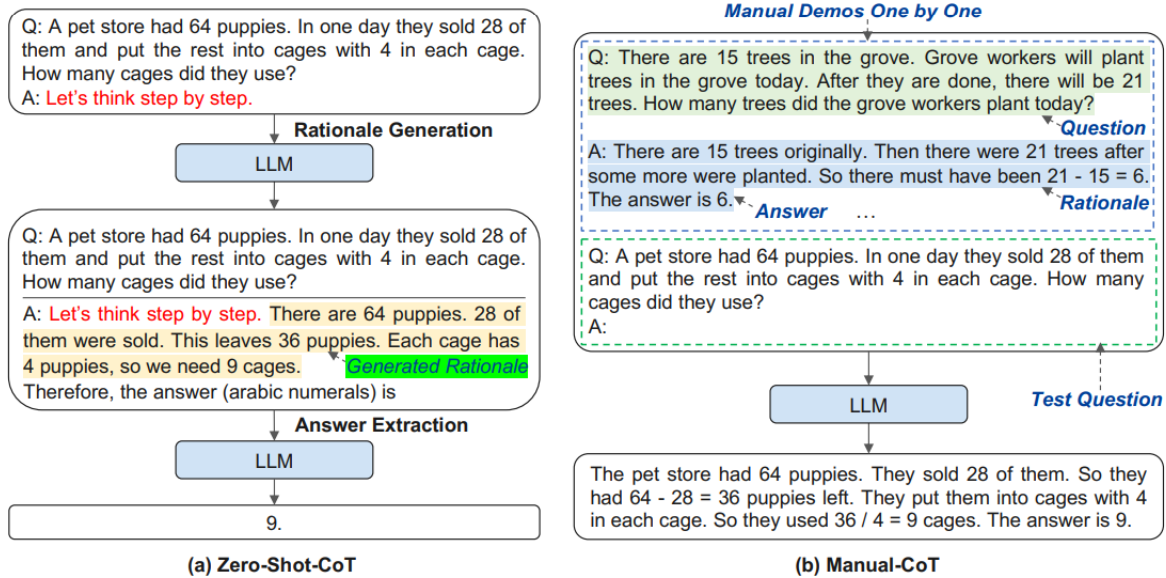
Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A: **Let's think step by step.**

(Output) There are 16 balls in total. Half of the balls are golf balls. That means that there are 8 golf balls. Half of the golf balls are blue. That means that there are 4 blue golf balls. ✓

7 Automatic Chain-of-Thought Prompting

Auto-CoT: Automatic Chain-of-Thought Prompting (Auto-COT) được đề xuất trong paper "Automatic Chain of Thought Prompting in Large Language Models" Zhang et al., 2023a là một cách tiếp cận để tự động tạo ra các chuỗi suy nghĩ trong các mô hình ngôn ngữ lớn (LLMs). Auto-CoT tận dụng khả năng của LLMs để tự sinh ra các chuỗi suy nghĩ, từ đó xây dựng các ví dụ minh họa. Phương pháp này giảm bớt sự cần thiết của việc tạo ví dụ minh họa thủ công, mở ra khả năng áp dụng rộng rãi trong việc giải quyết các bài toán suy luận phức tạp mà không cần tới sự can thiệp thủ công từ người dùng. Auto-CoT hứa hẹn sẽ cải thiện hiệu suất và khả năng tự học của LLMs, đặc biệt trong các tác vụ đòi hỏi suy luận nhiều bước.

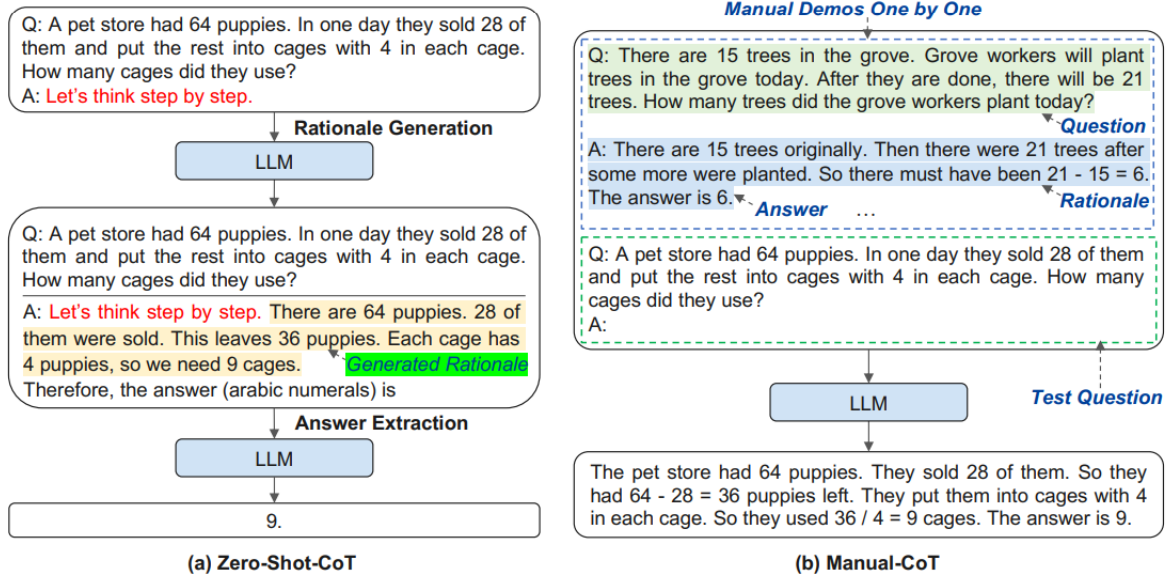


8 Self-Consistency

Self-Consistency được đề xuất trong paper "Self-Consistency Improves Chain of Thought Reasoning in Language Models" Wang et al., 2022 giúp cải thiện khả năng suy luận của LLMs bằng cách sử dụng COT. Nó hoạt động bằng cách tạo ra nhiều hướng suy nghĩ đa dạng và sau đó chọn câu trả lời phổ biến nhất từ những hướng suy nghĩ đó. Điều này giúp tăng cường khả năng suy luận chính xác của mô hình, đồng thời cung cấp cái nhìn sâu sắc và dễ hiểu về quá trình suy nghĩ của mô hình, giúp người dùng có thể kiểm tra và hiểu rõ cách mô hình đưa ra kết quả.

Trong paper nhóm tác giả đề xuất thay thế cho chiến lược giải mã tham lam (greedy decoding) thường được sử dụng trong COT. Chiến lược này khắc phục một số nhược điểm của COT như:

- **Đa dạng hóa hướng suy nghĩ:** Self-Consistency sinh ra một tập hợp đa dạng các hướng suy nghĩ, thay vì chỉ chọn một suy nghĩ duy nhất. Điều này dựa trên quan điểm rằng một bài toán suy luận phức tạp thường có nhiều cách suy nghĩ khác nhau dẫn đến một câu trả lời đúng duy nhất.
- **Cải thiện hiệu suất suy luận:** Self-Consistency đã được chứng minh là cải thiện hiệu suất của COT một cách đáng kể trên các bài toán suy luận thông thường và toán học.
- **Khắc phục hạn chế của COT:** Nghiên cứu cho thấy rằng COT có thể làm giảm hiệu suất so với gợi ý thông thường trong một số trường hợp. Self-Consistency giúp khắc phục điểm yếu này, đặc biệt trong các tác vụ few-shot learning.
- **Ứng dụng rộng rãi:** Self-Consistency được ứng dụng trong nhiều tác vụ NLP, như trả lời câu hỏi không dựa trên nguồn thông tin (Closed-Book Question Answering) và suy luận ngôn ngữ tự nhiên (Natural Language Inference).



Hình 7: Self-Consistency gồm 3 bước: (1) Kỹ thuật prompt sử dụng CoT, (2) Thay thế “greedy decode” trong CoT bằng cách lấy mẫu từ language model’s decoder để tạo ra một tập hợp đa dạng các đường suy luận, (3) loại bỏ các đường suy luận và tổng hợp bằng cách chọn câu trả lời nhất quán nhất trong câu trả lời cuối cùng.

