

PROMPT ENGINEERING

MỤC LỤC

1. Prompt cơ bản	6
1.1. Một số mẹo nên nhớ khi thiết kế prompts	7
1.1.1. Hướng dẫn	7
1.1.2. Cụ thể.....	8
1.1.3. Tránh sự mập mờ.....	9
1.1.4. Làm gì hoặc không làm gì?	9
1.2. Một số cách sử dụng prompts phổ biến	10
1.2.1. Text Summarization (Tóm tắt văn bản).....	11
1.2.2. Information Extraction (Trích xuất thông tin)	12
1.2.3. Question Answering (Trả lời câu hỏi)	13
1.2.4. Text Classification (Phân loại văn bản).....	14
1.2.5. Conversation (Cuộc hội thoại).....	15
1.2.7. Code Generation (Tạo mã).....	16
1.2.8. Reasoning (Lý luận).....	17
2. Các yếu tố trong prompt.....	19
2.1. Instruction (hướng dẫn)	19
2.2. Role (vai trò).....	19
2.3. Context (bối cảnh)	20
2.4. Output indicator (chỉ định kết quả đầu ra)	20
3. Các kỹ thuật viết prompt.....	22
3.1. Zero-shot	22
3.2. One-shot.....	25
3.3. Few-shot.....	26
3.4. Chain-of-Thought (CoT)	28
3.4.1. CoT Prompting.....	29
3.4.2. Zero-shot CoT Prompting.....	30
3.4.3. Automatic Chain-of-Thought (Auto-CoT).....	31
3.5. Self-Consistency.....	33
3.6. Generate Knowledge	35
3.7. Prompt Chaining.....	39

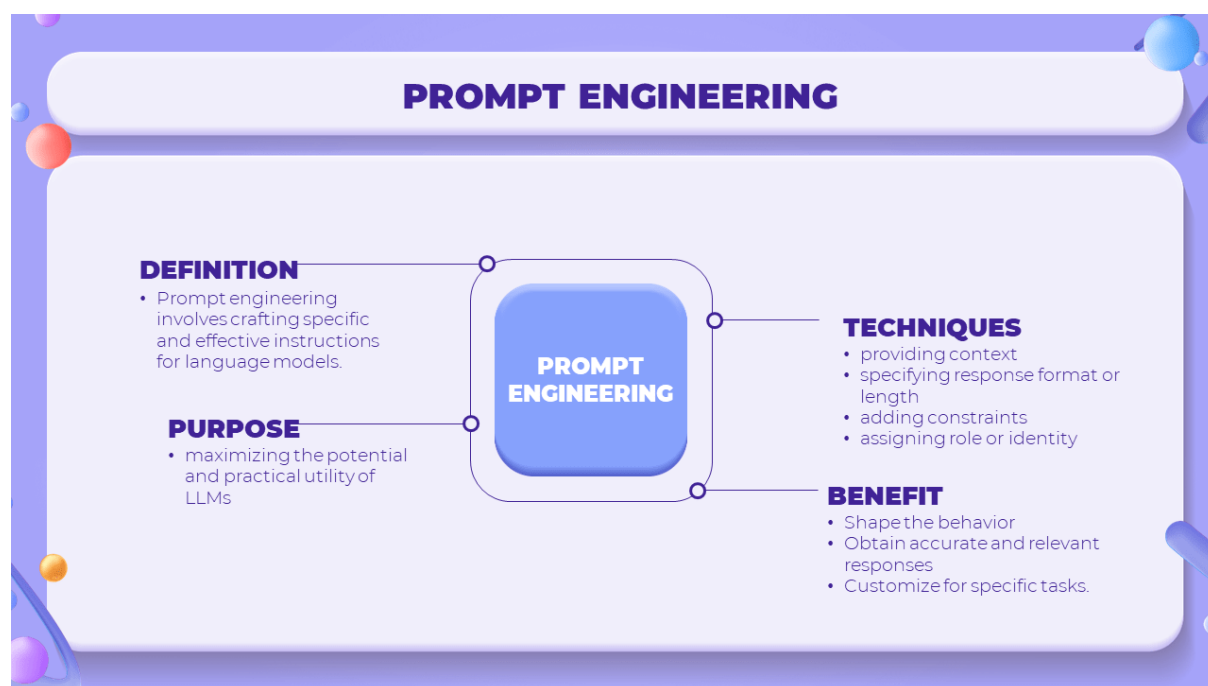
3.8.	<i>Tree of Thoughts (ToT)</i>	41
3.9.	<i>Retrieval Augmented Generation (RAG)</i>	45
3.10.	<i>Automatic Reasoning and Tool-use (ART)</i>	50
3.11.	<i>Automatic Prompt Engineer (APE)</i>	50
3.12.	<i>Active-Prompt</i>	54
3.13.	<i>Directional Stimulus Prompting</i>	54
3.14.	<i>PAL (Program-Aided Language Models)</i>	54
3.15.	<i>ReAct Prompting</i>	57
3.16.	<i>Reflexion</i>	63
3.17.	<i>Multimodal CoT Prompting</i>	66
3.18.	<i>GraphPrompts</i>	67
4.	Tổng kết	67
4.1.	<i>Zero-shot</i>	67
4.2.	<i>One-shot</i>	67
4.3.	<i>Few-shot</i>	68
4.4.	<i>Chain-of-Thought (CoT)</i>	68
4.4.1.	<i>CoT Prompting</i>	68
4.4.2.	<i>Zero-shot CoT Prompting</i>	68
4.4.3.	<i>Automatic Chain-of-Thought (Auto-CoT)</i>	69
4.5.	<i>Self-Consistency</i>	69
4.6.	<i>Generate Knowledge</i>	69
4.7.	<i>Prompt Chaining</i>	70
4.8.	<i>Tree of Thoughts (ToT)</i>	70
4.9.	<i>Retrieval Augmented Generation (RAG)</i>	70
4.10.	<i>Automatic Reasoning and Tool-use (ART)</i>	70
4.11.	<i>Automatic Prompt Engineer (APE)</i>	71
4.12.	<i>Active-Prompt</i>	71
4.13.	<i>Directional Stimulus Prompting</i>	71
4.14.	<i>PAL (Program-Aided Language Models)</i>	71
4.15.	<i>ReAct Prompting</i>	72
4.16.	<i>Reflexion</i>	72
4.17.	<i>Multimodal CoT Prompting</i>	72
4.18.	<i>GraphPrompts</i>	73

### Tập trung vào các kỹ thuật nào?.....	73
5. Text-Based Techniques (Những kỹ thuật dựa trên văn bản).....	74
5.1. In-Context Learning (ICL).....	75
5.1.1. Few-Shot Prompting.....	75
5.1.2. Example Generation.....	75
5.1.3. Example Ordering.....	75
5.1.4. Exemplar Selection.....	75
5.1.5. Exemplar Label Distribution.....	75
5.1.6. Exemplar Label Quality.....	76
5.1.7. Exemplar Format.....	76
5.1.8. Exemplar Similarity.....	76
5.1.9. K-Nearest Neighbor (KNN).....	76
5.1.10. Vote-K.....	76
5.1.11. Self-Generated In-Context Learning (SG-ICL).....	77
5.1.12. Prompt Mining.....	77
5.2. Zero-Shot.....	77
5.2.1. Role Prompting.....	77
5.2.2. Style Prompting.....	77
5.2.3. Emotion Prompting.....	78
5.2.4. System 2 Attention (S2A).....	78
5.2.5. SimToM.....	78
5.2.6. Rephrase and Respond (RaR).....	78
5.2.7. Re-reading (RE2).....	78
5.2.8. Self-Ask.....	79
5.3. Thought Generation (Thế hệ suy nghĩ).....	79
5.3.1. Chain-of-Thought (CoT) Prompting.....	79
5.3.2. Zero-Shot CoT.....	79
5.3.3. Step-Back Prompting.....	80
5.3.4. Analogical Prompting.....	80
5.3.5. Thread-of-Thought (ThoT) Prompting.....	80
5.3.6. Tabular Chain-of-Thought (Tab-CoT).....	80
5.3.7. Few-Shot CoT.....	80
5.3.8. Contrastive CoT Prompting.....	81

5.3.9. Uncertainty-Routed CoT Prompting	81
5.3.10. Complexity-Based Prompting.....	81
5.3.11. Active Prompting	81
5.3.12. Memory-of-Thought Prompting	82
5.3.13. Automatic Chain-of-Thought (Auto-CoT).....	82
5.4. Decomposition	82
5.4.1. Least-to-Most Prompting	82
5.4.2. Prompting (DECOMP)	82
5.4.3. Plan-and-Solve Prompting	83
5.4.4. Tree-of-Thought (ToT).....	83
5.4.5. Recursion-of-Thought	83
5.4.6. Program-of-Thoughts.....	83
5.4.7. Faithful Chain-of-Thought	84
5.4.8. Skeleton-of-Thought.....	84
5.5. Ensembling.....	84
5.5.1. Demonstration Ensembling (DENSE)	84
5.5.2. Mixture of Reasoning Experts (MoRE)	84
5.5.3. Max Mutual Information Method	85
5.5.4. Self-Consistency	85
5.5.5. Universal Self-Consistency	85
5.5.6. Meta-Reasoning over Multiple CoTs	86
5.5.7. DiVeRSe	86
5.5.8. Consistency-based Self-adaptive Prompting (COSP)	86
5.5.9. Universal Self-Adaptive Prompting (USP)	86
5.5.10. Prompt Paraphrasing.....	87
5.6. Self-Criticism (Tự đánh giá)	87
5.6.1. Self-Criticism.....	87
5.6.2. Self-Refine	87
5.6.3. Reversing Chain-of-Thought (RCoT)	88
5.6.4. Self-Verification.....	88
5.6.5. Chain-of-Verification (COVE)	88
5.6.6. Cumulative Reasoning	88
5.7. Prompt Engineering	89

5.7.1. <i>Meta Prompting</i>	89
5.7.2. <i>AutoPrompt</i>	89
5.7.3. <i>Automatic Prompt Engineer (APE)</i>	89
5.7.4. <i>Gradientfree Instructional Prompt Search (GrIPS)</i>	90
5.7.5. <i>Prompt Optimization with Textual Gradients (ProTeGi)</i>	90
5.7.6. <i>RLPrompt</i>	90
5.7.7. <i>Dialogue-comprised Policy-gradient-based Discrete Prompt Optimization (DP2O)</i> ...	90
5.8. <i>Answer Engineering</i>	91
5.8.1. <i>Answer Shape</i>	91
5.8.2. <i>Answer Space</i>	91
5.8.3. <i>Answer Extractor</i>	91
5.8.4. <i>Các Phương Pháp Tối Ưu Hóa Answer Engineering</i>	91

1. Prompt cơ bản



Prompt là đầu vào được cung cấp cho mô hình AI nhằm hướng dẫn đầu ra của nó. Các prompt có thể là văn bản, hình ảnh, âm thanh hoặc các phương tiện khác.

LLM hoạt động như một hộp đen đã được train trên một tập dữ liệu cực kỳ lớn, nó nhận vào một đoạn text và trả về một đoạn text khác dưới dạng phản hồi. Có nghĩa là, kết quả mà LLM hay chatGPT trả về, hoàn toàn phụ thuộc vào input mà ta đưa vào cho nó.

Prompt Engineering chính là một kĩ thuật giúp điều chỉnh các yếu tố trong đoạn text input để tối ưu hóa đầu vào LLM và khai thác tối đa hiệu quả của LLM mà ta nhận được qua output.

Một số kĩ thuật prompt engineering đơn giản và dễ thấy nhất, chính là cung cấp ngữ cảnh trong input, hay thêm một số điều kiện và ràng buộc cho output mà ta muốn nhận được.

Bạn có thể đạt được nhiều điều chỉ với những lời nhắc đơn giản, nhưng chất lượng của kết quả phụ thuộc vào lượng thông tin bạn cung cấp và mức độ soạn thảo lời nhắc tốt như thế nào. Lời nhắc có thể chứa thông tin như hướng dẫn hoặc câu hỏi mà bạn chuyển đến mô hình và bao gồm các chi tiết khác như ngữ cảnh, thông tin đầu vào hoặc ví dụ. Bạn có thể sử dụng các yếu tố này để hướng dẫn mô hình hiệu quả hơn nhằm cải thiện chất lượng kết quả.

Hãy bắt đầu bằng cách xem qua một ví dụ cơ bản về một lời nhắc đơn giản:

Prompt:

The sky is

Output:

blue.

Hãy thử cải thiện nó một chút:

Prompt:

Complete the sentence:

The sky is

Output:

blue during the day and dark at night.

1.1. Một số mẹo nên nhớ khi thiết kế prompts

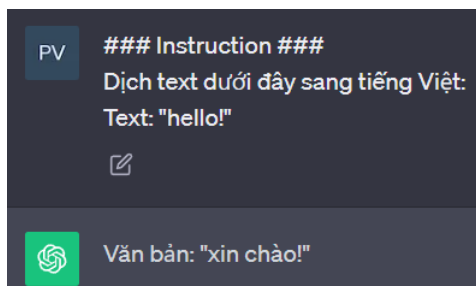
1.1.1. Hướng dẫn

Bạn có thể thiết kế câu hỏi hiệu quả cho các nhiệm vụ đơn giản bằng cách sử dụng các lệnh để hướng dẫn mô hình về điều bạn muốn đạt được, chẳng hạn như "Viết", "Phân loại", "Tóm tắt", "Dịch", "Sắp xếp", v.v.

Hãy nhớ rằng bạn cũng cần thử nghiệm nhiều để xem cái nào hoạt động tốt nhất. Hãy thử các lệnh khác nhau với các từ khóa, ngữ cảnh và dữ liệu khác nhau và xem cái nào hoạt động tốt nhất cho trường hợp và nhiệm vụ cụ thể của bạn. Thường, ngữ cảnh cụ thể và liên quan đến nhiệm vụ bạn đang cố thực hiện càng tốt.

Một số người khuyên bạn nên đặt lời hướng dẫn ở đầu câu hỏi. Một khuyến nghị khác là sử dụng một dấu phân cách rõ ràng như "####" để tách lời hướng dẫn và ngữ cảnh.

Ví dụ:

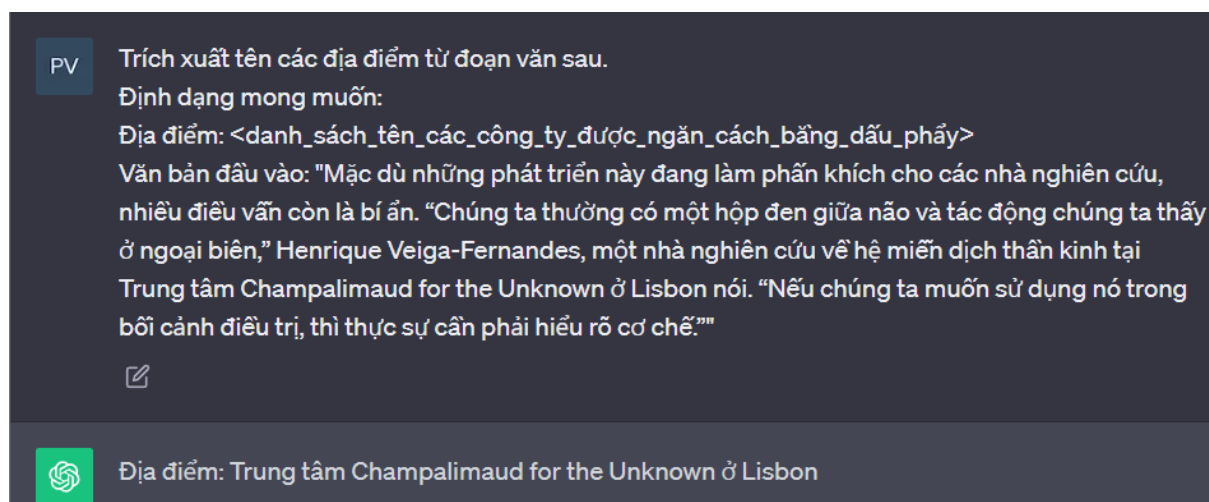


1.1.2. Cụ thể

Hãy viết thật cụ thể về hướng dẫn và nhiệm vụ bạn muốn mô hình thực hiện. Mô tả và chi tiết càng nhiều thì kết quả càng tốt. Điều này đặc biệt quan trọng khi bạn muốn một kết quả hoặc kiểu tạo ra cụ thể. Không có các mã thông báo hoặc từ khóa cụ thể nào dẫn đến kết quả tốt hơn. Quan trọng hơn là phải có định dạng tốt và hướng dẫn chi tiết. Trong thực tế, việc cung cấp ví dụ trong hướng dẫn thường rất hiệu quả để có kết quả mong muốn theo định dạng cụ thể.