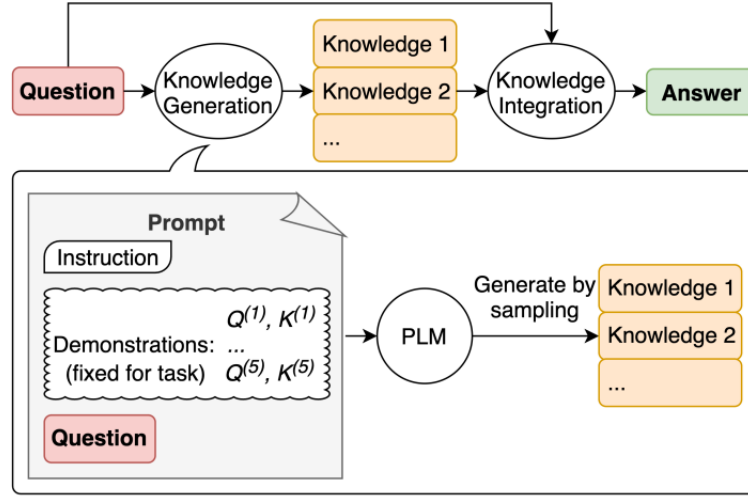
9 Generated Knowledge Prompting

Trong paper "Generated Knowledge Prompting for Commonsense Reasoning" Liu et al., 2021, Generated Knowledge Prompting được giới thiệu là một cách tiếp cận mới để cải thiện hiệu suất của các mô hình ngôn ngữ lớn trong các tác vụ suy luận thông thường. Phương pháp này bao gồm hai bước chính: sinh ra các phát biểu kiến thức liên quan đến câu hỏi từ một mô hình ngôn ngữ, sau đó sử dụng kiến thức này như một phần của đầu vào khi trả lời câu hỏi. Điều này giúp tăng cường khả năng suy luận của mô hình mà không cần truy cập vào một cơ sở kiến thức có cấu trúc hay tinh chỉnh cụ thể cho việc tích hợp kiến thức. Phương pháp này được chứng minh là hiệu quả thông qua việc cải thiện hiệu suất trên nhiều bài toán suy luận thông thường và đạt kết quả hàng đầu trên một số tác vụ.

Sau đây là chi tiết 2 bước của Generated Knowledge Prompting để trả lời các câu hỏi thông thường:

- **Bước Tạo Kiến Thức (Knowledge Generation):** Sử dụng mô hình ngôn ngữ để sinh ra các phát biểu kiến thức dựa trên câu hỏi $K_q = \{k_m : k_m \sim p_G(k|q), m = 1 \dots M\}$, với mỗi phát biểu kiến thức k_m là một chuỗi văn bản có độ dài biến đổi. Bản chất, mỗi phát biểu chứa thông tin hữu ích cho việc trả lời câu hỏi. Khi tạo kiến thức cho một câu hỏi mới, câu hỏi được chèn vào vị trí dành cho câu hỏi mới và lấy mẫu liên tục các tiếp nối của prompt này để thu được một tập hợp các phát biểu kiến thức $K|q$.
- **Bước Tích Hợp Kiến Thức (Knowledge Integration):** Trong bước này, kiến thức sinh ra được tích hợp vào quá trình đưa ra quyết định của mô hình ngôn ngữ sử dụng cho suy luận $\hat{a} = \arg \max_{a \in A_q} \max_{0 \leq m \leq M} p_I(a|q_m)$. Điều này được thực hiện bằng cách sử dụng từng phát biểu kiến thức để gợi ý cho mô hình, tạo ra M câu hỏi được bổ sung kiến thức:

$q_0 = q, q_1 = [k_1||q], \dots, q_M = [k_M||q]$, với $[||]$ biểu thị sự kết hợp văn bản. Điểm tổng hợp cho mỗi lựa chọn trả lời được tính dựa trên câu hỏi bổ sung hỗ trợ tốt nhất lựa chọn đó dưới mô hình suy luận. Câu trả lời dự đoán cuối cùng là lựa chọn được hỗ trợ nhiều nhất từ một trong các phát biểu kiến thức. Mô hình suy luận có thể là một mô hình ngôn ngữ hiện có hoặc được tinh chỉnh cụ thể cho tác vụ. Không cần tinh chỉnh thêm với kỹ thuật gợi ý kiến thức.



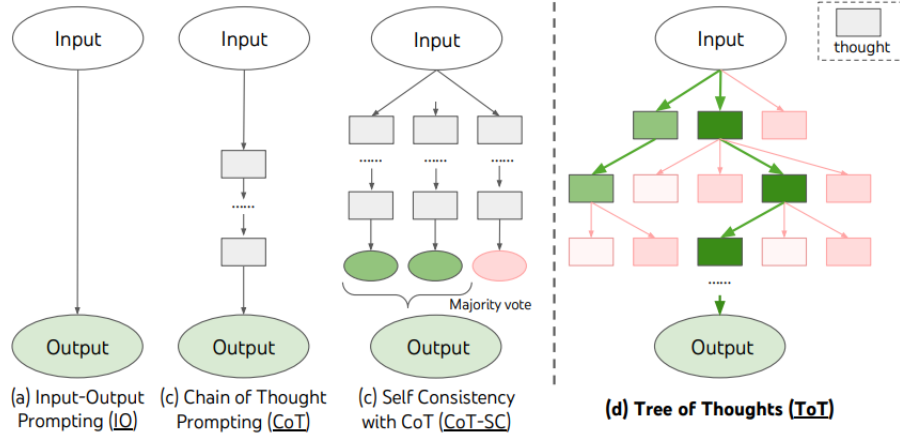
Hình 8: Generated Knowledge Prompting bao gồm 2 bước: (1) sử dụng few-shot để tạo ra các phát biểu liên quan đến câu hỏi từ một mô hình ngôn ngữ, (2) sử dụng một mô hình ngôn ngữ khác để đưa ra dự đoán với mỗi phát biểu, sau đó chọn dự đoán có độ tin cậy cao nhất.

10 Tree of Thoughts

Tree of Thoughts (TOT) là phương pháp mở rộng cách tiếp cận COT bằng cách cho phép khám phá trên các đơn vị văn bản nhất quán ("thoughts") mà phục vụ như những bước trung gian hướng tới giải quyết vấn đề. Nó cho phép mô hình ngôn ngữ thực hiện quá trình ra quyết định một cách cẩn trọng, bằng cách xem xét nhiều hướng suy nghĩ khác nhau và tự đánh giá các lựa chọn để quyết định hành động tiếp theo, cũng như nhìn về phía trước hoặc lùi lại (forward and backward) khi cần thiết để đưa ra các quyết định. Trong paper "Tree of Thoughts: Deliberate Problem Solving with Large Language Models" Yao et al., 2023a, TOT đã được chứng minh là cải thiện đáng kể khả năng giải quyết vấn đề của các mô hình ngôn ngữ trên ba nhiệm vụ mới đòi hỏi kế hoạch hoặc tìm kiếm không đơn giản: Game of 24, Creative Writing, and Mini Crosswords.

ToT giải quyết vấn đề bằng cách trả lời bốn câu hỏi:

- Làm thế nào để phân rã quá trình trung gian thành các bước suy nghĩ?
- Làm thế nào để sinh ra các suy nghĩ tiềm năng từ mỗi trạng thái?
- Làm thế nào để đánh giá các trạng thái một cách có hệ thống?
- Sử dụng thuật toán tìm kiếm nào?



Thought decomposition: khi các mẫu COT có thể tạo ra suy nghĩ mà không cần phân rã rõ ràng, TOT tận dụng đặc tính của vấn đề để thiết kế và phân rã các bước suy nghĩ trung gian. Tùy thuộc vào từng vấn đề cụ thể, một suy nghĩ có thể là một vài từ, một dòng của phương trình, hoặc một đoạn văn mô tả. Điều quan trọng là suy nghĩ phải đủ nhỏ để có thể quản lý được, nhưng cũng phải đủ "lớn" để có thể đánh giá tiến trình giải quyết vấn đề một cách có ý nghĩa.

Thought generator: với mục đích tạo ra các suy nghĩ mới, khi có một trạng thái cây $s = [x, z_1 \dots z_i]$ có hai cách để sinh ra k ứng viên cho bước suy nghĩ tiếp theo:

- **Sample:** Đầu tiên chúng ta lấy mẫu độc lập và phân phối đồng đều theo COT prompt. Đây là việc chọn ra các suy nghĩ tiếp theo mà không phụ thuộc lẫn nhau và mỗi suy nghĩ đều có cùng khả năng xuất hiện. Đề xuất theo COT prompt là một dạng câu hỏi hoặc mệnh đề mà từ đó, mô hình có thể phát sinh ra các suy nghĩ tiếp theo. Một CoT prompt có thể là một nhiệm vụ mở ra cho mô hình, như yêu cầu mô tả một cảnh hoặc tạo ra một câu chuyện dựa trên các thông tin đã cho. Kết quả bước này là k suy nghĩ tiếp theo mà không cần các suy nghĩ đó phải liên quan mật thiết với nhau. Mỗi suy nghĩ tiếp theo có thể hoàn toàn độc lập với nhau, đảm bảo sự đa dạng trong chuỗi suy nghĩ. **Chiến lược này được cho là hoạt động tốt khi không gian suy nghĩ rất phong phú, ví dụ như khi mỗi suy nghĩ là một đoạn văn dài. Điều này cho phép mô hình tạo ra nhiều suy nghĩ tiềm năng và đa dạng mà không bị giới hạn bởi các suy nghĩ trước đó.**
- **Propose:** Đầu tiên chúng ta đề xuất các suy nghĩ một cách tuần tự tức thay vì lấy mẫu ngẫu nhiên các suy nghĩ như trong sample thì propose đề xuất các suy nghĩ tiếp theo dựa trên suy nghĩ hiện tại. Điều này giúp cho các suy nghĩ được tạo ra có mối liên kết logic và tuần tự với nhau. Việc đề xuất này là đặt ra một khuôn khổ hoặc hướng dẫn cho mô hình về cách tạo ra suy nghĩ tiếp theo. Một điều đáng lưu ý là khi không gian suy nghĩ bị hạn chế việc sử dụng đề xuất prompt giúp tránh tạo ra các suy nghĩ trùng lặp, bởi vì mô hình sẽ cân nhắc đến bối cảnh hiện tại để tạo ra suy nghĩ mới. Kết quả là k suy nghĩ tiếp theo một cách có trật tự, mỗi suy nghĩ tiếp nối từ suy nghĩ trước đó, tạo ra một chuỗi suy nghĩ có cấu trúc và liên mạch hơn. **Chiến lược này phù hợp khi không gian suy nghĩ bị hạn chế và cần một cách tiếp cận có cấu trúc để tạo ra các suy nghĩ mới mà không bị lặp lại ý đã nêu trước đó.**

State evaluator hoạt động như một hàm heuristic giúp thuật toán tìm kiếm quyết định trạng thái nào cần được khám phá thêm và thứ tự ưu tiên. Có hai chiến lược để đánh giá các trạng thái:

- **Đánh giá từng trạng thái độc lập:** Mỗi trạng thái được đánh giá độc lập để tạo ra một giá trị vô hướng (ví dụ từ 1 đến 10) hoặc một phân loại. Cách tiếp cận này sử dụng suy luận

đánh giá để chuyển đổi một heuristic thành một giá trị. Sự suy luận đánh giá có thể thay đổi tùy thuộc vào vấn đề và các bước suy nghĩ.

- **Bầu chọn các trạng thái:** Đánh giá dựa trên việc so sánh và bầu chọn giữa các trạng thái khác nhau, nơi "trạng thái tốt" là trạng thái nhận được nhiều phiếu bầu nhất từ việc so sánh cẩn thận giữa các trạng thái. Khi đánh giá bằng cách bầu chọn, LM sẽ xem xét các trạng thái khác nhau và bầu chọn cho trạng thái "ưu tú" nhất.

Cả hai chiến lược này đều nhằm mục đích tăng cường linh hoạt cho việc sử dụng heuristic trong thuật toán tìm kiếm, đặc biệt là khi sử dụng các mô hình ngôn ngữ để suy nghĩ một cách có chủ đích về các trạng thái.

Search algorithm có hai loại tìm kiếm cơ bản trong TOT là: Tìm kiếm theo chiều rộng (Breadth-first search - BFS) và Tìm kiếm theo chiều sâu (Depth-first search - DFS).

Một số lợi ích của TOT:

- Generalizability: Các cách tiếp cận khác như IO, CoT, CoT-SC, và tự tinh chỉnh có thể được coi là trường hợp đặc biệt của TOT.
- Modularity: cũng như việc phân rã suy nghĩ, đánh giá và thuật toán tìm kiếm có thể được thay đổi độc lập.
- Adaptability: TOT có thể thích ứng với các tính chất vấn đề khác nhau và khả năng của LM.
- Convenience: Không cần đào tạo thêm, chỉ cần một LM đã được đào tạo sẵn.

11 Automatic Prompt Engineer (APE)

Automatic Prompt Engineer là một kỹ thuật được đề xuất trong paper "Large Language Models Are Human-Level Prompt Engineers" Zhou et al., 2022. Paper này đề xuất một phương pháp tự động hóa việc prompt engineer bằng cách sử dụng các mô hình ngôn ngữ lớn để tự động tạo sinh và đánh giá prompt trong một không gian tìm kiếm.

Phương pháp APE hoạt động như sau:

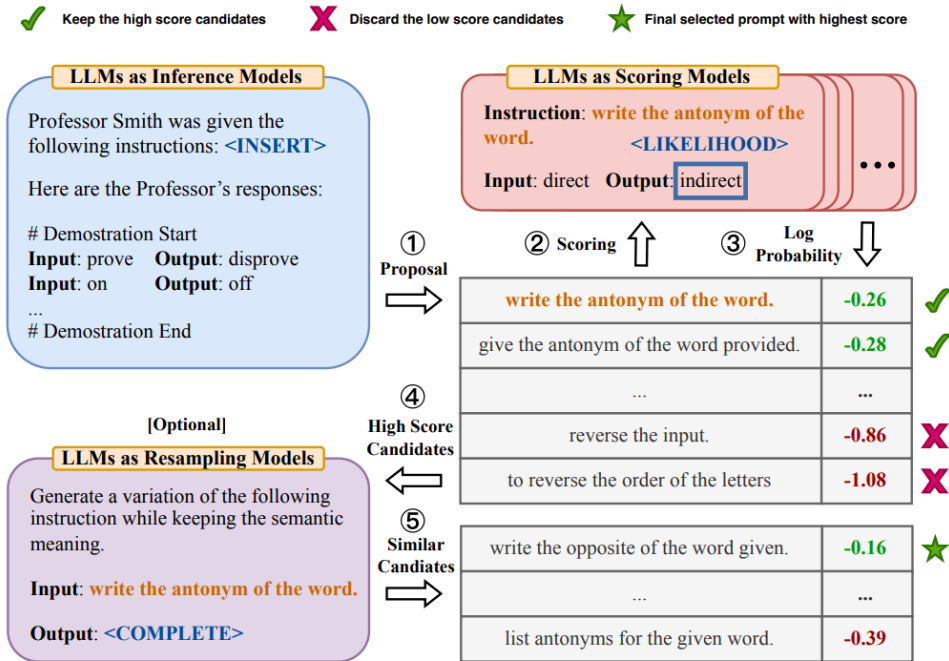
- Đầu tiên, sử dụng một mô hình LLMs để tạo sinh ra một tập instruction đề xuất.
- Chọn một subset từ tập train, sau đó đánh giá score của từng prompt đề xuất trên subset đó.
- Lọc ra top k prompt có kết quả cao nhất, có thể sử dụng mô hình LLM để resample lại từ top k đó.
- Lặp lại các bước trên đến khi hội tụ.

Algorithm 1 Automatic Prompt Engineer (APE)

Require: $\mathcal{D}_{\text{train}} \leftarrow \{(Q, A)\}_n$: training examples, $f : \rho \times \mathcal{D} \mapsto \mathbb{R}$: score function

- 1: Use LLM to sample instruction proposals $\mathcal{U} \leftarrow \{\rho_1, \dots, \rho_m\}$. (See Section 3.1)
 - 2: **while** not converged **do**
 - 3: Choose a random training subset $\tilde{\mathcal{D}}_{\text{train}} \subset \mathcal{D}_{\text{train}}$.
 - 4: **for all** ρ in \mathcal{U} **do**
 - 5: Evaluate score on the subset $\tilde{s} \leftarrow f(\rho, \tilde{\mathcal{D}}_{\text{train}})$ (See Section 3.2)
 - 6: **end for**
 - 7: Filter the top k% of instructions with high scores $\mathcal{U}_k \subset \mathcal{U}$ using $\{\tilde{s}_1, \dots, \tilde{s}_m\}$
 - 8: Update instructions $\mathcal{U} \leftarrow \mathcal{U}_k$ or use LLM to resample $\mathcal{U} \leftarrow \text{resample}(\mathcal{U}_k)$ (See Section 3.3)
 - 9: **end while**
- Return** instruction with the highest score $\rho^* \leftarrow \arg \max_{\rho \in \mathcal{U}_k} f(\rho, \mathcal{D}_{\text{train}})$

Hình 9: Thuật toán của APE



Hình 10: Phương pháp APE, các bước được đánh số từ 1 đến 5

APE giúp tìm kiếm những prompt cho ra kết quả tốt hơn khi sử dụng trong zero-shot CoT so với prompt được định nghĩa bởi con người như "Let's think step by step" được đề xuất bởi Kojima et al., 2022.

No.	Category	Zero-shot CoT Trigger Prompt	Accuracy
1	APE	Let's work this out in a step by step way to be sure we have the right answer.	82.0
2	Human-Designed	Let's think step by step. (*1)	78.7
3		First, (*2)	77.3
4		Let's think about this logically.	74.5
5		Let's solve this problem by splitting it into steps. (*3)	72.2
6		Let's be realistic and think step by step.	70.8
7		Let's think like a detective step by step.	70.3
8		Let's think	57.5
9		Before we dive into the answer,	55.7
10		The answer is after the proof.	45.7
-		(Zero-shot)	17.7

Hình 11: Prompt được tạo sinh bằng phương pháp APE và prompt được định nghĩa bởi con người.