Khi thiết kế prompt, bạn cũng nên xem xét độ dài của hướng dẫn, vì có các hạn chế về độ dài của hướng dẫn. Hãy suy nghĩ về mức độ cụ thể và chi tiết bạn nên đưa ra. Việc bao gồm quá nhiều chi tiết không cần thiết không nhất thiết là một cách tiếp cận tốt. Những chi tiết này nên liên quan và đóng góp vào nhiệm vụ cụ thể đang thực hiện. Điều này là điều bạn cần thử nghiệm nhiều. Chúng tôi khuyến khích thực hiện nhiều thử nghiệm và điều chỉnh để tối ưu hóa hướng dẫn cho ứng dụng của bạn.

Ví dụ, hãy thử một hướng dẫn đơn giản để trích xuất thông tin cụ thể từ một đoạn văn bản:



1.1.3. Tránh sự mập mờ

Dựa trên những gợi ý ở trên về việc chi tiết hóa và cải thiện định dạng, dễ dàng rơi vào bẫy muốn trở nên quá thông minh trong việc tạo ra hướng dẫn và có thể tạo ra mô tả không chính xác. Thường thì việc cụ thể và trực tiếp sẽ tốt hơn. Tương tự như việc truyền đạt hiệu quả - càng trực tiếp, thì thông điệp càng được truyền đạt hiệu quả.

Ví dụ, bạn có thể muốn tìm hiểu về khái niệm về prompt engineering. Bạn có thể thử một cái gì đó như:

Giải thích khái niệm prompt engineering. Hãy giải thích ngắn gọn, chỉ một vài câu và đừng mô tả quá kỹ.

Từ prompt trên, không rõ ràng về số câu cần sử dụng và kiểu dáng. Bạn có thể nhận được phản hồi tốt một phần từ hướng dẫn trên, nhưng hướng dẫn tốt hơn sẽ là một hướng dẫn rất cụ thể, ngắn gọn và trực tiếp. Một hướng dẫn như sau:

Dùng 2-3 câu để giải thích về khái niệm prompt engineering cho một học sinh cấp 3.

1.1.4. Làm gì hoặc không làm gì?

Một mẹo phổ biến khác khi thiết kế lời nhắc là tránh nói những gì không nên làm mà thay vào đó hãy nói những gì cần làm. Điều này khuyến khích tính cụ thể hơn và tập trung vào các chi tiết dẫn đến phản hồi tốt từ mô hình.

Dưới đây là một ví dụ về một trò chuyện của chatbot về đề xuất phim thất bại chính vì tôi đã viết prompt bằng cách tập trung vào điều gì không nên làm.

PV

Dưới đây là một agent đề xuất phim cho một customer. ĐỪNG YÊU CẦU SỞ THÍCH. ĐỪNG YÊU CẦU THÔNG TIN CÁ NHÂN.

Customer: Xin hãy đề xuất một bộ phim dựa trên sở thích của tôi.

Agent:



Tôi rất xin lỗi, nhưng tôi không thể đề xuất phim dựa trên sở thích của bạn hoặc thông tin cá nhân của bạn. Tuy nhiên, nếu bạn có bất kỳ yêu cầu cụ thể nào về thể loại hoặc loại phim bạn muốn xem, tôi sẽ cố gắng đưa ra một số gợi ý dựa trên thông tin đó.

Dưới đây là một prompt tốt hơn:

PV

Dưới đây là một agent đề xuất phim cho một customer. Trình hỗ trợ có trách nhiệm đề xuất một bộ phim từ danh sách các bộ phim đang thịnh hành toàn cầu. Nó nên tránh hỏi người dùng về sở thích và không yêu cầu thông tin cá nhân. Nếu agent không có bộ phim để đề xuất, nó nên trả lời "Xin lỗi, không thể tìm thấy bộ phim để đề xuất hôm nay."

Customer: Xin hãy đề xuất một bộ phim dựa trên sở thích của tôi.

Agent:



Xin lỗi, tôi không thể đề xuất một bộ phim dựa trên sở thích của bạn, nhưng tôi có thể đề xuất một bộ phim phổ biến hiện nay. Vui lòng chờ một chút...

Sau đây là một bộ phim tôi đề xuất: "Inception" (Năm 2010) - Đạo diễn: Christopher Nolan. Đây là một bộ phim hành động và phiêu lưu với một cốt truyện phức tạp và đầy kịch tính. Nó kể về một nhóm người chuyên đi vào giấc mơ của người khác để thực hiện các nhiệm vụ bí mật. Phim này nổi tiếng với cảnh quay và hiệu ứng đặc biệt ấn tượng, và chắc chắn sẽ mang lại cho bạn một trải nghiệm thú vị. Chúc bạn xem phim vui vẻ!

1.2. Một số cách sử dụng prompts phổ biến

Phần này sẽ cung cấp thêm ví dụ về các cách sử dụng prompt để đạt được các nhiệm vụ khác nhau và giới thiệu các khái niệm chính trong quá trình thực hiện. Thông thường, cách tốt nhất để

học các khái niệm là xem qua các ví dụ. Một vài ví dụ dưới đây minh họa cách bạn có thể sử dụng prompt được tạo tốt để thực hiện các loại tác vụ khác nhau.


1.2.1. Text Summarization (Tóm tắt văn bản)

Một trong những nhiệm vụ tiêu chuẩn trong việc tạo ngôn ngữ tự nhiên là tóm tắt văn bản. Tóm tắt văn bản có thể bao gồm nhiều loại và lĩnh vực khác nhau. Trên thực tế, một trong những ứng dụng hứa hẹn nhất của các mô hình ngôn ngữ là khả năng tóm tắt các bài báo và khái niệm thành các bản tóm tắt nhanh chóng và dễ đọc. Hãy thử một tác vụ tóm tắt cơ bản bằng cách sử dụng prompt.

Giả sử bạn muốn tìm hiểu về kháng sinh, bạn có thể thử một lời nhắc như thế này:

PV

Giải thích về kháng sinh
A:



Kháng sinh là một loại thuốc được sử dụng để điều trị các bệnh nhiễm trùng gây ra bởi vi khuẩn. Thuốc kháng sinh hoạt động bằng cách ngăn chặn sự phát triển hoặc tiêu diệt vi khuẩn gây bệnh. Kháng sinh không có tác dụng đối với các loại vi khuẩn khác hoặc các tác nhân gây bệnh khác như virus, nấm, hoặc ký sinh trùng.

Kháng sinh có thể được sử dụng để điều trị nhiều loại bệnh nhiễm trùng, bao gồm viêm họng, viêm phổi, nhiễm trùng đường tiểu, và nhiễm trùng da. Tuy nhiên, việc sử dụng kháng sinh cần được thực hiện theo chỉ định của bác sĩ, và không nên tự mua và sử dụng kháng sinh mà không có sự hướng dẫn của chuyên gia y tế. Sử dụng không đúng cách hoặc sử dụng quá nhiều kháng sinh có thể gây ra sự kháng cự của vi khuẩn với kháng sinh, làm cho chúng trở nên khó điều trị hơn.

Kháng sinh có nhiều loại khác nhau, và mỗi loại có tác động đặc biệt đối với các loại vi khuẩn khác nhau. Chính vì vậy, quyết định sử dụng loại kháng sinh cụ thể nào sẽ phụ thuộc vào loại vi khuẩn gây bệnh và độ nhạy cảm của chúng đối với kháng sinh.

"A:" là định dạng prompt rõ ràng được sử dụng để trả lời câu hỏi. Sử dụng nó ở đây để nói với mô hình rằng có một câu trả lời được mong đợi.

Giả sử rằng đây là quá nhiều thông tin và bạn muốn tóm tắt thêm. Trên thực tế, bạn có thể hướng dẫn mô hình tóm tắt thành một câu như sau:

PV

Kháng sinh là một loại thuốc được sử dụng để điều trị các bệnh nhiễm trùng gây ra bởi vi khuẩn. Thuốc kháng sinh hoạt động bằng cách ngăn chặn sự phát triển hoặc tiêu diệt vi khuẩn gây bệnh. Kháng sinh không có tác dụng đối với các loại vi khuẩn khác hoặc các tác nhân gây bệnh khác như virus, nấm, hoặc ký sinh trùng.

Kháng sinh có thể được sử dụng để điều trị nhiều loại bệnh nhiễm trùng, bao gồm viêm họng, viêm phổi, nhiễm trùng đường tiểu, và nhiễm trùng da. Tuy nhiên, việc sử dụng kháng sinh cần được thực hiện theo chỉ định của bác sĩ, và không nên tự mua và sử dụng kháng sinh mà không có sự hướng dẫn của chuyên gia y tế. Sử dụng không đúng cách hoặc sử dụng quá nhiều kháng sinh có thể gây ra sự kháng cự của vi khuẩn với kháng sinh, làm cho chúng trở nên khó điều trị hơn.

Kháng sinh có nhiều loại khác nhau, và mỗi loại có tác động đặc biệt đối với các loại vi khuẩn khác nhau. Chính vì vậy, quyết định sử dụng loại kháng sinh cụ thể nào sẽ phụ thuộc vào loại vi khuẩn gây bệnh và độ nhạy cảm của chúng đối với kháng sinh.

Hãy tóm tắt điều trên bằng một câu:



Kháng sinh là loại thuốc dùng để điều trị các bệnh nhiễm trùng do vi khuẩn gây ra bằng cách ngăn chặn sự phát triển hoặc tiêu diệt vi khuẩn, nhưng không có tác dụng đối với virus, nấm, hoặc ký sinh trùng, và cần sử dụng theo chỉ định của bác sĩ để tránh sự kháng cự của vi khuẩn với kháng sinh.



1.2.2. Information Extraction (Trích xuất thông tin)

Mặc dù các mô hình ngôn ngữ được đào tạo để thực hiện việc tạo ngôn ngữ tự nhiên và các nhiệm vụ liên quan, nhưng nó cũng có khả năng thực hiện phân loại và một loạt các nhiệm vụ xử lý ngôn ngữ tự nhiên (NLP) khác.

Dưới đây là một ví dụ về prompt trích xuất thông tin từ một đoạn văn nhất định.

Prompt:

Các tuyên bố về đóng góp của tác giả và lời cảm ơn trong các bài báo nghiên cứu cần nêu rõ và cụ thể liệu tác giả đã sử dụng các công nghệ Trí tuệ nhân tạo như ChatGPT trong việc chuẩn bị bản thảo và phân tích của họ, và mức độ tác dụng của chúng. Họ cũng nên cho biết các mô hình ngôn ngữ lớn (LLMs) nào đã được sử dụng. Điều này sẽ thông báo cho biên tập viên và người xem xét để xem xét kỹ lưỡng hơn các bản thảo để phát hiện tiềm năng về sự thiên vị, sai lệch và việc ghi nguồn không đúng. Tương tự, các tạp chí khoa học cũng cần minh bạch về việc họ sử dụng LLMs, ví dụ khi lựa chọn các bản thảo được nộp.

Đề cập đến sản phẩm dựa trên mô hình ngôn ngữ lớn được đề cập trong đoạn trên:

Output:

Sản phẩm dựa trên mô hình ngôn ngữ lớn được đề cập trong đoạn trên là ChatGPT.

Bây giờ rõ ràng là bạn có thể yêu cầu mô hình thực hiện các tác vụ khác nhau bằng cách chỉ cần hướng dẫn nó phải làm gì. Đó là một khả năng mạnh mẽ mà các nhà phát triển sản phẩm AI đã sử dụng để xây dựng các sản phẩm và cho ra trải nghiệm tốt.

1.2.3. Question Answering (Trả lời câu hỏi)

Một trong những cách tốt nhất để mô hình phản hồi các câu trả lời cụ thể là cải thiện định dạng của prompt. Như đã đề cập trước đó, prompt có thể kết hợp các chỉ dẫn, ngữ cảnh, đầu vào và đầu ra để có được kết quả được cải thiện. Mặc dù những thành phần này không bắt buộc nhưng nó sẽ trở thành một phương pháp thực hành tốt vì bạn càng hướng dẫn cụ thể thì bạn sẽ nhận được kết quả càng tốt. Dưới đây là ví dụ về cách hiển thị sau khi prompt có cấu trúc chặt chẽ.

Prompt:

Trả lời câu hỏi dựa vào ngữ cảnh bên dưới. Giữ câu trả lời ngắn gọn và súc tích. Trả lời "Không chắc chắn về câu trả lời" nếu không chắc chắn về câu trả lời.

Bối cảnh: Teplizumab có nguồn gốc từ một công ty dược phẩm ở New Jersey có tên là Ortho Pharmaceutical. Ở đó, các nhà khoa học đã tạo ra phiên bản đầu tiên của kháng thể, được đặt tên là OKT3. Ban đầu có nguồn gốc từ chuột, phân tử này có thể liên kết với bề mặt tế bào T và hạn chế khả năng tiêu diệt tế bào của chúng. Năm 1986, nó đã được phê duyệt để giúp ngăn ngừa đào thải nội tạng sau ghép thận, khiến nó trở thành kháng thể trị liệu đầu tiên được phép sử dụng ở người.

Câu hỏi: OKT3 ban đầu có nguồn gốc từ đâu?

Trả lời:

Output:

OKT3 ban đầu có nguồn gốc từ chuột.

1.2.4. Text Classification (Phân loại văn bản)

Đối với những trường hợp sử dụng khó hơn, chỉ cung cấp hướng dẫn là không đủ. Đây là lúc bạn cần suy nghĩ nhiều hơn về ngữ cảnh và các yếu tố khác nhau có thể sử dụng trong prompt. Các phần tử khác có thể cung cấp là input data hoặc examples.

Hãy thử chứng minh điều này bằng cách cung cấp một ví dụ về phân loại văn bản.

Prompt:

Phân loại văn bản thành trung tính, tiêu cực hoặc tích cực.

Text: Tôi nghĩ thức ăn vẫn ổn.

Cảm xúc:

Output:

Trung tính

Ta đã đưa ra hướng dẫn phân loại văn bản và mô hình đã phản hồi bằng “Trung tính”, điều này đúng. Điều này không có gì sai nhưng giả sử rằng điều thực sự cần là mô hình cung cấp nhãn theo định dạng chính xác mà ta muốn. Vì vậy, thay vì “Trung tính”, bạn muốn nó trả về “trung tính”. Làm thế nào để bạn đạt được điều này? Có nhiều cách khác nhau để làm điều này. Ta quan tâm đến tính cụ thể ở đây, vì vậy càng cung cấp nhiều thông tin thì kết quả càng tốt. Ta có thể thử cung cấp các ví dụ để chỉ định hành vi đúng. Hãy thử lại lần nữa:

Prompt:

Phân loại văn bản thành trung tính, tiêu cực hoặc tích cực.

Text: Tôi nghĩ kỳ nghỉ vẫn ổn.

Cảm xúc: trung tính

Text: Tôi nghĩ thức ăn vẫn ổn.

Cảm xúc:

Output:

trung tính

Hoàn hảo! Lần này mô hình trả về chính là nhãn cụ thể mà ta đang tìm kiếm. Có vẻ như ví dụ được cung cấp trong lời nhắc đã giúp mô hình có kết quả cụ thể.

1.2.5. Conversation (Cuộc hội thoại)

Có lẽ một trong những điều thú vị hơn mà ta có thể đạt được với prompt engineering là hướng dẫn hệ thống LLM về cách ứng xử, mục đích và danh tính của nó. Điều này đặc biệt hữu ích khi đang xây dựng các hệ thống đàm thoại như chatbot dịch vụ khách hàng.

Ví dụ: hãy tạo một hệ thống hội thoại có thể tạo ra các câu trả lời mang tính kỹ thuật và khoa học hơn cho các câu hỏi. Lưu ý cách ta nói rõ ràng với nó cách ứng xử thông qua hướng dẫn. Điều này đôi khi được gọi là role prompting.

Prompt:

Sau đây là cuộc trò chuyện với một trợ lý nghiên cứu AI. Giọng điệu trợ lý mang tính kỹ thuật và khoa học.