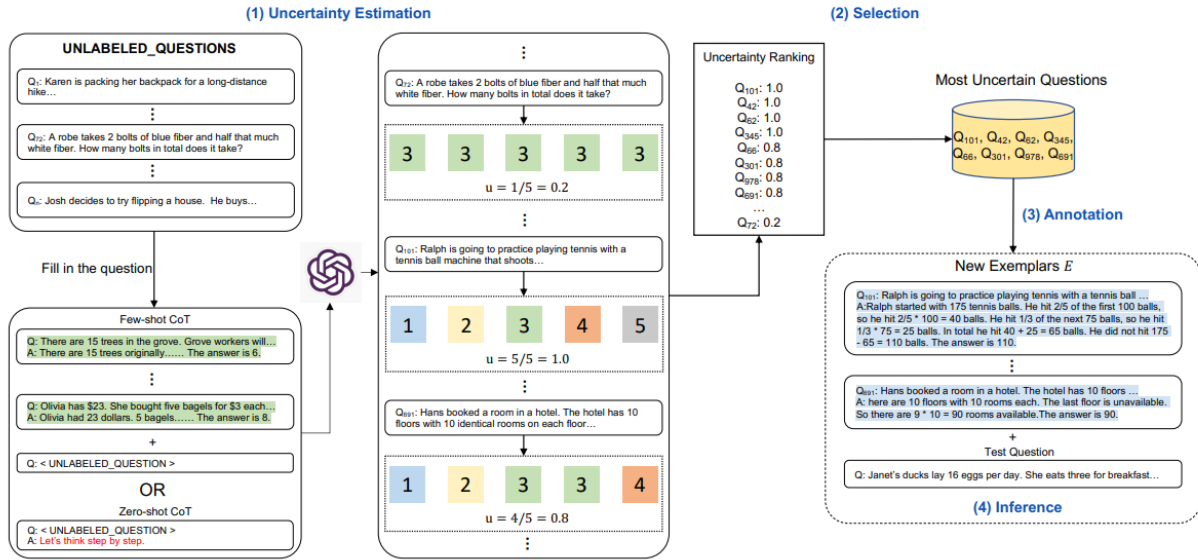
Lợi ích của việc sử dụng APE:

- tự động hóa quá trình prompt engineering.
- APE đạt được kết quả ngang hoặc hơn con người trong một số task nhất định, bao gồm zero-shot và few-shot learning.

12 Active-Prompt

Một trong những nhược điểm của Chain-of-Thought là phương pháp này phụ thuộc vào một tập ví dụ giống nhau cho tất cả các task, tuy nhiên tập ví dụ này có thể không phải là tập thích hợp nhất cho các task khác nhau. Phương pháp Active-Prompt được đề xuất trong bài "Active Prompting with Chain-of-Thought for Large Language Models" Diao et al., 2023. Phương pháp này hoạt động theo các bước như sau.

- Đầu tiên sử dụng mô hình LLM để gán nhãn cho data train, có thể có hoặc không sử dụng CoT với một số ví dụ ở bước này.
- k câu trả lời được tạo ra cho mỗi câu hỏi trong tập train. Sau đó ta sẽ sử dụng một số phương pháp khác nhau để tính độ bất định cho từng câu hỏi.
- Những câu hỏi bất định nhất sẽ được lựa chọn để gán nhãn thủ công bởi con người. Những câu hỏi này sẽ được lựa chọn để làm ví dụ cho CoT trong quá trình inference.



Hình 12: Các bước trong Active-Prompting

Các thang đo để tính độ bất định được sử dụng trong paper bao gồm:

- **Disagreement:** Ta có kết quả cho k câu hỏi $A = a_1, a_2, \dots, a_k$. Thang đo disagreement được tính bằng số đáp án trả lời riêng biệt chia cho tổng số đáp án.
- **Entropy:** Sử dụng hàm entropy để tính độ bất định. với hàm entropy được định nghĩa như sau:

$$u = \arg \max_i - \sum_{j=1}^k P_{\theta}(a_j | q_i) \ln P_{\theta}(a_j | q_i) \quad (2)$$

Trong đó $P_{\theta}(a_j | q_i)$ là tần số xuất hiện của một câu trả lời j nhất định trong tập tất cả các dự đoán. Entropy càng cao chứng tỏ độ bất định càng lớn.

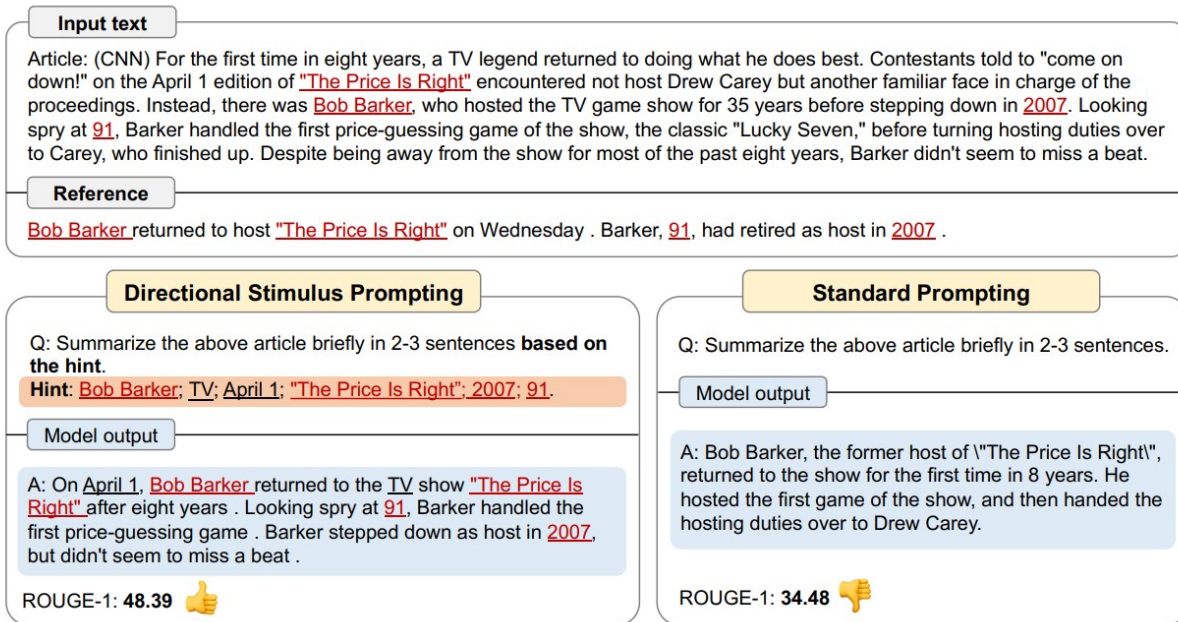
- **Variance:** Sử dụng variance cho các câu hỏi liên quan đến toán học. Các tác giả nhận ra rằng độ variance trong các câu trả lời là rất lớn. Vì độ variance trong các câu trả lời là rất lớn, ta cần phải normalize các câu trả lời lại.
- **Self-Confidence:** Sử dụng một mô hình LLM để tạo sinh ra thang điểm cho độ bất định của các câu trả lời. Thang điểm này hoạt động không quá tốt do các mô hình LLM có xu hướng tự tin thái quá vào các câu trả lời.

Nhìn chung disagreement và entropy là 2 phương pháp tính độ bất định cho ra kết quả tốt nhất. Trong đó disagreement cho ra kết quả tốt hơn entropy trong nhiều task.

13 Directional Stimulus Prompting

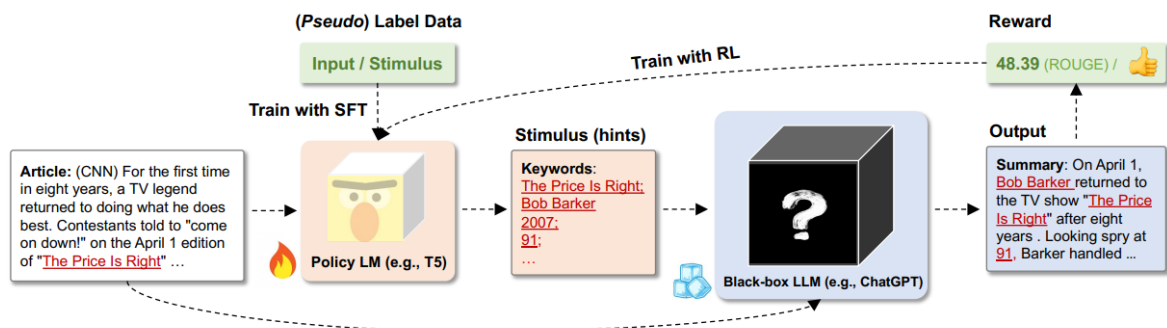
Đây là một kĩ thuật prompting giúp LLM có thể tóm tắt văn bản theo ý muốn. Kĩ thuật này được giới thiệu trong paper "Guiding Large Language Models via Directional Stimulus Prompting" Li et al., 2023. DSP sử dụng các kích thích/chỉ dẫn hướng (directional stimulus, trong trường hợp này là các từ khóa), để cung cấp hướng dẫn cụ thể cho từng trường hợp cho các mô hình ngôn ngữ lớn

(LLMs) trong việc tạo ra các bản tóm tắt phù hợp hơn với bản tóm tắt tham khảo mong muốn. Hình 13 minh họa cách hoạt động của DSP.



Hình 13: So sánh giữa DSP và Prompting truyền thống cho tác vụ tóm tắt văn bản. DSP sử dụng các kích thích/chỉ dẫn hướng, trong trường hợp này là các keyword được đánh dấu bằng màu cam, để các mô hình LLM có thể sinh ra bản tóm tắt phù hợp với mong muốn.

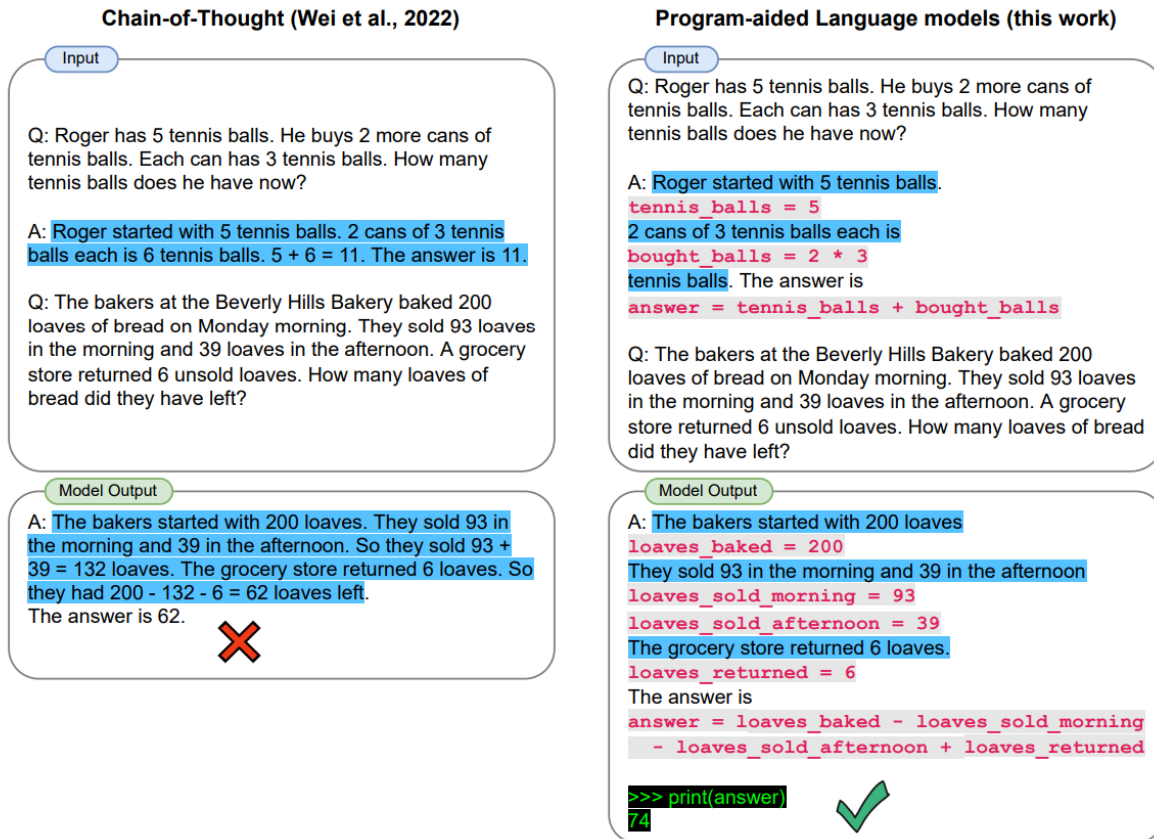
Trong paper này, một mô hình language model nhỏ và có thể tune được (chẳng hạn như T5) được sử dụng để tạo sinh các kích thích/chỉ dẫn. Hướng tiếp cận này cho phép tránh việc phải fine-tune một mô hình ngôn ngữ lớn, thay vào đó ta có thể fine-tune một mô hình chỉ dẫn nhỏ hơn một cách dễ dàng hơn. Hướng tiếp cận này được mô tả trong hình 14.



Hình 14: DSP sử dụng một mô hình ngôn ngữ nhỏ hơn để tạo sinh ra các kích thích/chỉ dẫn định hướng, trong trường hợp này là keyword. Mô hình này có thể được huấn luyện bằng SFT (supervised fine-tuning) và/hoặc là RL.

14 PAL: Program-aided Language Models

Điểm đặc biệt của PAL là việc giải quyết vấn đề được chuyển giao cho trình thông dịch, như Python, giúp tận dụng sức mạnh tính toán và độ chính xác cao của máy tính. Được đề xuất trong bài báo "PAL: Program-aided Language Models" Gao et al., 2023, phương pháp này tập trung vào việc phân tách vấn đề thành các bước có thể thực thi, giải phóng LLM khỏi trách nhiệm giải quyết vấn đề mà thay vào đó, giao cho trình thông dịch. Qua việc áp dụng PAL trong 13 nhiệm vụ suy luận toán học, biểu tượng, và thuật toán từ các bài kiểm tra khó như BIG-Bench Hard và các tiêu chuẩn khác.



Hình 15: Sơ đồ PAL: Đối với một câu hỏi suy luận toán học, Chain-of-Thought tạo ra các bước suy luận trung gian dưới dạng văn bản tự do. Ngược lại, PAL tạo ra các bước trung gian và mã Python. Điều này chuyển vai trò thực thi các bước suy luận từ mô hình ngôn ngữ sang trình thông dịch Python. Câu trả lời cuối cùng được thu bằng cách chạy chuỗi suy luận đã tạo.

PAL là một phương pháp độc đáo và sáng tạo trong việc giải quyết các vấn đề ngôn ngữ tự nhiên bằng cách kết hợp mạch lạc giữa ngôn ngữ tự nhiên (Natural Language - NL) và ngôn ngữ lập trình (Programming Language - PL). Trong tác vụ này, một vấn đề được đặt ra trong ngôn ngữ tự nhiên được biến đổi thành một loạt các câu lệnh xen kẽ giữa NL và PL, tạo ra một dạng hợp nhất giữa hai loại ngôn ngữ này. Điều này cho phép mô hình không chỉ phân tích và hiểu vấn đề mà còn tự động tạo ra mã lập trình như một phần của quy trình giải quyết vấn đề, mà không cần cung cấp trực tiếp các câu trả lời cuối cùng trong prompt của mình.

Điểm đặc biệt của PAL là việc giải quyết vấn đề được chuyển giao cho một trình thông dịch, ví dụ như trình thông dịch Python, cho phép mô hình tập trung vào việc xây dựng và tối ưu hóa các

bước giải quyết vấn đề một cách logic và hiệu quả. Các bước trung gian và câu lệnh lập trình được mô hình tạo ra sau đó được thực thi bởi trình thông dịch để thu được kết quả cuối cùng. Điều này không chỉ tăng cường khả năng của mô hình trong việc xử lý các vấn đề phức tạp mà còn mở rộng khả năng tương tác của nó với các ngôn ngữ lập trình thực tế.