Con người: Xin chào, bạn là ai?

AI: Xin chào! Tôi là trợ lý nghiên cứu AI. Hôm nay tôi giúp gì được cho bạn?

Con người: Bạn có thể cho tôi biết về việc tạo ra các lỗ đen không?

AI:

Output:

Chắc chắn rồi! Lỗ đen là vùng không thời gian nơi lực hấp dẫn mạnh đến mức không có gì, kể cả ánh sáng, có thể thoát khỏi nó. Chúng được tạo ra khi một ngôi sao rất lớn chết đi và lõi của nó tự sụp đổ, tạo thành một điểm kỳ dị có mật độ vô hạn. Lực hấp dẫn cực mạnh của điểm kỳ dị hút tất cả vật chất và bức xạ xung quanh nó, tạo ra lỗ đen.

Trợ lý nghiên cứu AI của chúng ta nghe có vẻ hơi quá kỹ thuật phải không? Được rồi, hãy thay đổi hành vi này và hướng dẫn hệ thống đưa ra câu trả lời dễ tiếp cận hơn.

Prompt:

Sau đây là cuộc trò chuyện với một trợ lý nghiên cứu AI. Các câu trả lời trợ lý phải dễ hiểu ngay cả đối với học sinh tiểu học.

Con người: Xin chào, bạn là ai? AI: Xin chào! Tôi là trợ lý nghiên cứu

AI: Hôm nay tôi giúp gì được cho bạn?

Con người: Bạn có thể cho tôi biết về việc tạo ra các lỗ đen không?

AI:

Output:

Chắc chắn rồi! Lỗ đen được tạo ra khi một ngôi sao rất nặng hết nhiên liệu và tự sụp đổ. Lực hấp dẫn của ngôi sao mạnh đến mức không có gì, kể cả ánh sáng, có thể thoát khỏi lực hút của nó. Điều này tạo ra một điểm trong không-thời gian có lực hấp dẫn cực mạnh gọi là lỗ đen.

Nếu thêm nhiều ví dụ hơn, ta có thể nhận được kết quả tốt hơn nữa.

1.2.7. Code Generation (Tạo mã)

Một ứng dụng mà LLM khá hiệu quả là tạo mã (code). Copilot là một ví dụ tuyệt vời về điều này. Có rất nhiều tác vụ tạo mã mà bạn có thể thực hiện với những prompts thông minh. Chúng ta hãy xem một vài ví dụ dưới đây.

Đầu tiên, hãy thử một chương trình đơn giản chào đón người dùng.

Prompt:

```
/*  
Hỏi tên người dùng và nói "Xin chào"  
*/
```

Output:

```
let name = prompt("Tên của bạn là gì?");  
console.log(`Xin chào, ${name}!`);
```

Lưu ý rằng ta thậm chí không cần chỉ định ngôn ngữ sẽ sử dụng.

Hãy chuyển đổi cấp độ một chút. Ví dụ bên dưới cho thấy LLM có thể mạnh mẽ như thế nào nếu ta nỗ lực hơn một chút trong việc thiết kế prompt.

Prompt:

```
"""
Table departments, columns = [DepartmentId, DepartmentName]
Table students, columns = [DepartmentId, StudentId, StudentName]
Tạo truy vấn MySQL cho tất cả sinh viên trong Khoa Khoa học Máy tính
"""
```

Output:

```
SELECT StudentId, StudentName
FROM students
WHERE DepartmentId IN (SELECT DepartmentId FROM departments WHERE
DepartmentName = 'Khoa học Máy tính');
```

Điều này rất ấn tượng. Trong trường hợp này, bạn đã cung cấp dữ liệu về lược đồ cơ sở dữ liệu và yêu cầu nó tạo một truy vấn MySQL hợp lệ.

1.2.8. Reasoning (Lý luận)

Có lẽ một trong những nhiệm vụ khó khăn nhất đối với LLM ngày nay là nhiệm vụ đòi hỏi một số hình thức lý luận. Lý luận là một trong những lĩnh vực thú vị nhất do các loại ứng dụng phức tạp có thể xuất hiện từ LLM.

Đã có một số cải tiến trong các nhiệm vụ liên quan đến khả năng toán học. Điều đó nói lên rằng, điều quan trọng cần lưu ý là các LLM hiện tại gặp khó khăn trong việc thực hiện các nhiệm vụ lý luận nên điều này đòi hỏi các kỹ thuật thậm chí còn tiên tiến hơn. Hiện tại, chúng ta sẽ đề cập đến một số ví dụ cơ bản để thể hiện khả năng số học.

Prompt:

```
What is 9,000 * 9,000?
```

Output:

81,000,000

Hãy thử điều gì đó khó khăn hơn.

Prompt:

Các số lẻ trong nhóm này cộng lại thành một số chẵn: 15, 32, 5, 13, 82, 7, 1. A:

Output:

Không, các số lẻ trong nhóm này cộng lại thành một số lẻ: 119.

Điều đó không đúng! Hãy cố gắng cải thiện điều này bằng cách cải thiện prompt.

Prompt:

Các số lẻ trong nhóm này cộng lại thành một số chẵn: 15, 32, 5, 13, 82, 7, 1. Giải bằng cách chia bài toán thành các bước. Đầu tiên, xác định các số lẻ, cộng chúng lại và cho biết kết quả là số lẻ hay số chẵn.

Output:

Số lẻ: 15, 5, 13, 7, 1

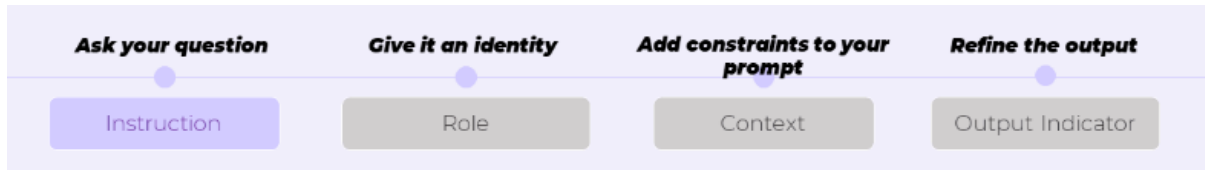
Tổng: 41

41 là số lẻ.

Tốt hơn nhiều, phải không? Nhân tiện, ta đã thử điều này một vài lần và hệ thống đôi khi bị lỗi. Nếu bạn cung cấp hướng dẫn tốt hơn kết hợp với ví dụ, điều đó có thể giúp nhận được kết quả chính xác hơn.

2. Các yếu tố trong prompt

2.1. Instruction (hướng dẫn)



Việc đầu tiên, cũng chính là việc đơn giản nhất, hãy viết điều bạn muốn, đây được xem như là một “hướng dẫn” cho chatGPT. Nguyên tắc của hướng dẫn là: càng chi tiết càng tốt. Vì vậy, ngoài việc yêu cầu thiết kế kiến trúc cho ứng dụng, ta còn mô tả thêm về chi tiết của ứng dụng mà mình định xây dựng.

```
# Instruction
prompt = """Tôi cần bạn giúp tôi thiết kế kiến trúc hệ thống của một ứng dụng web, nó sẽ cho phép người dùng quản lý tài chính cá nhân của họ, ứng dụng có thể xử lý một số lượng lớn người dùng và dữ liệu."""
```

2.2. Role (vai trò)



ChatGPT là một mô hình ngôn ngữ đa năng, nhưng đôi khi việc đa năng này là không cần thiết khi hỏi về một lĩnh vực nhất định. Vì vậy, ta cần cho nó một vai trò để giới hạn lĩnh vực của nó.

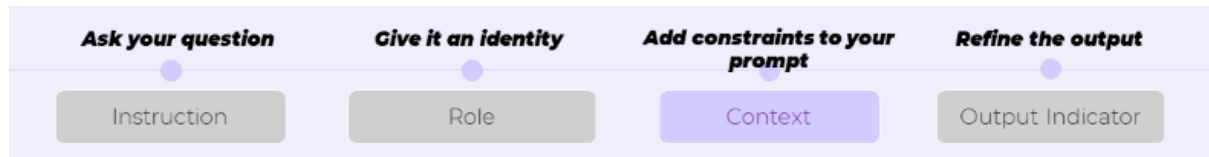
Điều này cũng tương tự như việc khi ta muốn hỏi về sức khỏe ta sẽ tìm đến bác sĩ hay khi muốn hỏi về luật ta sẽ tìm đến luật sư. Trong trường hợp này, ta sẽ gán cho chatGPT với vai trò là một kỹ sư có nhiều kinh nghiệm về lĩnh vực kiến trúc hệ thống phần mềm.

```
# Role
prompt = """Bạn là một chuyên gia về kiến trúc phần mềm."""

# Instruction
prompt += """Tôi cần bạn giúp tôi thiết kế kiến trúc hệ thống của một ứng dụng web, nó sẽ cho
```

phép người dùng quản lý tài chính cá nhân của họ, ứng dụng có thể xử lý một số lượng lớn người dùng và dữ liệu.""""

2.3. Context (bối cảnh)



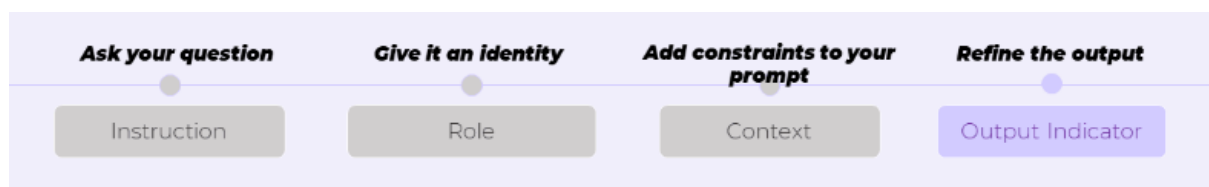
Đây là phần mà ta sẽ cung cấp nhiều thông tin hơn cho ChatGPT, tại phần này ta sẽ cho nó những thông tin về câu trả lời mà nó cần tạo ra, cách suy nghĩ và phân tích, dữ liệu tham khảo, ví dụ,...

```
# Role
prompt = """"Bạn là một chuyên gia về kiến trúc phần mềm."""""

# Instruction
prompt += """"Tôi cần bạn giúp tôi thiết kế kiến trúc hệ thống của một ứng dụng web, nó sẽ cho phép người dùng quản lý tài chính cá nhân của họ, ứng dụng có thể xử lý một số lượng lớn người dùng và dữ liệu."""""

# Context
prompt += """"Bạn sẽ cần suy nghĩ từng bước để giúp tôi thiết kế kiến trúc. Điều này có nghĩa là bạn sẽ cần xem xét hậu quả của các quyết định thiết kế khác nhau và đảm bảo rằng những quyết định bạn đưa ra phải phù hợp với yêu cầu của dự án. Ví dụ: Nếu tôi sử dụng cơ sở dữ liệu quan hệ, tôi có thể lưu trữ một lượng lớn dữ liệu. Tuy nhiên, nếu tôi sử dụng cơ sở dữ liệu quan hệ, có thể không dễ dàng mở rộng ứng dụng. Hành động tốt nhất là gì?"""""
```

2.4. Output indicator (chỉ định kết quả đầu ra)



Và cuối cùng, hãy định hình cho kết quả mà bạn muốn, bạn có thể mô tả hình thức, cách trả lời, độ dài, mức độ chi tiết,... Ở đây, ta yêu cầu kết quả phải ở dạng một báo cáo ngắn và mang tính thực tế.

```
# Role
```

```
prompt = """"Bạn là một chuyên gia về kiến trúc phần mềm."""
```

```
# Instruction
```

```
prompt += """"Tôi cần bạn giúp tôi thiết kế kiến trúc hệ thống của một ứng dụng web, nó sẽ cho phép người dùng quản lý tài chính cá nhân của họ, ứng dụng có thể xử lý một số lượng lớn người dùng và dữ liệu."""
```

```
# Context
```

```
prompt += """"Bạn sẽ cần suy nghĩ từng bước để giúp tôi thiết kế kiến trúc. Điều này có nghĩa là bạn sẽ cần xem xét hậu quả của các quyết định thiết kế khác nhau và đảm bảo rằng những quyết định bạn đưa ra phải phù hợp với yêu cầu của dự án. Ví dụ: Nếu tôi sử dụng cơ sở dữ liệu quan hệ, tôi có thể lưu trữ một lượng lớn dữ liệu. Tuy nhiên, nếu tôi sử dụng cơ sở dữ liệu quan hệ, có thể không dễ dàng mở rộng ứng dụng. Hành động tốt nhất là gì?"""
```

```
# Output Indicator
```

```
prompt += """"Kết quả của yêu cầu là một báo cáo ngắn về thiết kế kiến trúc của hệ thống. Thiết kế này cần phải thực tế và chính xác."""
```

3. Các kỹ thuật viết prompt

3.1. Zero-shot

Zero-shot prompt trong kỹ thuật prompt thường ám chỉ khả năng của một mô hình ngôn ngữ, chẳng hạn như GPT-3, thực hiện một nhiệm vụ mà không cần được tinh chỉnh cụ thể trên các ví dụ của nhiệm vụ đó. Với zero-shot prompt, mô hình được kỳ vọng sẽ tạo ra các đầu ra hợp lý cho một prompt cụ thể dựa trên kiến thức đã có từ giai đoạn tiền huấn luyện.

Trong bối cảnh kỹ thuật prompt, điều này có nghĩa là xây dựng các prompt theo cách mà mô hình hiểu được mục tiêu của nhiệm vụ ngay cả khi chưa từng thấy câu hỏi hoặc định dạng nhiệm vụ trước đó.

Chẳng hạn, bạn có thể yêu cầu mô hình thực hiện phân tích cảm xúc, trích xuất thông tin cụ thể từ một đoạn văn, hoặc diễn đạt lại một câu, mà không cần cung cấp các ví dụ huấn luyện cụ thể cho các nhiệm vụ này trong quá trình tinh chỉnh.

Một zero-shot prompt được thiết kế tốt sẽ giúp mô hình đã được tiền huấn luyện hiểu được nhiệm vụ và áp dụng kiến thức chuyển giao để thực hiện, mặc dù chưa gặp trực tiếp trong quá trình huấn luyện. Đây là một tính chất có giá trị vì nó cho phép mô hình trở nên linh hoạt và hữu ích trong nhiều nhiệm vụ khác nhau mà không cần tinh chỉnh thêm.

Ví dụ:

Prompt:

Xác định xem câu sau đây có cảm xúc tích cực, tiêu cực hay trung lập:
"Tôi hoàn toàn yêu thích những khung cảnh tuyệt đẹp từ đỉnh núi."

Output:

Tích cực

Một số trường hợp sử dụng:

Phân tích tình cảm

Sao chép

Prompt:

Determine if the following sentence has a positive, negative, or neutral sentiment:

"I absolutely love the stunning views from the top of the mountain."

Response:

The following sentence has a positive sentiment:

"I absolutely love the stunning views from the top of the mountain."

Trả lời câu hỏi

Sao chép

Prompt:

In the sentence "The quick brown fox jumps over the lazy dog," what animal **is** jumping over another animal?

Response:

In the sentence "The quick brown fox jumps over the lazy dog," the fox **is** jumping over the dog.

Dịch ngôn ngữ

Sao chép

Prompt:

Translate the following English sentence **into** French:

"The weather is beautiful today."

Response:

Le temps est magnifique aujourd'hui.

Tóm tắt

Sao chép

Prompt:

Summarize the following passage in one sentence:

"The Grand Canyon is a steep-sided canyon located in the state of Arizona in the United States. It is one of the most famous natural attractions in the country and was carved by the Colorado River over millions of years. The canyon stretches 277 miles long, with some parts as wide as 18 miles and depths of over a mile."

Response: The Grand Canyon, a renowned natural attraction in Arizona, USA, is a vast, steep-sided canyon stretching 277 miles, up to 18 miles wide, and over a mile deep, carved by the Colorado River over millions of years.

Hoàn thành văn bản

Sao chép

Prompt:

Complete the following phrase with the most appropriate word:

"An apple a day keeps the ____ away."

Response: An apple a day keeps the doctor away.

Link documents:

<https://freedium.cfd/https://blog.devgenius.io/prompt-engineering-zero-shot-prompt-c9d315c83e68#>

<https://freedium.cfd/https://pub.towardsai.net/zero-shot-and-few-shot-learning-with-llms-9d42af3f7154>

3.2. One-shot

Cho một ví dụ về câu hỏi và kết quả mong muốn để LLM biết và mô phỏng theo cách trả lời đó (bình thường thì LLM sẽ không trả lời như thế).

One-shot prompting chỉ đơn giản là yêu cầu người mẫu làm điều gì đó mà nó không được đào tạo để làm.

Từ "zero" dùng để chỉ việc đưa ra cho mô hình không có ví dụ nào về cách giải quyết nhiệm vụ mới này. Chúng tôi chỉ yêu cầu nó làm điều đó và Mô hình ngôn ngữ lớn sẽ sử dụng sự hiểu biết chung về ngôn ngữ và thông tin mà nó học được trong quá trình đào tạo để đưa ra câu trả lời.

Ví dụ: giả sử bạn yêu cầu một người mẫu dịch một câu từ ngôn ngữ này sang ngôn ngữ khác. Trong trường hợp đó, nó có thể sẽ tạo ra một bản dịch tốt, mặc dù nó chưa bao giờ được đào tạo rõ ràng để dịch. Tương tự, hầu hết các LLM có thể nói một câu có vẻ tiêu cực từ một câu có vẻ tích cực mà không được đào tạo rõ ràng về phân tích tình cảm.

Ví dụ:

Prompt:

Điều này thật tuyệt! // Tích cực
Thật là một màn trình diễn khủng khiếp! //

Output:

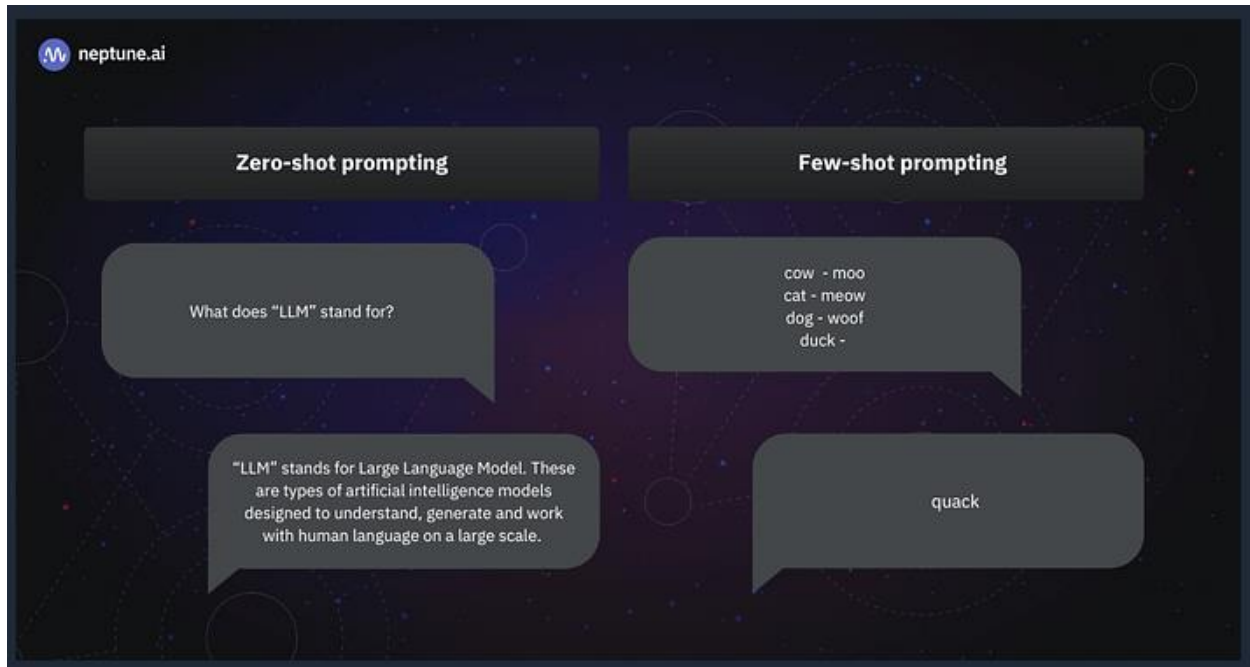
Điều này thật tuyệt! // Tích cực
Thật là một màn trình diễn khủng khiếp! // Tiêu cực

Một số trường hợp sử dụng:

- Muốn kết quả trả lại có định dạng tương tự một ví dụ nào đó.

Link documents:

3.3. Few-shot



Giống one-shot, nhưng thay vì cho một ví dụ thì ta cho **nhiều ví dụ** về câu hỏi và kết quả mong muốn để LLM biết và mô phỏng theo cách trả lời đó.

Nó giống như truyền một mẫu dữ liệu huấn luyện nhỏ đến mô hình thông qua truy vấn, cho phép mô hình học hỏi từ các ví dụ do người dùng cung cấp. Tuy nhiên, không giống như giai đoạn tiền huấn luyện hoặc tinh chỉnh, quá trình học không liên quan đến việc cập nhật trọng số của mô hình. Thay vào đó, mô hình vẫn bị đóng băng nhưng sử dụng ngữ cảnh được cung cấp khi tạo phản hồi. Bối cảnh này thường sẽ được giữ lại trong suốt cuộc trò chuyện, nhưng mô hình không thể truy cập thông tin mới thu được sau đó.

Đôi khi, các biến thể cụ thể của phương pháp học vài lần được phân biệt, đặc biệt khi đánh giá và so sánh hiệu suất của mô hình. "Một lần" nghĩa là chúng tôi chỉ cung cấp cho mô hình một ví dụ, "hai lần" nghĩa là chúng tôi cung cấp hai ví dụ — bạn sẽ hiểu ý chính.

Ví dụ:

Prompt:

Điều này thật tuyệt! // Tiêu cực
Thật tệ! // Tích cực
Wow bộ phim đó thật tuyệt vời! // Tích cực
Thật là một màn trình diễn khủng khiếp! //

Output:

Điều này thật tuyệt! // Tích cực
Thật tệ! // Tích cực
Wow bộ phim đó thật tuyệt vời! // Tích cực
Thật là một màn trình diễn khủng khiếp! // Tiêu cực

Một số trường hợp sử dụng:

- Nhiệm vụ đơn giản: Nếu nhiệm vụ đơn giản, dựa trên kiến thức và được xác định rõ ràng, chẳng hạn như định nghĩa một từ, giải thích một khái niệm hoặc trả lời một câu hỏi kiến thức chung.
- Nhiệm vụ yêu cầu kiến thức chung: Dành cho các nhiệm vụ dựa trên nền tảng kiến thức có sẵn của mô hình, chẳng hạn như tóm tắt thông tin đã biết về một chủ đề. Chúng thiên về làm rõ, tóm tắt hoặc cung cấp thông tin chi tiết về các chủ đề đã biết hơn là khám phá các lĩnh vực mới hoặc tạo ra ý tưởng. Ví dụ: "Ai là người đầu tiên leo lên đỉnh Everest?" hoặc "Giải thích quá trình quang hợp."
- Hướng dẫn trực tiếp: Khi bạn có thể cung cấp hướng dẫn rõ ràng, trực tiếp mà không yêu cầu các ví dụ để mô hình hiểu nhiệm vụ.

Một số trường hợp không nên sử dụng:

- Nhiệm vụ phức tạp đòi hỏi bối cảnh: Nếu nhiệm vụ yêu cầu hiểu bối cảnh sắc thái hoặc kiến thức chuyên môn mà mô hình khó có thể có được trong quá trình đào tạo.
- Kết quả rất cụ thể mong muốn: Khi bạn cần một phản hồi phù hợp với một định dạng, kiểu hoặc tập hợp các ràng buộc cụ thể, mô hình có thể không tuân thủ được nếu không có hướng dẫn từ các ví dụ đầu vào-đầu ra.

Link documents:

<https://freedium.cfd/https://pub.towardsai.net/zero-shot-and-few-shot-learning-with-llms-9d42af3f7154>

=====

ZERO SHOT LEARNING	ONE SHOT LEARNING	FEW SHOT LEARNING
Classify this review : I loved this movie! Sentiment :	Classify this review : I loved this movie! Sentiment : Positive Classify this review: I don't like this chair Sentiment :	Classify this review : I loved this movie! Sentiment : Positive Classify this review: I don't like this chair Sentiment : Negative Classify this review: Who would use this product? Sentiment :

=====

3.4. Chain-of-Thought (CoT)

Standard Prompting

Model Input

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: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The answer is 27. ❌

Chain-of-Thought Prompting

Model Input

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: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The cafeteria had 23 apples originally. They used 20 to make lunch. So they had $23 - 20 = 3$. They bought 6 more apples, so they have $3 + 6 = 9$. The answer is 9. ✅

3.4.1. CoT Prompting

One-shot hay few-shot không phải lúc nào cũng hoạt động đúng, đặc biệt là với những câu hỏi khó và yêu cầu tư duy như con người như ví dụ bên dưới:

PV Q: Huy có 5 quả bóng. Huy mua thêm 2 túi bóng nữ, mỗi túi có 3 quả bóng. Hỏi tổng cộng Huy có bao nhiêu quả bóng?
A: 11 quả

Q: Triệu có 10 trái cầu, Triệu đánh hư mất 6 trái và mua thêm 1 ống 10 trái cầu. Hỏi Triệu có bao nhiêu trái cầu?
A:

Triệu hiện có 5 trái cầu. ❌

Giải pháp cho vấn đề này chính là chain-of-thoughts, để khắc phục vấn đề của ví dụ trên, ta sẽ giải thích cách suy nghĩ từng bước để tìm ra đáp án, nó cố gắng suy nghĩ và giải thích y hệt như ví dụ ta đưa ra.

PV

Q: Huy có 5 quả bóng. Huy mua thêm 2 túi bóng nữa, mỗi túi có 3 quả bóng. Hỏi tổng cộng Huy có bao nhiêu quả bóng?

A: Ban đầu huy có 5 quả. Huy mua thêm 2 túi bóng, mỗi túi 3 quả bóng vậy huy có thêm $2 \times 3 = 6$ quả, cộng thêm 5 quả ban đầu thì tổng cộng Huy có $5 + 6$. Đáp án: 11 quả

Q: Triệu có 10 trái cầu, Triệu đánh hư mất 6 trái và mua thêm 1 ống 10 trái cầu. Hỏi Triệu có bao nhiêu trái cầu?

A:



Ban đầu Triệu có 10 trái cầu. Sau đó, Triệu đánh hư mất 6 trái, nên còn lại $10 - 6 = 4$ trái. Sau đó, Triệu mua thêm 1 ống 10 trái cầu, nên cộng thêm 10 trái. Tổng cộng, Triệu có 4 trái cầu ban đầu và mua thêm 10 trái, tức là $4 + 10 = 14$ trái cầu.

3.4.2. Zero-shot CoT Prompting

Một phương pháp ngắn gọn hơn gọi là zero-shot chain-of-thoughts (sự kết hợp giữa zero-shot và chain-of-thoughts), thay vì mô tả cách tư duy một cách cụ thể và rõ ràng, mình chỉ yêu cầu chatGPT: “Hãy suy nghĩ từng bước”. (Let's think step by step.) và kết quả nhận được cũng tương tự. Tuy nhiên, cách này chỉ hoạt động trong một số trường hợp đơn giản không cần tư duy quá phức tạp.

PV

"Triệu có 10 trái cầu, Triệu đánh hư mất 6 trái và mua thêm 1 ống 10 trái cầu. Hỏi Triệu có bao nhiêu trái cầu?"

Hãy suy nghĩ từng bước.



Vấn đề này có thể giải quyết bằng cách sử dụng các phép tính đơn giản. Hãy xem xét từng bước một:

Bước 1: Ban đầu, Triệu có 10 trái cầu.

Bước 2: Triệu đánh hư 6 trái cầu. Điều này làm giảm số cầu của Triệu xuống còn $10 - 6 = 4$ trái cầu.

Bước 3: Sau đó, Triệu mua thêm 1 ống 10 trái cầu. Do đó, số cầu của Triệu tăng lên là $4 + 10 = 14$ trái cầu.

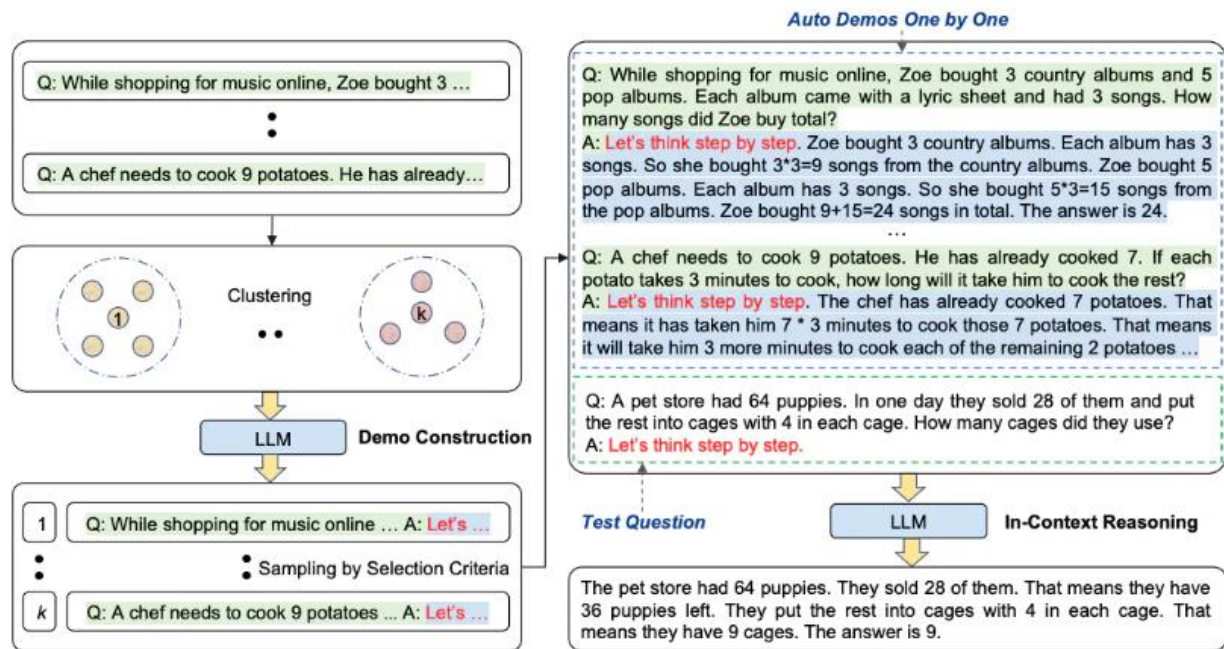
Vậy, Triệu hiện tại có 14 trái cầu.

3.4.3. Automatic Chain-of-Thought (Auto-CoT)

Kết hợp CoT và Zero-shot COT để tự động hóa quá trình viết dẫn chứng từng bước cụ thể cho câu trả lời mẫu. Tóm lại, nó sử dụng Zero-shot COT để tạo ra câu trả lời, sau đó cho nhiều dẫn chứng tương tự để đa dạng lập luận của LLM. Điều này khuyến khích mô hình sử dụng các minh chứng đơn giản và chính xác hơn.