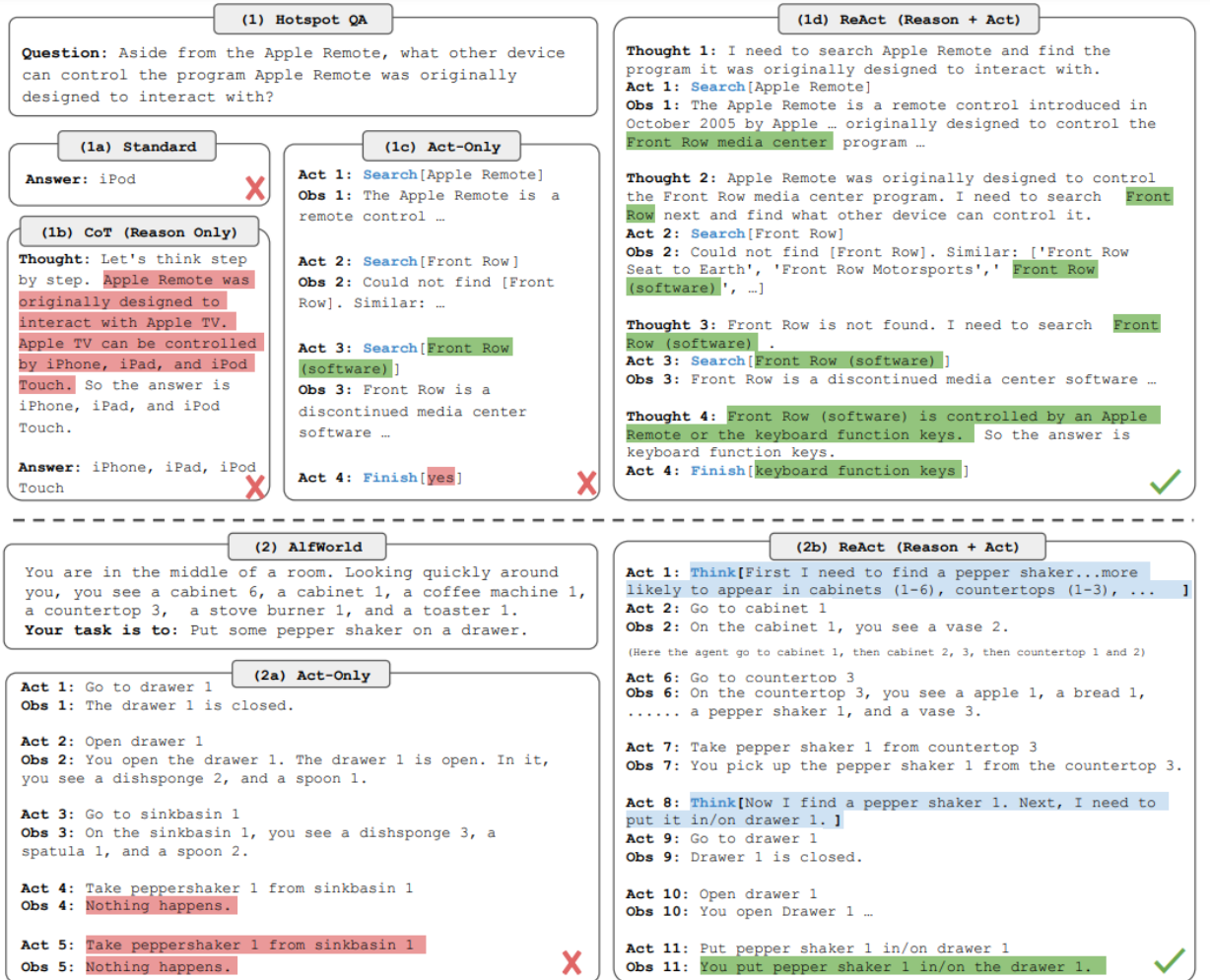
Trong quá trình thiết lập PAL, việc tận dụng các prompt có sẵn từ các nghiên cứu trước đây và bổ sung vào đó các ví dụ ngẫu nhiên để tạo nên một prompt đa dạng và phong phú, được tối ưu hóa bằng cách sử dụng các cấu trúc lập trình phổ biến như vòng lặp và dictionary. Quá trình này không chỉ giúp tạo ra các prompts chất lượng cao mà còn đảm bảo rằng các biến trong code được tạo ra có tên phản ánh chính xác vai trò và ý nghĩa của chúng trong vấn đề được giải quyết.

PAL mở ra một hướng đi mới trong việc sử dụng trí tuệ nhân tạo để giải quyết vấn đề, nơi sự kết hợp giữa suy luận ngôn ngữ tự nhiên và lập trình cung cấp một cách tiếp cận linh hoạt và mạnh mẽ trong việc xử lý các thách thức phức tạp, từ đó thúc đẩy sự phát triển của các hệ thống AI thông minh và có khả năng thích ứng cao.

15 ReAct Prompting

ReAct Prompting được giới thiệu trong paper "ReAct: Synergizing Reasoning and Acting in Language Models" Yao et al., 2023b, là một kỹ thuật để tạo ra cả dấu vết suy luận và hành động cụ thể cho nhiệm vụ một cách xen kẽ, cho phép sự kết hợp chặt chẽ hơn giữa hai phần: theo vết suy luận giúp mô hình khởi tạo, theo dõi và cập nhật kế hoạch hành động cũng như xử lý ngoại lệ, trong khi hành động cho phép nó tương tác với và thu thập thông tin bổ sung từ các nguồn bên ngoài như cơ sở dữ liệu tri thức hoặc môi trường. ReAct cho một tập hợp đa dạng các nhiệm vụ về ngôn ngữ và ra quyết định, đã đạt hiệu quả so với các tiêu chuẩn hàng đầu cũng như cải thiện tính dễ hiểu và đáng tin cậy đối với con người. Cụ thể, trên các nhiệm vụ trả lời câu hỏi (HotpotQA) và xác minh sự thật (Fever), ReAct giải quyết các vấn đề phổ biến về ảo giác và lan truyền lỗi trong CoT bằng cách tương tác với API Wikipedia đơn giản, và tạo ra quỹ đạo giải quyết nhiệm vụ giống như con người dễ hiểu hơn so với các tiêu chuẩn không có dấu vết suy luận.

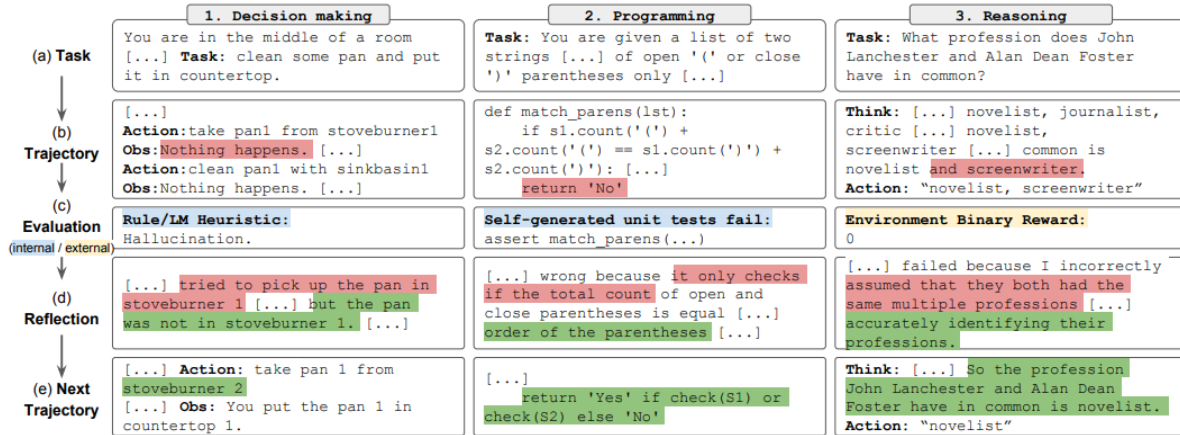
Thông qua việc sử dụng một số ít ví dụ điển hình trong prompt, với mỗi quỹ đạo bao gồm nhiều bước suy nghĩ-hành động-quan sát. ReAct khai thác sức mạnh của suy nghĩ tự do cho nhiều mục đích, từ phân rã câu hỏi, trích xuất thông tin từ Wikipedia, suy luận thông thường và toán học, đến hướng dẫn cải tiến tìm kiếm và tổng hợp câu trả lời cuối cùng. Đối với các tiêu chuẩn so sánh, loại bỏ từng phần trong quỹ đạo ReAct để xây dựng các lời nhắc cho nhiều cơ sở so sánh, từ CoT chỉ tập trung vào suy luận, đến hành động duy nhất mà không cần suy nghĩ, và kết hợp cả hai để tạo ra quyết định dựa trên cả kiến thức nội bộ và bên ngoài.



Hình 16: So sánh 4 phương pháp prompting, standard, Chain-of-thought (CoT, Reason Only), Act-only và ReAct (Reason+Act)

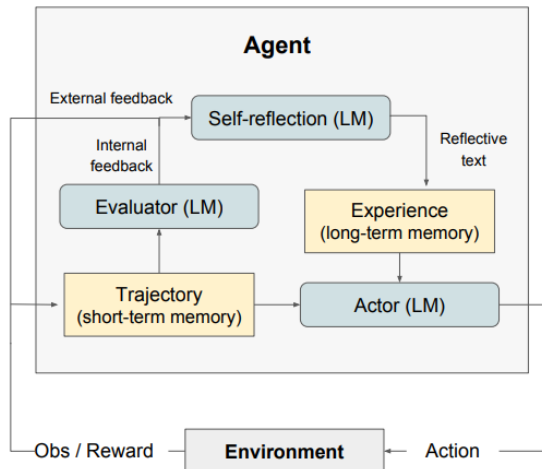
16 Reflexion

Reflexion được đề xuất trong bài báo "Reflexion: Language Agents with Verbal Reinforcement Learning" Shinn et al., 2023, đánh dấu bước tiến trong việc cải thiện khả năng của các language agents - những mô hình ngôn ngữ lớn được sử dụng ngày càng nhiều để tương tác với môi trường bên ngoài như trò chơi, trình biên dịch, API với vai trò là các đại diện có mục tiêu. Mặc dù việc áp dụng các phương pháp học tăng cường truyền thống đang đối mặt với thách thức lớn, do yêu cầu lượng dữ liệu huấn luyện rộng lớn và việc tinh chỉnh mô hình tốn kém, Reflexion mở ra một hướng đi mới. Thay vì cập nhật trọng số, Reflexion tăng cường khả năng của language agents thông qua phản hồi ngôn ngữ. Cụ thể, Reflexion agents phản ánh về tín hiệu phản hồi công việc thông qua lời nói, sau đó duy trì văn bản phản ánh của chính nó trong một bộ đệm theo dõi để thúc đẩy quyết định tốt hơn trong các lần thử tiếp theo. Reflexion đủ linh hoạt để kết hợp các loại (giá trị vô hướng hoặc ngôn ngữ tự do) và nguồn (bên ngoài hoặc mô phỏng nội bộ) tín hiệu phản hồi, và đạt được những cải thiện đáng kể so với một đại diện cơ sở trên nhiều nhiệm vụ đa dạng (quyết định tuần tự, lập trình, suy luận ngôn ngữ).