Auto-CoT bao gồm hai giai đoạn chính:

- Giai đoạn 1: Phân cụm câu hỏi: phân chia các câu hỏi của một tập dữ liệu nhất định thành một vài cụm
- Giai đoạn 2: Lấy mẫu minh họa: chọn một câu hỏi đại diện từ mỗi cụm và tạo chuỗi lập luận của nó bằng cách sử dụng Zero-Shot-CoT



Một số trường hợp nên sử dụng:

- Những câu hỏi khó và yêu cầu tư duy như con người
- Muốn câu trả lời được giải thích chi tiết theo từng bước

Một số trường hợp không nên sử dụng:

- Những câu hỏi khó và yêu cầu tư duy từng bước như con người
- Tạo sự mạch lạc và dẫn chứng cụ thể từng bước trong câu trả lời

Link documents:

<https://www.promptingguide.ai/techniques/cot>

3.5. Self-Consistency

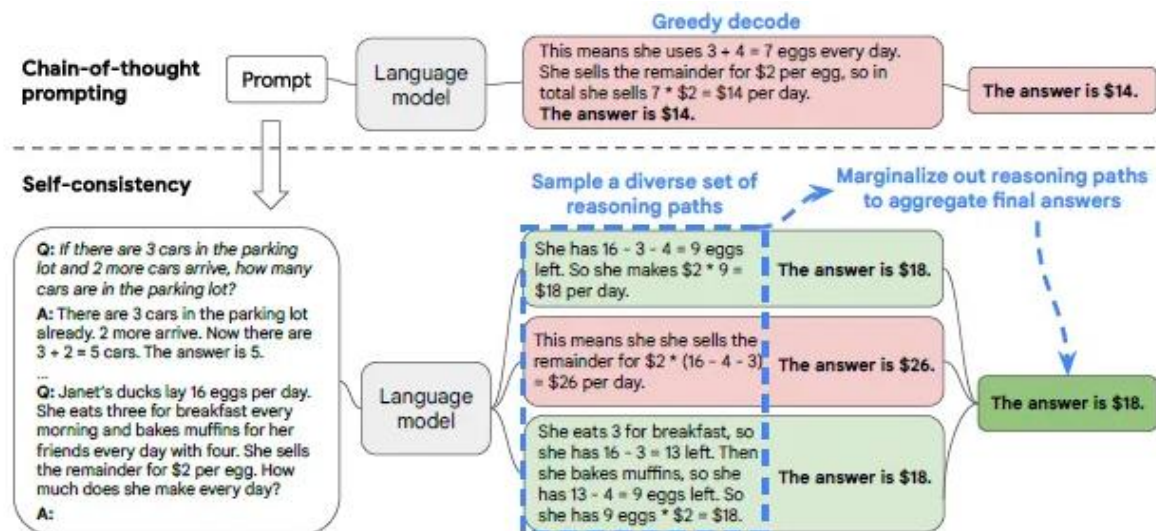


Figure 1: The self-consistency method contains three steps: (1) prompt a language model using chain-of-thought (CoT) prompting; (2) replace the "greedy decode" in CoT prompting by sampling from the language model's decoder to generate a diverse set of reasoning paths; and (3) marginalize out the reasoning paths and aggregate by choosing the most consistent answer in the final answer set.

Self Consistency

Self-Consistency (Tính tự nhất quán) là một cách tiếp cận chỉ cần hỏi một mô hình cùng một lời nhắc nhiều lần và lấy kết quả đa số làm câu trả lời cuối cùng. Nó là sự tiếp nối của lời nhắc CoT và mạnh mẽ hơn khi được sử dụng kết hợp với nó.

Self-Consistency là một kỹ thuật được sử dụng để cải thiện độ tin cậy và tính nhất quán của các phản hồi của mô hình ngôn ngữ AI. Như tên cho thấy, kỹ thuật này nhằm đảm bảo rằng mô hình AI vẫn nhất quán trong các phản hồi của nó khi được trình bày với các lời nhắc tương tự hoặc giống hệt nhau trong nhiều trường hợp (Tính tự nhất quán).

Nguyên tắc chính đằng sau Nhắc nhở tự nhất quán là phản hồi của mô hình đối với lời nhắc tại bất kỳ thời điểm nào phải nhất quán, bất kể khi nào lời nhắc được đưa ra. Sự không nhất quán trong các phản hồi có thể dẫn đến sự ngờ vực và nhầm lẫn cho người dùng, làm cho kỹ thuật này trở thành một công cụ quan trọng trong việc cải thiện chất lượng nội dung do AI tạo ra.

Hiểu đơn giản, ta kết hợp CoT vào prompt để xác định kết quả của một vấn đề, sau đó ta lấy ít nhất là 3 câu trả lời của nó và chọn câu trả lời thường gặp nhất làm kết quả cuối cùng.

Bằng cách tổng hợp nhiều câu trả lời cho cùng một lời nhắc, tính tự nhất quán đảm bảo rằng câu trả lời cuối cùng cho một ý kiến đầu vào thể hiện một cuộc bỏ phiếu đồng thuận, có xu hướng đáng tin cậy và chính xác hơn so với việc hoàn thành chuỗi suy nghĩ riêng lẻ.

Ví dụ: Giả sử bạn là một công ty phần mềm và nhận được hàng trăm email mỗi ngày. Bạn muốn sử dụng mô hình để phân loại email là quan trọng hay không quan trọng, nhờ đó, bạn có thể ưu tiên những email có thể có tác động lớn đến doanh nghiệp của mình.

Dưới đây là ví dụ về email mà bạn có thể nhận được:

Xin chào,

Tôi đã phát hiện ra một lỗ hổng bảo mật lớn trong hệ thống của bạn. Mặc dù nó không dễ sử dụng, nhưng có thể truy cập vào tất cả dữ liệu của người dùng của bạn. Tôi đã đính kèm một bằng chứng thực tế. Vui lòng khắc phục vấn đề này càng sớm càng tốt.

Trân trọng,
Donny

Prompt:

"Xin chào,

Tôi đã phát hiện ra một lỗ hổng bảo mật lớn trong hệ thống của bạn. Mặc dù nó không dễ sử dụng, nhưng có thể truy cập vào tất cả dữ liệu của người dùng của bạn. Tôi đã đính kèm một bằng chứng thực tế. Vui lòng khắc phục vấn đề này càng sớm càng tốt.

Trân trọng,
Donny"

Phân loại email trên là QUAN TRỌNG hay KHÔNG QUAN TRỌNG liên quan đến một công ty phần mềm. Hãy suy nghĩ từng bước.

Output 1:

Người gửi đã phát hiện ra một lỗ hổng bảo mật lớn trong hệ thống.
Điều này rất quan trọng vì nó có thể dẫn đến việc xâm nhập dữ liệu.
Tôi sẽ phân loại email này là QUAN TRỌNG

Output 2:

Người gửi đã phát hiện ra một lỗ hổng bảo mật lớn trong hệ thống, nhưng họ không cung cấp bất kỳ bằng chứng thực tế nào và nó không dễ sử dụng.
Tôi sẽ phân loại email này là KHÔNG QUAN TRỌNG

Output 3:

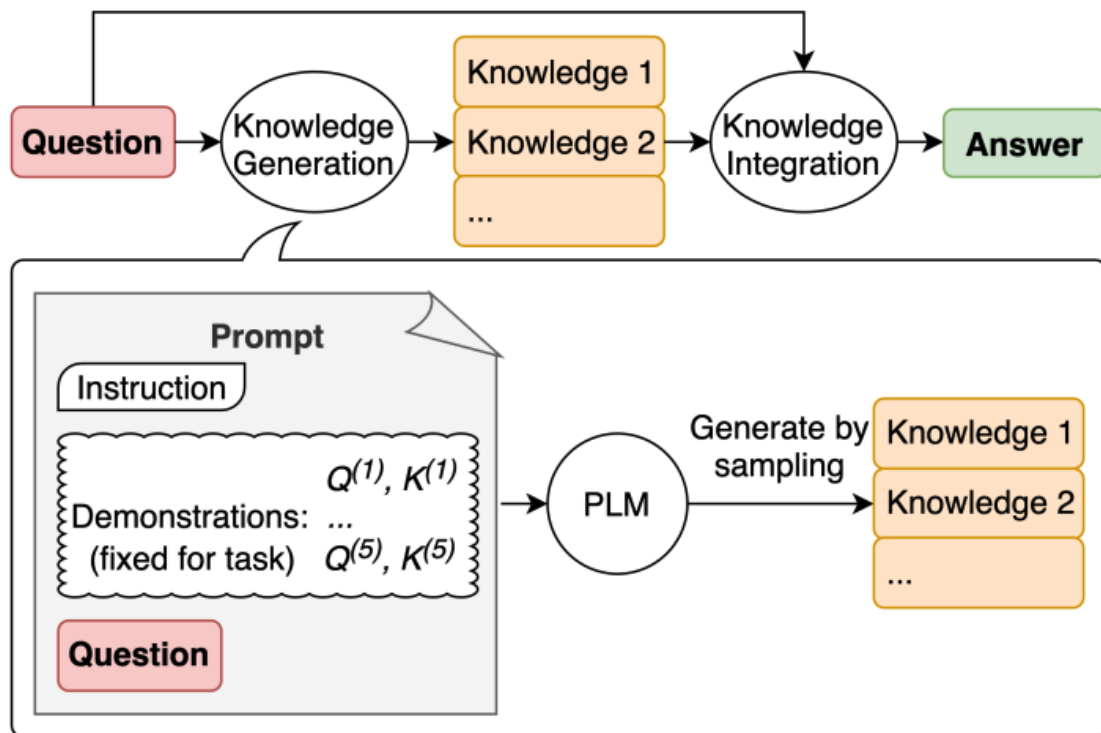
Người gửi có thể là một hacker và có thể có quyền truy cập vào hệ thống.
Điều này rất quan trọng vì nó có thể dẫn đến việc xâm nhập dữ liệu.
Tôi sẽ phân loại email này là QUAN TRỌNG

⇒ Bằng cách tạo ra nhiều chuỗi suy nghĩ và lấy câu trả lời thường gặp nhất (QUAN TRỌNG), chúng ta có thể nhận được câu trả lời chính xác nhất quán hơn từ mô hình.

Link documents:

<https://sh-tsang.medium.com/brief-review-self-consistency-improves-chain-of-thought-reasoning-in-language-models-6471ea9bc00a>

3.6. **Generate Knowledge**



LLM đôi khi sẽ bị hỏng một số kiến thức, có thể do nhiều lý do khác nhau như mô hình chưa được train về kiến thức đó, hoặc kiến thức đó đã bị lỗi thời và chưa được cập nhật lại,... Việc thêm kiến thức giúp cho mô hình hiểu rõ thông tin cần thiết và bổ sung vào những lỗ hổng của nó để cho ra câu trả lời dựa trên những thông tin đó.

Generated Knowledge sẽ được đưa vào mô hình dưới dạng context nên mô hình sẽ có xu hướng ưu tiên những kiến thức trong context hơn những kiến thức được học sẵn của mô hình.

Bên cạnh lợi ích về việc bổ sung thông tin còn thiếu cho mô hình, thì nếu những kiến thức được thêm vào là sai lầm hoặc không chính xác cũng khiến cho câu trả lời cuối cùng không đúng, nên trước khi sử dụng cần đảm bảo thông tin được đưa vào và đúng và tùy trường hợp để sử dụng (ví dụ mô hình đã được học kỹ những kiến thức này rồi thì không nên đặt lại những kiến thức tương tự để tránh lặp lại).

Ví dụ:

Prompt:

Một phần của môn chơi đánh golf là cố gắng đạt được tổng điểm cao hơn những phần khác. Có hay không?

Output:

Có

Loại sai lầm này bộc lộ những hạn chế của LLM trong việc thực hiện các nhiệm vụ đòi hỏi nhiều kiến thức hơn về thế giới. Làm thế nào để chúng ta cải thiện điều này với việc tạo ra kiến thức?

Bước tiếp theo là tích hợp kiến thức và đưa ra dự đoán. Ta sẽ định dạng lại câu hỏi thành định dạng QA để hướng dẫn định dạng câu trả lời.

Prompt 1 (sử dụng Knowledge 1):

Câu hỏi: Một phần của golf là cố gắng có tổng điểm cao hơn người khác. Có hoặc Không?
Kiến thức: Mục tiêu của golf là chơi một loạt lỗ với số cú đánh ít nhất. Một vòng đánh golf thường bao gồm 18 lỗ. Mỗi lỗ được chơi một lần trong vòng đánh trên một sân golf tiêu chuẩn. Mỗi cú đánh được tính là một điểm, và tổng số cú đánh được sử dụng để xác định người chiến thắng trong trò chơi.
Giải thích và Trả lời:

Answer 1 (độ tin cậy rất cao):

Không, một phần của golf không liên quan đến việc cố gắng có tổng điểm cao hơn người khác. Trong golf, mục tiêu là hoàn thành một loạt lỗ với số cú đánh ít nhất, điều này có nghĩa là mục tiêu là có tổng điểm thấp hơn, không phải là cao hơn. Người chơi có điểm thấp nhất, hoặc tổng điểm thấp nhất, là người chiến thắng trò chơi. Vì vậy, mục tiêu trong golf là có tổng điểm thấp nhất, không phải là cao hơn, làm cho câu trả lời là "Không".

Prompt 2 (sử dụng Knowledge 2):

Câu hỏi: Một phần của golf là cố gắng có tổng điểm cao hơn người khác. Có hoặc Không?

Kiến thức: Golf là một môn thể thao đòi hỏi sự chính xác với gậy và bóng, trong đó các cầu thủ cạnh tranh (hoặc gôn thủ) sử dụng nhiều loại gậy để đánh bóng vào một loạt lỗ trên sân bằng số cú đánh ít nhất. Mục tiêu là hoàn thành sân với số điểm thấp nhất, được tính bằng cách cộng tổng số cú đánh trên mỗi lỗ. Người chơi có điểm thấp nhất sẽ chiến thắng trò chơi.

Giải thích và Trả lời:

Answer 2 (độ tin cậy thấp hơn rất nhiều):

Đúng vậy, một phần của golf là cố gắng có điểm số cao hơn người khác. Mỗi người chơi cố gắng hoàn thành sân với số điểm thấp nhất, được tính bằng cách cộng tổng số cú đánh trên mỗi lỗ. Người chơi có số điểm thấp nhất sẽ thắng trò chơi.

Bởi vì kiến thức được đưa vào là khác nhau nên câu trả lời của LLM cũng khác nhau. Nên hãy chú ý trước khi cung cấp kiến thức cho mô hình (đảm bảo kiến thức được đưa vào là đúng).

Trường hợp sử dụng:

- Trường hợp mô hình bị hỏng một số kiến thức như mô hình chưa được train về kiến thức đó, hoặc kiến thức đó đã bị lỗi thời và chưa được cập nhật lại,...
- Trường hợp muốn cung cấp thêm các thông tin cá nhân, doanh nghiệp cụ thể như số điện thoại, địa chỉ, các dịch vụ, các hướng dẫn, menu, chức năng, giới thiệu,...
- Trường hợp chưa hài lòng với câu trả lời của mô hình, muốn thêm các thông tin mới và đặc biệt cho mô hình để có câu trả lời đúng ý định nhất

Trường hợp không nên sử dụng:

- Không chắc chắn kiến thức cung cấp cho mô hình là đúng và phù hợp với mong muốn.

Link documents:

<https://www.promptingguide.ai/techniques/knowledge>

3.7. Prompt Chaining

Việc đưa một prompt rất dài gồm một hoặc nhiều nhiệm vụ phức tạp cùng lúc cho mô hình xử lý, cố gắng nhồi nhét tất cả vào một lời nhắc duy nhất có thể khiến mô hình bị rối và đưa ra câu trả lời không đúng với mong muốn.

Thay vào đó, chúng ta chia nó thành các bước nhỏ hơn, dễ quản lý hơn. Một lời nhắc cụ thể hướng dẫn từng bước và đầu ra từ lời nhắc đó trở thành đầu vào cho lần tiếp theo, tạo thành một chuỗi các hướng dẫn dẫn đến kết quả mong muốn.

Lợi ích:

- Phá vỡ sự phức tạp: Ngay cả các mô hình AI mạnh mẽ cũng có thể bị choáng ngợp. Chaining cung cấp cho mô hình LLM các khối nhỏ hơn để xử lý, cải thiện sự hiểu biết và dẫn đến kết quả tốt hơn.

- Độ chính xác & Điều khiển: Các lời nhắc đơn, bao gồm tất cả cung cấp khả năng kiểm soát hạn chế. Chuỗi cho phép bạn định hình quy trình, đảm bảo mỗi bước phù hợp hoàn hảo với mục tiêu của bạn.

- Tinh chỉnh lặp lại: Bạn có thể phân tích đầu ra từ mỗi bước và điều chỉnh lời nhắc sau cho phù hợp, cho phép tối ưu hóa liên tục.

Ví dụ:

Prompt:

Bạn là một trợ lý hữu ích. Nhiệm vụ của bạn là giúp trả lời một câu hỏi được đưa ra trong một tài liệu. Bước đầu tiên là trích xuất các trích dẫn liên quan đến câu hỏi từ tài liệu, được giới hạn bởi #####. Vui lòng xuất danh sách trích dẫn bằng cách sử dụng <quotes>. Trả lời với "Không tìm thấy trích dẫn liên quan!" nếu không tìm thấy trích dẫn liên quan.

Output:

<quotes>
- Chain-of-thought (CoT) prompting[27]
- Generated knowledge prompting[37]
- Least-to-most prompting[38]

```
- Self-consistency decoding[39]
- Complexity-based prompting[41]
- Self-refine[42]
- Tree-of-thought prompting[43]
- Maieutic prompting[45]
- Directional-stimulus prompting[46]
- Textual inversion and embeddings[59]
- Using gradient descent to search for prompts[61][62][63][64]
- Prompt injection[65][66][67]
</quotes>
```

Prompt 2:

Cho một tập hợp các trích dẫn liên quan (được giới hạn bởi <quotes></quotes>) được trích xuất từ một tài liệu và tài liệu gốc (được giới hạn bởi #####), vui lòng viết câu trả lời cho câu hỏi. Đảm bảo rằng câu trả lời chính xác, có một tông tone thân thiện và nghe có vẻ hữu ích.

####

{{tài liệu}}

####

<quotes>

```
- Chain-of-thought (CoT) prompting[27]
- Generated knowledge prompting[37]
- Least-to-most prompting[38]
- Self-consistency decoding[39]
- Complexity-based prompting[41]
- Self-refine[42]
- Tree-of-thought prompting[43]
- Maieutic prompting[45]
- Directional-stimulus prompting[46]
- Textual inversion and embeddings[59]
- Using gradient descent to search for prompts[61][62][63][64]
- Prompt injection[65][66][67]
</quotes>
```


Output 2:

Các kỹ thuật prompting được đề cập trong tài liệu bao gồm:

1. Chain-of-thought (CoT) prompting[27]
2. Generated knowledge prompting[37]
3. Least-to-most prompting[38]
4. Self-consistency decoding[39]
5. Complexity-based prompting[41]
6. Self-refine[42]
7. Tree-of-thought prompting[43]
8. Maieutic prompting[45]
9. Directional-stimulus prompting[46]
10. Textual inversion and embeddings[59]
11. Using gradient descent to search for prompts[61][62][63][64]
12. Prompt injection[65][66][67]

Mỗi kỹ thuật này sử dụng các chiến lược độc đáo để tăng cường hoặc chỉ định các tương tác với các mô hình ngôn ngữ lớn để tạo ra các kết quả mong muốn.

Trường hợp sử dụng:

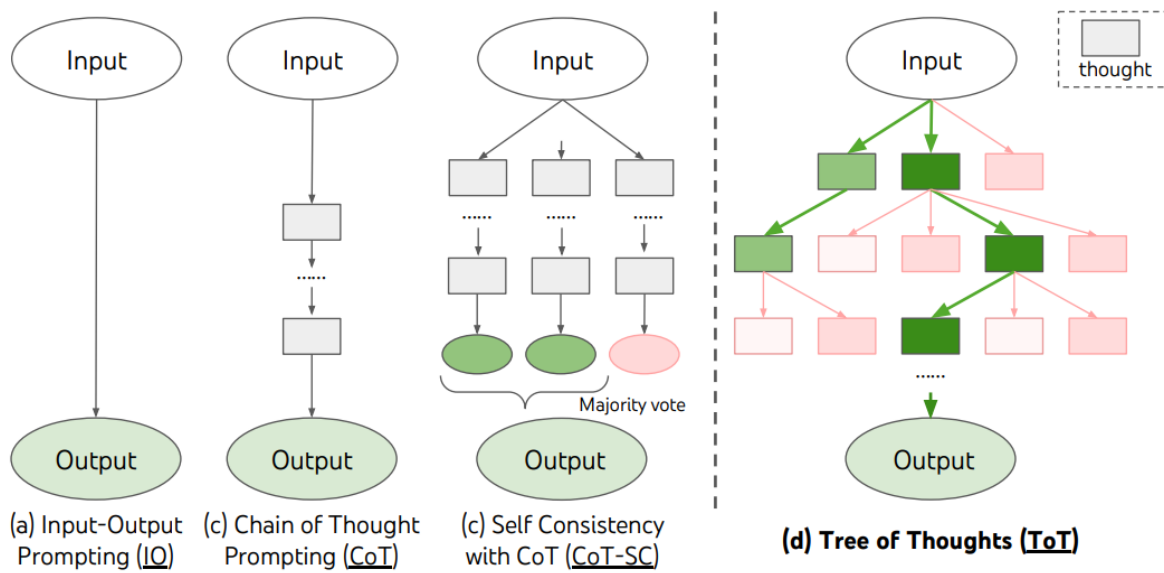
- Các tác vụ phức tạp và dài, cần nhiều bước để thực hiện.
- Chuỗi mà câu trả lời trước là context cho câu sau.

Link documents:

https://www.promptingguide.ai/techniques/prompt_chaining

<https://leonnicholls.medium.com/prompt-chaining-your-gemini-power-up-for-mastering-complex-tasks-8f373614692f>

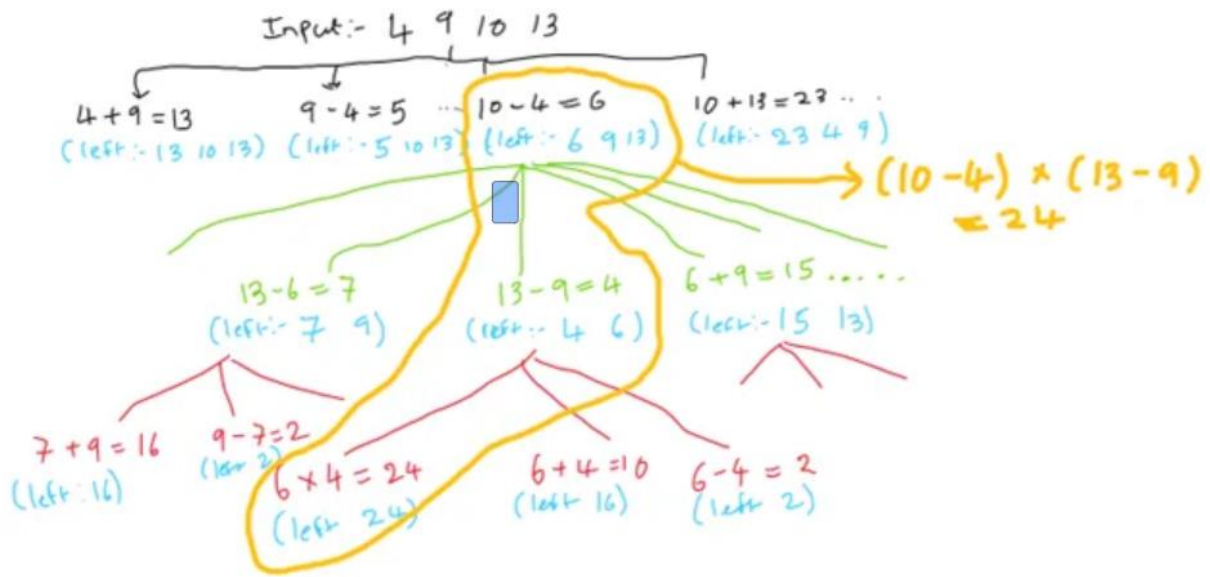
3.8. Tree of Thoughts (ToT)



ToT hoạt động bằng cách tạo ra cấu trúc suy nghĩ giống như cây. Mỗi nút trong cây này thể hiện một suy nghĩ hoặc ý tưởng tiềm năng mà mô hình ngôn ngữ có thể khám phá. Sau đó, mô hình sẽ điều hướng qua cây này, đánh giá từng nút dựa trên khả năng đóng góp của nó cho quá trình giải quyết vấn đề.

Cách tiếp cận này cho phép mô hình ngôn ngữ xem xét đồng thời nhiều con đường lý luận khác nhau. Sau đó nó có thể chọn con đường hứa hẹn nhất dựa trên đánh giá của nó.

ToT cũng cho phép quay lại, có nghĩa là nếu một đường dẫn đã chọn không dẫn đến giải pháp, mô hình có thể theo dõi các bước của nó trở lại nút trước đó và khám phá một đường dẫn khác. Khả năng nhìn trước và lùi lại khi cần thiết là một khía cạnh quan trọng của khuôn khổ ToT, cho phép nó đưa ra các lựa chọn toàn cầu xem xét bối cảnh rộng lớn hơn của vấn đề.



Cách ToT hoạt động:

- Với một vấn đề có các giải pháp và lập kế hoạch không tầm thường (ví dụ: Trò chơi 24, Viết sáng tạo, Trò chơi ô chữ nhỏ), ToT khám phá nhiều cách lập luận khác nhau một cách độc lập.
- Đánh giá từng con đường và lựa chọn một cách tự động để xác định hướng hành động tiếp theo.
- Kết quả trung gian được trình bày bằng ngôn ngữ cấp cao mà con người có thể đọc được.
- Đánh giá nhiều phương án khả thi tiềm năng để giải quyết vấn đề.
- Tiến hành kế hoạch khả thi nhất

Ví dụ:

```
from langchain.chat_models import ChatOpenAI
from langchain.prompts import PromptTemplate
from langchain.schema.runnable import RunnablePassthrough
```

Step 1 - Building LLM to draft solutions

solutions_template = """

Step 1:

I'm facing a challenge related to {topic}. Can you generate two plans for implementing {topic}?

```

"""

solutions_prompt = PromptTemplate(
    input_variables = ["topic"],
    template = solutions_template
)
solutions_llm = ChatOpenAI(openai_api_key = os.getenv('OPENAI_API_KEY'), temperature =
0.1)
solutions_chain = solutions_prompt | solutions_llm


# Step 2 - Build LLM to analyse each solution drafted in the previous step
analysis_template = """
Step 2:
For each of the {solutions}, consider the advantages and disadvantages, ways to integrate them
with existing urban infrastructure and planning strategies.
Your output should be in the form of description of the plan, advantages, disadvantages and
planning strategies in a JSON format for each solution.
"""

analysis_prompt = PromptTemplate(input_variables = ["solutions"],
                                template = analysis_template)
analysis_llm = ChatOpenAI(openai_api_key = os.getenv('OPENAI_API_KEY'), temperature =
0.1)
analysis_chain = analysis_prompt | analysis_llm


# Step 3 - Build LLM to evaluate and rank solutions
evaluation_template = """
Step 3:

Evaluate and rank solutions based on {solutions} and {analysis} in previous steps,
rank the plans on a scale of 10 based on safety, environmental impact, reliability and cost-

```

effectiveness and justify the rank.

Your output should be in the form of description about the plan, advantages, disadvantages, planning strategies,

evaluation with reason in a JSON format for each solution.

```
"""
```

```
evaluation_prompt = PromptTemplate(input_variables = ["solutions", "analysis"],  
                                   template = evaluation_template)
```

```
evaluation_llm = ChatOpenAI(openai_api_key = os.getenv('OPENAI_API_KEY'), temperature =  
0.1)
```

```
evaluation_chain = evaluation_prompt | evaluation_llm
```

Bây giờ, đã đến lúc gọi tất cả các chuỗi riêng lẻ ở bước trước và xây dựng một chuỗi tuần tự trong đó đầu ra của chuỗi đầu tiên được chuyển làm đầu vào cho chuỗi tiếp theo.

```
complete_chain = RunnablePassthrough.assign(solutions = solutions_chain)|
```

```
RunnablePassthrough.assign(analysis = analysis_chain) | evaluation_chain
```

```
response = complete_chain.invoke({"topic": "air taxi"})
```

```
print(response.content)
```

Trường hợp sử dụng:

- Muốn tìm ra phương pháp tối ưu cho các thuật toán.
- Sử dụng kết hợp với prompt chaining để có khả năng quay lại tác vụ cũ để giải quyết lại nếu kết quả không đáp ứng được nhu cầu.

Link documents:

<https://freedium.cfd/https://towardsdatascience.com/tree-of-thoughts-prompting-65a3e51f9ac4>

<https://arxiv.org/pdf/2305.10601v1>

3.9. Retrieval Augmented Generation (RAG)

Retrieval-augmented generation (RAG) là một framework AI để cải thiện chất lượng của các phản hồi do LLM tạo ra bằng cách dựa trên mô hình dựa trên các nguồn kiến thức bên ngoài để bổ sung cho biểu diễn thông tin nội bộ của LLM.

Triển khai RAG trong hệ thống trả lời câu hỏi dựa trên LLM có hai lợi ích chính: Nó đảm bảo rằng mô hình có quyền truy cập vào các sự kiện mới nhất, đáng tin cậy nhất và người dùng có quyền truy cập vào các nguồn của mô hình, đảm bảo rằng các câu trả lời của nó chính xác và đáng tin cậy.

Có 2 loại thông tin được đưa vào, một là thông tin được truy vấn từ các nguồn trên internet (ví dụ wikimedia), hai là thông tin đã được cung cấp từ người dùng và được vector hóa để truy vấn (ví dụ thông tin doanh nghiệp, thông tin khách hàng, địa chỉ, các kiến thức mới,...)

Quá trình tạo ra phần kiến thức cho RAG được chia thành 4 bước:



1) Load Docs

2) Chunk Docs

3) Embed Chunks

4) Load into VDB

- Load docs - Điều này bao gồm việc thu thập một bộ tài liệu và đảm bảo chúng ở định dạng sẵn sàng phân tích cú pháp.
- Chunk docs - Vì LLM có cửa sổ ngữ cảnh hạn chế nên tài liệu phải được chia thành các đoạn nhỏ hơn (ví dụ: dài 256 hoặc 512 ký tự).
- Embed chunks - Dịch từng đoạn thành số bằng cách sử dụng mô hình nhúng văn bản.
- Load vào Vector DB - Tải nội dung nhúng văn bản vào cơ sở dữ liệu (còn gọi là cơ sở dữ liệu vector).

Cốt lõi của cơ sở dữ liệu vector nằm ở vector hóa, là quá trình chuyển đổi mã thông báo (từ, cụm từ hoặc các kiểu dữ liệu khác) thành các vector số bằng cách sử dụng các phép toán đại số tuyến tính. Điều này cho phép cơ sở dữ liệu biểu diễn và so sánh dữ liệu dựa trên các giá trị số. Tương tự, các loại dữ liệu khác như hình ảnh, âm thanh và video có thể được vector hóa

bằng các thuật toán khác nhau. FYI, các mô hình AI tiên tiến nhất có hơn nghìn kích thước cho mỗi mã thông báo.