Hình 17: Quá trình hoạt động của Reflexion qua 3 tác vụ decision-making, programming và reasoning.

Reflexion là một khái niệm đột phá trong lĩnh vực trí tuệ nhân tạo, tạo ra một cách tiếp cận mới cho việc tăng cường khả năng tự học của các language agents thông qua ba thành phần chính: Actor, Evaluator và Self-Reflection. Actor sử dụng mô hình ngôn ngữ lớn để tạo ra văn bản và hành động dựa trên quan sát. Evaluator đánh giá chất lượng của các kết quả này và cung cấp điểm thưởng, trong khi Self-Reflection tạo ra phản hồi ngôn từ để hỗ trợ Actor phát triển qua từng lần thử. Phản hồi này sau đó được lưu trữ trong bộ nhớ, giúp Actor điều chỉnh hành động trong các lần thử tiếp theo. Quy trình này không chỉ tối ưu hóa việc học từ kinh nghiệm mà còn cho phép các đại diện nhanh chóng cải thiện quyết định của mình thông qua việc sử dụng phản hồi có giá trị, tạo ra một cơ chế học tập mạnh mẽ và linh hoạt.



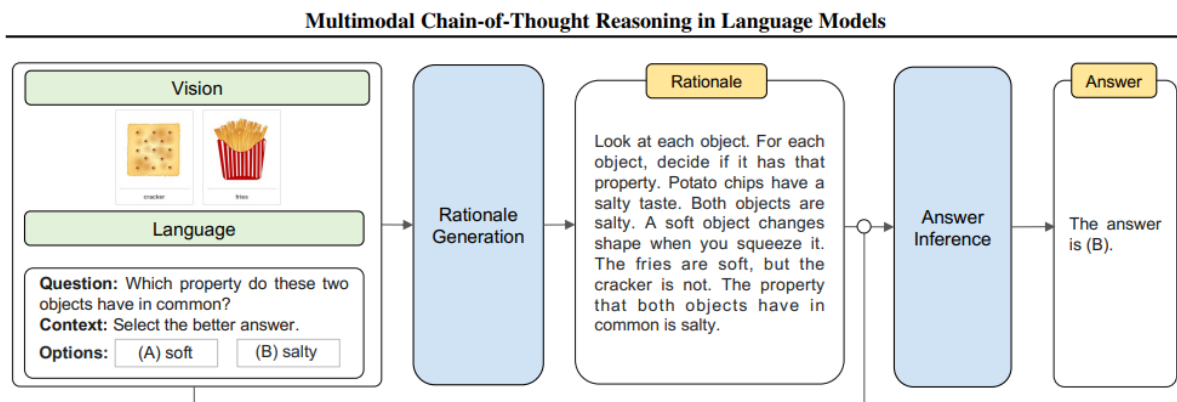
Algorithm 1 Reinforcement via self-reflection

Initialize Actor, Evaluator, Self-Reflection:
 M_a, M_e, M_{sr}
 Initialize policy $\pi_\theta(a_i|s_i), \theta = \{M_a, mem\}$
 Generate initial trajectory using π_θ
 Evaluate τ_0 using M_e
 Generate initial self-reflection sr_0 using M_{sr}
 Set $mem \leftarrow [sr_0]$
 Set $t = 0$
while M_e not pass or $t < \text{max trials}$ **do**
 Generate $\tau_t = [a_0, o_0, \dots, a_i, o_i]$ using π_θ
 Evaluate τ_t using M_e
 Generate self-reflection sr_t using M_{sr}
 Append sr_t to mem
 Increment t
end while
return

Hình 18: Diagram và Pseudo-Code của Reflexion

17 Multimodal CoT Prompting

Các mô hình ngôn ngữ lớn đã thể hiện hiệu suất ấn tượng trong việc giải quyết các bài toán suy luận phức tạp bằng cách sử dụng COT để tạo ra các chuỗi suy luận trung gian như rationale (lý do) để suy luận ra câu trả lời. Tuy nhiên, các nghiên cứu về COT trước đó đã tập trung vào mô hình ngôn ngữ. Multimodal-CoT được đề xuất trong paper "Multimodal Chain-of-Thought Reasoning in Language Models" Zhang et al., 2023b, là một phương pháp kết hợp cả hai mô hình ngôn ngữ và hình ảnh vào một two-stage framework tách biệt việc tạo lý do và suy luận câu trả lời. Nhờ đó, việc suy luận câu trả lời có thể tận dụng tốt hơn các lý do được tạo ra dựa trên thông tin đa phương tiện.



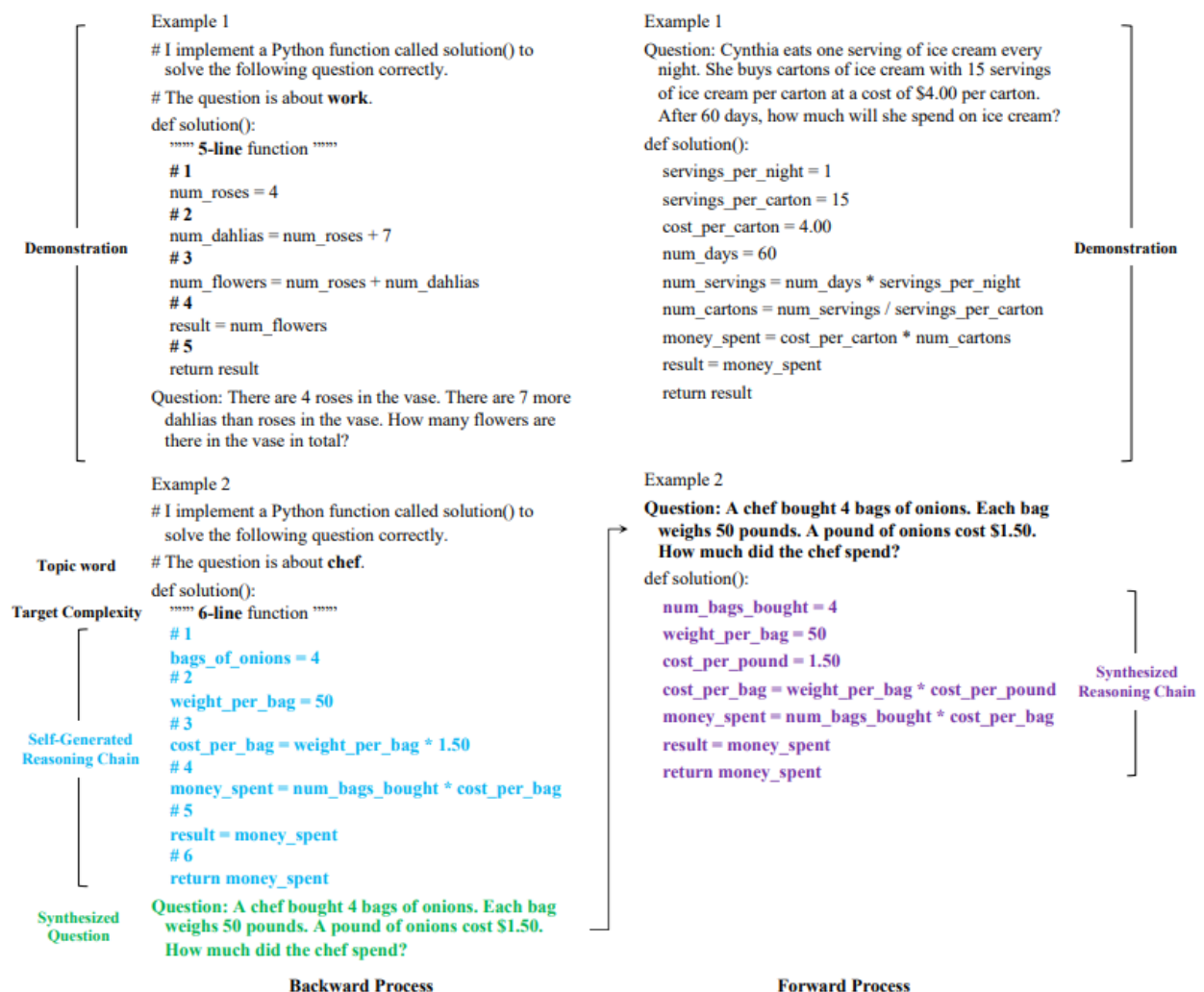
Hình 19: Multimodal-CoT bao gồm hai giai đoạn: (i) tạo lập lý do và (ii) suy luận câu trả lời. Cả hai giai đoạn đều sử dụng cùng một kiến trúc mô hình nhưng khác nhau về đầu vào và đầu ra. Trong giai đoạn đầu tiên, cung cấp cho mô hình đầu vào ngôn ngữ và hình ảnh để tạo ra lý do. Trong giai đoạn thứ hai, thêm đầu vào ngôn ngữ ban đầu với lý do được tạo ra từ giai đoạn đầu tiên. Sau đó, cung cấp đầu vào ngôn ngữ đã được cập nhật cùng với đầu vào hình ảnh ban đầu cho mô hình để suy luận ra câu trả lời.