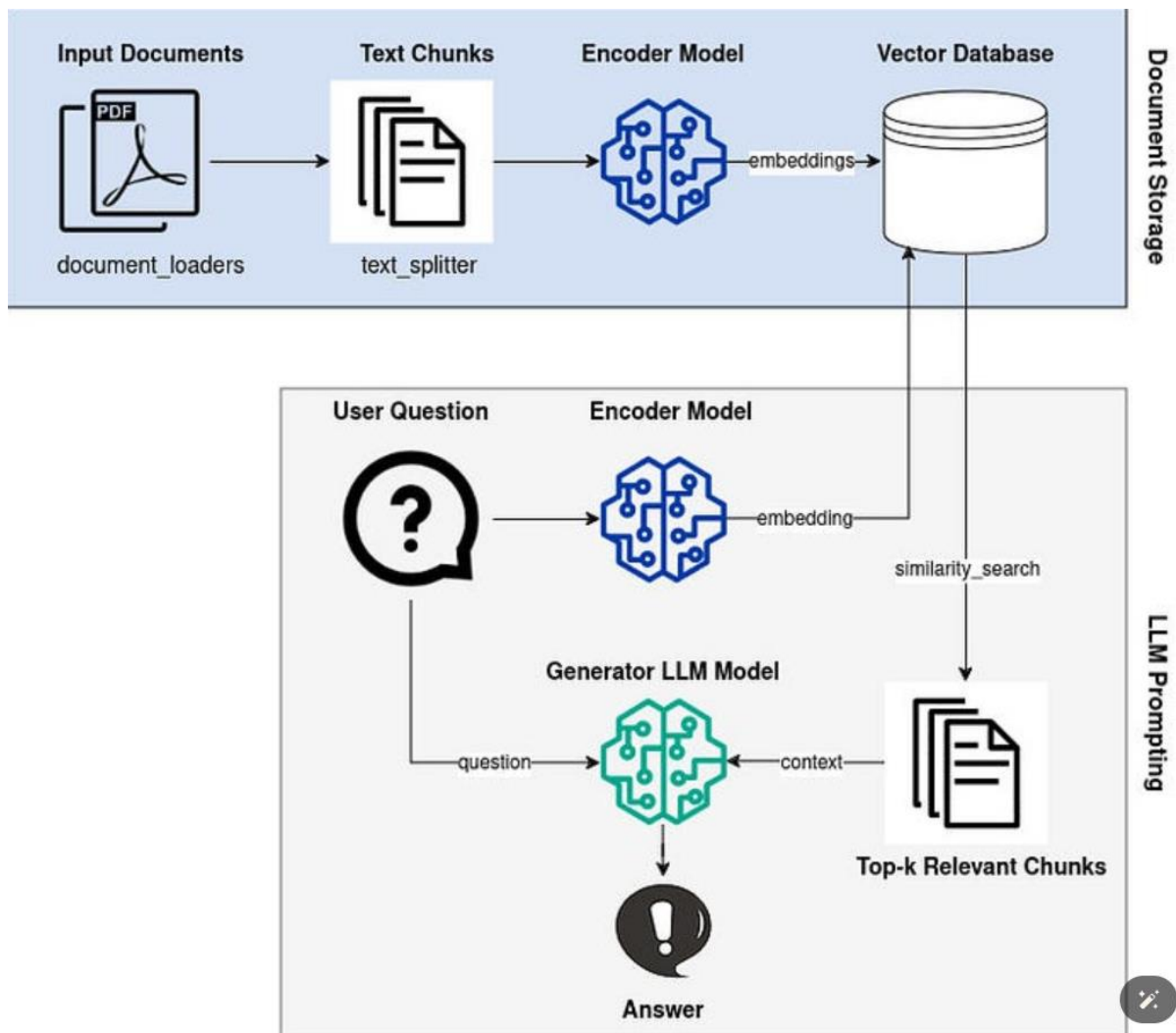
Giả sử một văn bản như "hệ thống định vị cục bộ" có thể được chuyển đổi thành những vector như [0,2341219, 0,02874, 0,824358, 0,125, 0,349876]. Một văn bản có ý nghĩa tương tự sẽ có số tương tự. Ví dụ: "hệ thống định vị trong nhà" có thể có những như [0.247865, 0.020274, 0.88236, 0.11455, 0.324628]. Hướng của hai vector five-dimensional này sẽ tương tự nhau, do đó làm cho giá trị cosign của nó về góc của hai vector này gần với một. Cosign(0) bằng 1. Giá trị cosign của góc của hai vector này có thể được chia bằng cách chia tích chập của hai vector với tích của độ dài của mỗi vector.

Từ dữ liệu được chuyển thành vector ở trên, ta đưa nó cho RAG để biến nó thành dạng Context của mô hình.

Chatbots dựa trên Mô hình ngôn ngữ lớn (LLM) có thể được cải thiện bằng cách cung cấp kiến thức bên ngoài thông qua Hệ thống tăng cường truy xuất (RAG).

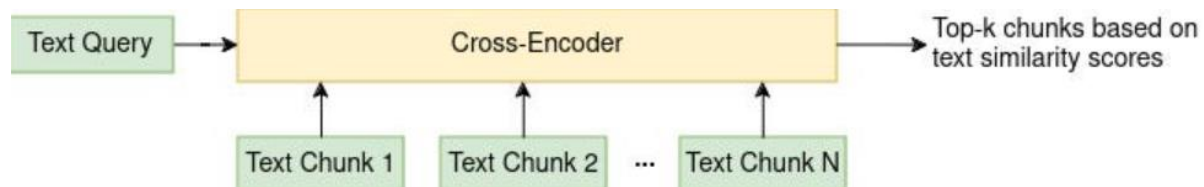
Kiến thức bên ngoài này có thể làm giảm các câu trả lời sai (ảo giác) và cũng cung cấp cho mô hình quyền truy cập vào thông tin không có trong dữ liệu huấn luyện của nó.

Với RAG, chúng ta cung cấp thông tin LLM của mình, chẳng hạn như tài liệu PDF hoặc bài viết Wikipedia, dưới dạng ngữ cảnh bổ sung trong lời nhắc của chúng ta.



Một cải tiến đối với quy trình RAG cơ bản là việc sử dụng trình xếp hạng lại (re-ranker). Re-ranker lấy câu hỏi của người dùng và tất cả các tài liệu được truy xuất ban đầu làm đầu vào và xếp hạng lại các tài liệu này dựa trên mức độ chúng khớp với câu hỏi.

Nhiệm vụ của re-rank là ước tính điểm về mức độ liên quan của từng văn bản ứng viên với một truy vấn văn bản nhất định. Không giống như bộ mã hóa hai chiều, bộ mã hóa chéo xử lý đồng thời hai cặp văn bản đầu vào.



Ví dụ:

Load dữ liệu và tách nó thành nhiều đoạn, sau đó embedding dữ liệu đó để load nó vào vector database chroma.

```
splitter = RecursiveJsonSplitter(max_chunk_size=256)
embedding_function = SentenceTransformerEmbeddings(model_name="all-MiniLM-L6-v2")

documents = []
for small_json in data:
    doc = splitter.create_documents(texts=[small_json])
    documents.append(doc[0])
print(documents)

db = Chroma.from_documents(documents=documents, embedding=embedding_function,
persist_directory=persist_directory)
```

Load datavector từ chroma.

```
def load_chroma_data(persist_directory):
    db = Chroma(persist_directory=persist_directory, embedding_function=embedding_function)

    return db
```

Truy vấn từ dữ liệu vector đó.

```
def similarity_search_chroma_data(db, query):
    docs = db.similarity_search(query=query)

    result = docs[0].page_content
    return result
```

Trường hợp sử dụng:

- Truy vấn dữ liệu từ bên ngoài mô hình

- Tránh quá số lượng token cho phép của LLM khi Knowledge được đưa vào dưới dạng context quá nhiều. RAG sẽ chỉ truy vấn những dữ liệu liên quan đến câu hỏi và cho ra các đoạn tương ứng (k=3 thì sẽ có 3 đoạn được trả về), mỗi đoạn có độ dài dựa vào chunk size (thường là 256 hoặc 512)

Link documents:

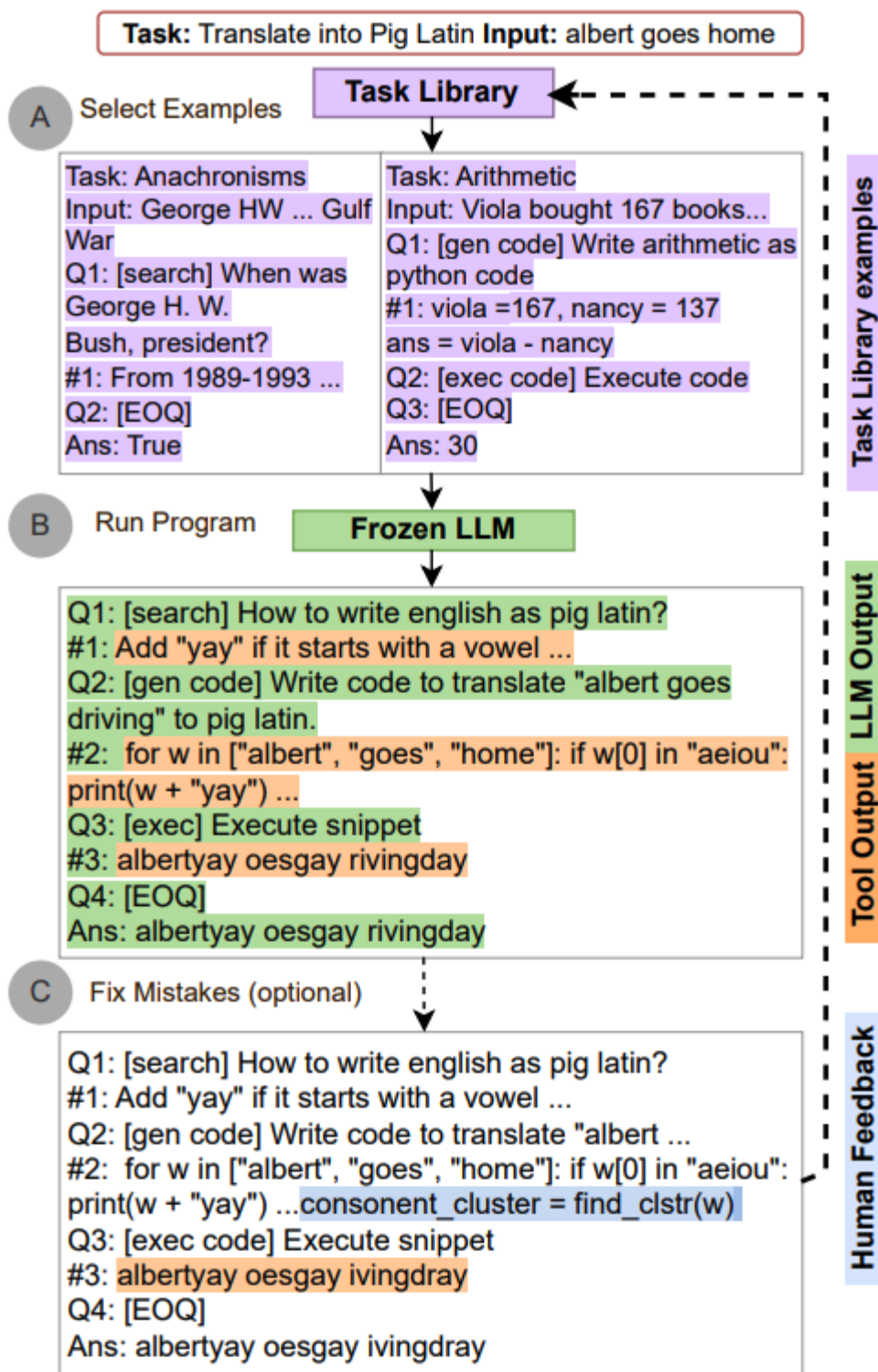
<https://arxiv.org/pdf/2312.10997>

3.10. Automatic Reasoning and Tool-use (ART)

Về cốt lõi, ART xoay quanh việc trao quyền cho máy móc với khả năng không chỉ xử lý mà còn hiểu thực sự các lệnh hoặc lời nhắc theo cách phản ánh lý luận của con người. Khía cạnh này của AI vượt qua các cơ chế phản hồi lệnh thông thường, mở ra một kỷ nguyên mà máy móc có thể suy luận, suy luận và thậm chí dự đoán nhu cầu đằng sau các yêu cầu của chúng ta. Hãy tưởng tượng yêu cầu thiết bị sắp xếp lịch trình của bạn và nó không chỉ làm như vậy mà còn gợi ý thời điểm tốt nhất để nghỉ giải lao dựa trên các tùy chọn trong quá khứ. Đó là ART đang hoạt động.

...

3.11. Automatic Prompt Engineer (APE)



Theo bài báo, Kỹ sư nhắc nhở tự động (APE) là một thuật toán mới tạo và chọn các hướng dẫn ngôn ngữ tự nhiên tự động để giảm nỗ lực của con người liên quan đến việc tạo và xác nhận các hướng dẫn hiệu quả. APE coi việc tạo hướng dẫn là tổng hợp chương trình ngôn ngữ tự nhiên,

đóng khung nó như một vấn đề tối ưu hóa hộp đen được hướng dẫn bởi LLM. Thuật toán sử dụng LLM theo ba cách: như một mô hình suy luận, để hướng dẫn quá trình tìm kiếm và cải thiện lặp đi lặp lại các ứng cử viên với các biến thể hướng dẫn tương tự về mặt ngữ nghĩa.

Dưới đây là tóm tắt về quy trình:

- Cung cấp một vài cặp đầu vào-đầu ra làm ví dụ cho LLM.
- Tạo lời nhắc và các biến thể của lời nhắc bằng LLM.
- Chấm điểm các biến thể và chọn ra ứng cử viên hàng đầu có số điểm cao nhất.

Ví dụ:

```
#@title Install Dependencies
! pip install git+https://github.com/keirp/automatic_prompt_engineer

pip install openai

import openai
openai.api_key = 'xxxxxxxxxxxxxxxxxxxxxx'

# First, let's define a simple dataset consisting of words and their
antonyms.
words = ["sane", "direct", "informally", "unpopular", "subtractive",
"nonresidential",
        "inexact", "uptown", "incomparable", "powerful", "gaseous", "evenly",
"formality",
        "deliberately", "off"]
antonyms = ["insane", "indirect", "formally", "popular", "additive",
"residential",
        "exact", "downtown", "comparable", "powerless", "solid", "unevenly",
"informality",
        "accidentally", "on"]

# Now, we need to define the format of the prompt that we are using.
eval_template = \
"""Instruction: [PROMPT]
Input: [INPUT]
Output: [OUTPUT]"""

# Now, let's use APE to find prompts that generate antonyms for each word.
from automatic_prompt_engineer import ape

result, demo_fn = ape.simple_ape(
    dataset=(words, antonyms),
    eval_template=eval_template,
)
```

```
# Let's see the results.  
print(result)
```

Và dưới đây là danh sách các lời nhắc có thể được liệt kê theo điểm số.

```
score: prompt  
-----  
-0.24: give the opposite of the word provided.  
-0.25: produce an antonym for each word provided.  
-0.27: produce an antonym (opposite) for each word given.  
-0.28: "change all adjectives to their antonyms."  
-0.29: produce an antonym for each word given.  
-0.31: produce an input-output pair in which the output is the opposite of  
the input.  
-0.33: use an online thesaurus to find a word with the opposite meaning.  
-0.51: produce the opposite of the input.  
-0.63: make a list of antonyms.  
-0.85: "find the opposite of each word."
```

Code dưới đây có thể được sử dụng để chấm điểm lời nhắc của con người hoặc được nhập thủ công.

```
log(p): prompt  
-----  
-0.24: Write an antonym to the following word.
```

Link documents:

<https://cobusgreyling.medium.com/automatic-prompt-engineer-ui-aa7dd0770dd1>

<https://cobusgreyling.medium.com/automatic-prompt-engineering-907e230ece0>

https://github.com/keirp/automatic_prompt_engineer/

<https://freedium.cfd/https://blog.gopenai.com/unleashing-the-power-of-ai-to-self-generate-prompts-a-dive-into-the-automatic-prompt-engineer-d87f2db83f56>

3.12. *Active-Prompt*



Active-Prompt là kỹ thuật tạo ra các prompt chủ động để kích thích phản hồi cụ thể từ mô hình ngôn ngữ. Điều này giúp mô hình tạo ra các kết quả rõ ràng và cụ thể hơn. Kỹ thuật này quan trọng khi bạn cần kiểm soát chặt chẽ các câu trả lời của mô hình.

3.13. *Directional Stimulus Prompting*

Directional Stimulus Prompting hướng dẫn mô hình theo một hướng cụ thể bằng cách cung cấp các gợi ý hoặc dấu hiệu định hướng. Điều này hữu ích trong các ngữ cảnh mà bạn cần mô hình tập trung vào một khía cạnh hoặc chủ đề cụ thể của câu hỏi.

3.14. *PAL (Program-Aided Language Models)*

PAL là sự kết hợp giữa các mô hình ngôn ngữ và chương trình để tăng cường khả năng suy luận và đưa ra câu trả lời phức tạp. Đây là kỹ thuật quan trọng khi cần các kết quả có tính logic cao và yêu cầu tính toán hoặc suy luận phức tạp.