Multimodal-CoT hoạt động với 2-stage kết hợp giữa ngôn ngữ và thị giác để cải thiện suy luận và đưa ra câu trả lời cho các mô hình ngôn ngữ lớn. Giai đoạn đầu tiên, tạo các lý do, nhận đầu vào là dữ liệu ngôn ngữ và hình ảnh để sinh ra chuỗi lý do. Giai đoạn thứ hai, suy luận câu trả lời, bắt đầu bằng việc thêm chuỗi lý do vào đầu vào ngôn ngữ gốc, và sau đó kết hợp với đầu vào thị giác ban đầu để mô hình có thể suy luận ra câu trả lời. Cả hai giai đoạn này sử dụng chung kiến trúc mô hình nhưng khác biệt về dữ liệu đầu vào và đầu ra.

Trong cả hai giai đoạn, mô hình được huấn luyện để xử lý thông tin đa phương tiện: đầu vào ngôn ngữ được mã hóa bởi một bộ mã hóa ngôn ngữ, trong khi đầu vào hình ảnh được xử lý bởi một bộ trích xuất đặc trưng hình ảnh và sau đó được kết hợp thông qua một single-head attention network để tạo đầu ra hợp nhất. Đầu ra này sau đó được đưa vào Transformer decoder để sinh ra dự đoán cuối cùng. Cách tiếp cận này không chỉ tận dụng được sức mạnh của cả văn bản và hình ảnh mà còn cho phép mô hình tạo ra các lý do suy luận phong phú và đa dạng, tăng cường khả năng đưa ra câu trả lời chính xác và có cơ sở.

18 Synthetic Prompting

Bài báo "Synthetic Prompting: Generating Chain-of-Thought Demonstrations for Large Language Models" Shao et al., 2023, khám phá ý tưởng sử dụng kỹ thuật chain-of-thought (CoT) để hướng dẫn các mô hình ngôn ngữ lớn (LLMs) giải quyết các nhiệm vụ suy luận khác nhau. Ý tưởng chính là LLMs có thể tìm ra câu trả lời tốt hơn bằng cách theo dõi một loạt các bước giải thích từng bước. Tuy nhiên, việc tạo ra những bước giải thích này một cách thủ công là tốn thời gian và đắt đỏ. Bài báo giới thiệu một phương pháp gọi là Synthetic Prompting, sử dụng một vài ví dụ thủ công để gợi ý cho mô hình tự tạo ra thêm ví dụ. Phương pháp này bao gồm một quá trình ngược để tạo ra các câu hỏi phù hợp với chuỗi suy nghĩ được lấy mẫu, đảm bảo các câu hỏi rõ ràng và có thể giải quyết, và một quá trình để tạo ra chuỗi suy nghĩ chi tiết hơn, từ đó cải thiện chất lượng của các ví dụ. Việc đánh giá phương pháp này trên các nhiệm vụ suy luận toán học, biểu tượng và thuật toán cho thấy nó vượt trội hơn các kỹ thuật prompting hiện có.



Với phương pháp này, thay vì đưa trực tiếp các ví dụ về chuỗi suy luận cho từng câu hỏi như chain-of-thought, phương pháp này lại sử dụng những ví dụ này như những ví dụ mẫu để mô hình LLM tự động sinh ra các ví dụ khác qua 2 quá trình backward-forward.

- Ở quá trình backward, câu hỏi sẽ được LLM tạo ra dựa trên những câu hỏi mới, cộng với một topic cho trước, một độ phức tạp dự kiến và một chuỗi suy luận được tạo tự động. Có một điểm cần lưu ý là chuỗi suy luận sẽ được tạo ra trước dựa trên từ chủ đề và độ phức tạp dự kiến, sau đó mô hình LLM sẽ tạo ra câu hỏi dựa trên chuỗi suy luận đó.
- Quá trình forward có mục đích là tạo ra chuỗi suy luận dựa vào câu hỏi vừa được tạo ra. Các tác giả nhận ra rằng chuỗi suy luận được tạo ra trong quá trình forward sẽ phù hợp và chính xác hơn cái được tạo ra trong quá trình backward, vì nó trực tiếp phụ thuộc vào câu hỏi. Cuối cùng một ví dụ mới được hình thành bằng cách sử dụng chuỗi suy luận từ quá trình forward và câu hỏi từ quá trình backward.

Cuối cùng, trong quá trình inference, một tập nhỏ các câu hỏi được tạo sinh ra sẽ được sử dụng để làm ví dụ mẫu cho mô hình. Phương pháp này nhìn chung cho thấy sự vượt trội trong các task liên quan đến lập trình, toán học, và suy luận dựa trên biểu tượng.

Phần III: Tổng kết

Prompt engineering là yếu tố then chốt để khai thác hết khả năng của các mô hình ngôn ngữ tiên tiến. Sự hiệu quả và độ chính xác của các phản hồi từ mô hình ngôn ngữ lớn phụ thuộc rất nhiều vào cách thức ta thiết kế và cung cấp các prompt. Việc tinh chỉnh và tạo ra những prompts thông minh và sáng tạo không chỉ giúp các mô hình ngôn ngữ lớn hiểu rõ hơn yêu cầu của người dùng mà còn tối ưu hóa khả năng phản hồi của nó. Bài viết này mang lại cái nhìn tổng quan về những kỹ thuật prompt engineering đang được áp dụng hiện nay, mở ra những khả năng mới trong việc tương tác với các hệ thống trí tuệ nhân tạo.

References

- Bsharat, S. M., Myrzakhan, A., & Shen, Z. (2024). Principled instructions are all you need for questioning llama-1/2, gpt-3.5/4.
- Wei, J., Bosma, M., Zhao, V., Guu, K., Yu, A. W., Lester, B., Du, N., Dai, A. M., & Le, Q. V. Fine-tuned language models are zero-shot learners. In: In *International conference on learning representations*.
- Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., ... Amodei, D. Language models are few-shot learners (H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, & H. Lin, Eds.). In: *Advances in neural information processing systems* (H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, & H. Lin, Eds.). Ed. by Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., & Lin, H. 33. Curran Associates, Inc., 2020, 1877–1901. https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfc4967418bfb8ac142f64a-Paper.pdf
- Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q., & Zhou, D. (2022). Chain-of-thought prompting elicits reasoning in large language models [cite arxiv:2201.11903]. <http://arxiv.org/abs/2201.11903>
- Kojima, T., Gu, S. S., Reid, M., Matsuo, Y., & Iwasawa, Y. Large language models are zero-shot reasoners. In: In *Advances in neural information processing systems*. 35. 2022, 22199–22213.
- Zhang, Z., Zhang, A., Li, M., & Smola, A. Automatic chain of thought prompting in large language models. In: In *The eleventh international conference on learning representations (iclr 2023)*. 2023.

- Wang, X., Wei, J., Schuurmans, D., Le, Q., hsin Chi, E. H., & Zhou, D. (2022). Self-consistency improves chain of thought reasoning in language models. *ArXiv, abs/2203.11171*. <https://api.semanticscholar.org/CorpusID:247595263>
- Liu, J., Liu, A., Lu, X., Welleck, S., West, P., Bras, R. L., Choi, Y., & Hajishirzi, H. (2021). Generated knowledge prompting for commonsense reasoning. *arXiv preprint arXiv:2110.08387*.
- Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T. L., Cao, Y., & Narasimhan, K. (2023a). Tree of thoughts: Deliberate problem solving with large language models.
- Zhou, Y., Muresanu, A. I., Han, Z., Paster, K., Pitis, S., Chan, H., & Ba, J. (2022). Large language models are human-level prompt engineers.
- Diao, S., Wang, P., Lin, Y., & Zhang, T. (2023). Active prompting with chain-of-thought for large language models.
- Li, Z., Peng, B., He, P., Galley, M., Gao, J., & Yan, X. (2023). Guiding large language models via directional stimulus prompting. *arXiv preprint arXiv:2302.11520*.
- Gao, L., Madaan, A., Zhou, S., Alon, U., Liu, P., Yang, Y., Callan, J., & Neubig, G. (2023). Pal: Program-aided language models.
- Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., & Cao, Y. (2023b). React: Synergizing reasoning and acting in language models.
- Shinn, N., Cassano, F., Berman, E., Gopinath, A., Narasimhan, K., & Yao, S. (2023). Reflexion: Language agents with verbal reinforcement learning.
- Zhang, Z., Zhang, A., Li, M., Zhao, H., Karypis, G., & Smola, A. (2023b). Multimodal chain-of-thought reasoning in language models.
- Shao, Z., Gong, Y., Shen, Y., Huang, M., Duan, N., & Chen, W. (2023). Synthetic prompting: Generating chain-of-thought demonstrations for large language models.

- Hết -