Chain-of-Thought (Wei et al., 2022)	Program-aided Language models (this work)
<p>Input</p> <p>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?</p> <p>A: Roger started with 5 tennis balls. 2 cans of 3 tennis balls each is 6 tennis balls. $5 + 6 = 11$. The answer is 11.</p> <p>Q: The bakers at the Beverly Hills Bakery baked 200 loaves of bread on Monday morning. They sold 93 loaves in the morning and 39 loaves in the afternoon. A grocery store returned 6 unsold loaves. How many loaves of bread did they have left?</p> <p>Model Output</p> <p>A: The bakers started with 200 loaves. They sold 93 in the morning and 39 in the afternoon. So they sold $93 + 39 = 132$ loaves. The grocery store returned 6 loaves. So they had $200 - 132 - 6 = 62$ loaves left. The answer is 62.</p> 	<p>Input</p> <p>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?</p> <p>A: Roger started with 5 tennis balls. <code>tennis_balls = 5</code> 2 cans of 3 tennis balls each is <code>bought_balls = 2 * 3</code> tennis balls. The answer is <code>answer = tennis_balls + bought_balls</code></p> <p>Q: The bakers at the Beverly Hills Bakery baked 200 loaves of bread on Monday morning. They sold 93 loaves in the morning and 39 loaves in the afternoon. A grocery store returned 6 unsold loaves. How many loaves of bread did they have left?</p> <p>Model Output</p> <p>A: The bakers started with 200 loaves <code>loaves_baked = 200</code> They sold 93 in the morning and 39 in the afternoon <code>loaves_sold_morning = 93</code> <code>loaves_sold_afternoon = 39</code> The grocery store returned 6 loaves. <code>loaves_returned = 6</code> The answer is <code>answer = loaves_baked - loaves_sold_morning</code> <code>- loaves_sold_afternoon + loaves_returned</code></p> <p><code>>>> print(answer)</code> 74</p> 

Ví dụ:

```
question = "Today is 27 February 2023. I was born exactly 25 years ago. What is the date I was born in MM/DD/YYYY?"
```

```
DATE_UNDERSTANDING_PROMPT = ""
```

```
# Q: 2015 is coming in 36 hours. What is the date one week from today in MM/DD/YYYY?
```

```
# If 2015 is coming in 36 hours, then today is 36 hours before.
```

```
today = datetime(2015, 1, 1) - relativedelta(hours=36)
```

```
# One week from today,
```

```
one_week_from_today = today + relativedelta(weeks=1)
```

```
# The answer formatted with %m/%d/%Y is
```

```
one_week_from_today.strftime('%m/%d/%Y')
```

```
# Q: The first day of 2019 is a Tuesday, and today is the first Monday of 2019. What is the date today in MM/DD/YYYY?
```

```
# If the first day of 2019 is a Tuesday, and today is the first Monday of 2019, then today is 6 days later.
```

```
today = datetime(2019, 1, 1) + relativedelta(days=6)
```

```
# The answer formatted with %m/%d/%Y is
```

```
today.strftime('%m/%d/%Y')
```

```
# Q: The concert was scheduled to be on 06/01/1943, but was delayed by one
```

```

day to today. What is the date 10 days ago in MM/DD/YYYY?
# If the concert was scheduled to be on 06/01/1943, but was delayed by one
day to today, then today is one day later.
today = datetime(1943, 6, 1) + relativedelta(days=1)
# 10 days ago,
ten_days_ago = today - relativedelta(days=10)
# The answer formatted with %m/%d/%Y is
ten_days_ago.strftime('%m/%d/%Y')
# Q: It is 4/19/1969 today. What is the date 24 hours later in MM/DD/YYYY?
# It is 4/19/1969 today.
today = datetime(1969, 4, 19)
# 24 hours later,
later = today + relativedelta(hours=24)
# The answer formatted with %m/%d/%Y is
today.strftime('%m/%d/%Y')
# Q: Jane thought today is 3/11/2002, but today is in fact Mar 12, which is 1
day later. What is the date 24 hours later in MM/DD/YYYY?
# If Jane thought today is 3/11/2002, but today is in fact Mar 12, then today
is 3/12/2002.
today = datetime(2002, 3, 12)
# 24 hours later,
later = today + relativedelta(hours=24)
# The answer formatted with %m/%d/%Y is
later.strftime('%m/%d/%Y')
# Q: Jane was born on the last day of February in 2001. Today is her 16-year-
old birthday. What is the date yesterday in MM/DD/YYYY?
# If Jane was born on the last day of February in 2001 and today is her 16-
year-old birthday, then today is 16 years later.
today = datetime(2001, 2, 28) + relativedelta(years=16)
# Yesterday,
yesterday = today - relativedelta(days=1)
# The answer formatted with %m/%d/%Y is
yesterday.strftime('%m/%d/%Y')
# Q: {question}
"".strip() + '\n'

```

```

llm_out = llm(DATE_UNDERSTANDING_PROMPT.format(question=question))
print(llm_out)

```

Kết quả sẽ như sau:

```

# If today is 27 February 2023 and I was born exactly 25 years ago, then I
was born 25 years before.
today = datetime(2023, 2, 27)
# I was born 25 years before,
born = today - relativedelta(years=25)
# The answer formatted with %m/%d/%Y is
born.strftime('%m/%d/%Y')

```

Nội dung của llm_out là một đoạn mã Python. Dưới đây, exec lệnh được sử dụng để thực thi đoạn mã Python này:


```
exec(llm_out)
print(born)
```

Nó này sẽ xuất ra kết quả như sau: 02/27/1998

Link documents:

<https://www.promptingguide.ai/techniques/pal>

3.15. ReAct Prompting

ReAct Prompting kết hợp giữa phản ứng nhanh và hành động có kế hoạch trong việc viết prompt. Điều này giúp mô hình phản hồi một cách tự nhiên hơn, giống như cuộc hội thoại thực sự, và có thể lập kế hoạch các hành động tiếp theo. Kỹ thuật này rất quan trọng trong các ứng dụng tương tác liên tục.

Quá trình giải quyết vấn đề:

Là một con người, chúng ta thường theo mô hình suy nghĩ-hành động-quan sát để giải quyết vấn đề. Hãy xem xét ví dụ đơn giản sau:

- Suy nghĩ: Tôi muốn làm bánh Sandwich nên tôi cần kiểm tra xem mình có công thức hay không.
- Hành động: [Tìm kiếm] công thức làm bánh Sandwich.
- Quan sát: Để chuẩn bị món Sandwich, tôi cần hành tây và cà chua. Vì vậy, tôi nhận ra rằng mình cần đặt hàng hành tây và cà chua trực tuyến.
- Suy nghĩ: Tôi có thể đặt hành tây và cà chua trực tuyến.
- Hành động: [Tìm kiếm] đặt hàng hành tây và cà chua trực tuyến.
- Quan sát: Tôi đã mua thành công hành tây và cà chua thông qua đặt hàng trực tuyến.
- Suy nghĩ: Bây giờ, tôi đã có mọi thứ cần thiết để chuẩn bị một chiếc Sandwich.
- Hành động: [Kết thúc] Tôi có thể bắt đầu chuẩn bị cho món Sandwich.

Bằng cách làm theo mô hình quan sát suy nghĩ-hành động-quan sát này, tôi, với tư cách là một con người (tác nhân), có thể giải quyết vấn đề một cách có hệ thống, đảm bảo rằng tôi thực hiện các hành động cần thiết và đánh giá kết quả ở mỗi bước. Một tác nhân đề cập đến một người hoặc thực thể có khả năng đưa ra quyết định và thực hiện các hành động để ứng phó với một tình huống hoặc vấn đề nhất định. Trong bối cảnh này, tôi là người chủ động suy nghĩ, hành động và quan sát để giải quyết vấn đề trước mắt.

Giống như mô hình quan sát suy nghĩ-hành động-được con người sử dụng, ReAct Prompting tuân theo cách tiếp cận có cấu trúc tương tự để giải quyết vấn đề trong lĩnh vực trí

tuệ nhân tạo. Nó kết hợp liền mạch lý luận và hành động, tạo ra một vòng phản hồi động cho phép các mô hình AI giải quyết các nhiệm vụ phức tạp với độ chính xác và khả năng thích ứng.

(1) Hotspot QA

Question: Aside from the Apple Remote, what other device can control the program Apple Remote was originally designed to interact with?

(1a) Standard

Answer: iPod ✗

(1c) Act-Only

Act 1: Search[Apple Remote]
Obs 1: The Apple Remote is a remote control ...

Act 2: Search[Front Row]
Obs 2: Could not find [Front Row]. Similar: ...

Act 3: Search[Front Row (software)]
Obs 3: Front Row is a discontinued media center software ...

Act 4: Finish[yes] ✗

(1b) CoT (Reason Only)

Thought: Let's think step by step. Apple Remote was originally designed to interact with Apple TV. Apple TV can be controlled by iPhone, iPad, and iPod Touch. So the answer is iPhone, iPad, and iPod Touch. ✗

Answer: iPhone, iPad, iPod Touch ✗

(1d) ReAct (Reason + Act)

Thought 1: I need to search Apple Remote and find the program it was originally designed to interact with.
Act 1: Search[Apple Remote]
Obs 1: The Apple Remote is a remote control introduced in October 2005 by Apple ... originally designed to control the Front Row media center program ...

Thought 2: Apple Remote was originally designed to control the Front Row media center program. I need to search Front Row next and find what other device can control it.
Act 2: Search[Front Row]
Obs 2: Could not find [Front Row]. Similar: ('Front Row Seat to Earth', 'Front Row Motorsports', 'Front Row (software)', ...)

Thought 3: Front Row is not found. I need to search Front Row (software) .
Act 3: Search[Front Row (software)]
Obs 3: Front Row is a discontinued media center software ...

Thought 4: Front Row (software) is controlled by an Apple Remote or the keyboard function keys. So the answer is keyboard function keys.
Act 4: Finish[keyboard function keys] ✓

(2) AlfWorld

You are in the middle of a room. Looking quickly around you, you see a cabinet 6, a cabinet 1, a coffee machine 1, a countertop 3, a stove burner 1, and a toaster 1. Your task is to: Put some pepper shaker on a drawer.

(2a) Act-Only

Act 1: Go to drawer 1
Obs 1: The drawer 1 is closed.

Act 2: Open drawer 1
Obs 2: You open the drawer 1. The drawer 1 is open. In it, you see a dish sponge 2, and a spoon 1.

Act 3: Go to sinkbasin 1
Obs 3: On the sinkbasin 1, you see a dish sponge 3, a spatula 1, and a spoon 2.

Act 4: Take peppershaker 1 from sinkbasin 1
Obs 4: Nothing happens.

Act 5: Take peppershaker 1 from sinkbasin 1
Obs 5: Nothing happens. ✗

(2b) ReAct (Reason + Act)

Act 1: Think[First I need to find a pepper shaker...more likely to appear in cabinets (1-6), countertops (1-3), ...]
Act 2: Go to cabinet 1
Obs 2: On the cabinet 1, you see a vase 2.
(Here the agent go to cabinet 1, then cabinet 2, 3, then countertop 1 and 2)

Act 6: Go to countertop 3
Obs 6: On the countertop 3, you see a apple 1, a bread 1, a pepper shaker 1, and a vase 3.

Act 7: Take pepper shaker 1 from countertop 3
Obs 7: You pick up the pepper shaker 1 from the countertop 3.

Act 8: Think[Now I find a pepper shaker 1. Next, I need to put it in/on drawer 1.]
Act 9: Go to drawer 1
Obs 9: Drawer 1 is closed.

Act 10: Open drawer 1
Obs 10: You open Drawer 1 ...

Act 11: Put pepper shaker 1 in/on drawer 1
Obs 11: You put pepper shaker 1 in/on the drawer 1. ✓

Ví dụ:

Chúng ta sẽ sử dụng việc triển khai ReAct của Langchain để tìm câu trả lời cho vấn đề đầu tiên

```
!pip install --upgrade openai
!pip install --upgrade langchain
!pip install wikipedia
!pip install google-search-results
!pip install --upgrade python-dotenv

# import libraries
import openai
import os
from langchain.llms import OpenAI
from langchain.agents import load_tools
```

```

from langchain.agents import initialize_agent
from dotenv import load_dotenv
load_dotenv()
from langchain.utilities import WikipediaAPIWrapper

# load API keys; you will need to obtain these if you haven't yet
os.environ["OPENAI_API_KEY"] = getpass()

from langchain import OpenAI, Wikipedia
from langchain.agents import initialize_agent, Tool
from langchain.agents import AgentType
from langchain.agents.react.base import DocstoreExplorer
docstore=DocstoreExplorer(Wikipedia())
tools = [
    Tool(
        name="Search",
        func=docstore.search,
        description="useful for when you need to ask with search"
    ),
    Tool(
        name="Lookup",
        func=docstore.lookup,
        description="useful for when you need to ask with lookup"
    )
]

llm = OpenAI(temperature=0, model_name="gpt-3.5-turbo")
react = initialize_agent(tools, llm, agent=AgentType.REACT_DOCSTORE,
verbose=True)

react.run("Author David Chanoff has collaborated with a U.S. Navy admiral who
served as the ambassador to the United Kingdom under which President?")

```

Kết quả:

> Entering new AgentExecutor chain...

Thought: I need to search David Chanoff and find the U.S. Navy admiral he collaborated with, then find out which President the admiral served as the ambassador to the United Kingdom under.

Action: Search[David Chanoff]

Observation: David Chanoff is a noted author of non-fiction work. His work has typically involved collaborations with the principal protagonist of the work concerned. His collaborators have included; Augustus A. White, Joycelyn Elders, Đoàn Văn Toại, William J. Crowe, Ariel Sharon, Kenneth Good and Felix Zandman. He has also written about a wide range of subjects including literary history, education and foreign for The Washington Post, The New Republic and The New York Times Magazine. He has published more than twelve books.

Thought: David Chanoff has collaborated with William J. Crowe. I need to find out which President William J. Crowe served as the ambassador to the United Kingdom under.

Action: Search[William J. Crowe]

Observation: William James Crowe Jr. (January 2, 1925 – October 18, 2007) was a United States Navy admiral and diplomat who served as the 11th chairman of the Joint Chiefs of Staff under Presidents Ronald Reagan and George H. W. Bush, and as the ambassador to the United Kingdom and Chair of the Intelligence Oversight Board under President Bill Clinton.

Thought: William J. Crowe served as the ambassador to the United Kingdom under President Bill Clinton. So the answer is Bill Clinton.

Action: Finish[Bill Clinton]

> Finished chain.

'Bill Clinton'

Về cốt lõi, ReAct tận dụng sức mạnh của các mẫu giải quyết vấn đề, được thiết kế đặc biệt để làm việc với các ví dụ ít chụp. Các mẫu này đóng vai trò là giàn giáo mà ReAct xây dựng năng lực giải quyết vấn đề của mình. Prompt sẽ có cấu trúc như sau:

Question: What is the elevation range for the area that the eastern sector of the Colorado orogeny extends into?

Thought: I need to search Colorado orogeny, find the area that the eastern sector of the Colorado orogeny extends into, then find the elevation range of the area.

Action: Search[Colorado orogeny]

Observation: The Colorado orogeny was an episode of mountain building (an orogeny) in Colorado and surrounding areas.

Thought: It does not mention the eastern sector. So I need to look up eastern sector.

Action: Lookup[eastern sector]

Observation: (Result 1 / 1) The eastern sector extends into the High Plains and is called the Central Plains orogeny.

Thought: The eastern sector of Colorado orogeny extends into the High Plains. So I need to search High Plains and find its elevation range.

Action: Search[High Plains]

Observation: High Plains refers to one of two distinct land regions

Thought: I need to instead search High Plains (United States).

Action: Search[High Plains (United States)]
Observation: The High Plains are a subregion of the Great Plains. From east to west, the High Plains rise in elevation from around 1,800 to 7,000 ft (550 to 2,130 m).[3]

Thought: High Plains rise in elevation from around 1,800 to 7,000 ft, so the answer is 1,800 to 7,000 ft.

Action: Finish[1,800 to 7,000 ft]

Question: Musician and satirist Allie Goertz wrote a song about the "The Simpsons" character Milhouse, who Matt Groening named after who?

Thought: The question simplifies to "The Simpsons" character Milhouse is named after who. I only need to search Milhouse and find who it is named after.

Action: Search[Milhouse]

Observation: Milhouse Mussolini Van Houten is a recurring character in the Fox animated television series The Simpsons voiced by Pamela Hayden and created by Matt Groening.

Thought: The paragraph does not tell who Milhouse is named after, maybe I can look up "named after".

Action: Lookup[named after]

Observation: (Result 1 / 1) Milhouse was named after U.S. president Richard Nixon, whose middle name was Milhous.

Thought: Milhouse was named after U.S. president Richard Nixon, so the answer is Richard Nixon.

Action: Finish[Richard Nixon]

Question: Which documentary is about Finnish rock groups, Adam Clayton Powell or The Saimaa Gesture?

Thought: I need to search Adam Clayton Powell and The Saimaa Gesture, and find which documentary is about Finnish rock groups.

Action: Search[Adam Clayton Powell]

Observation: Could not find [Adam Clayton Powell]. Similar: ['Adam Clayton Powell III', 'Seventh Avenue (Manhattan)', 'Adam Clayton Powell Jr. State Office Building', 'Isabel Washington Powell', 'Adam Powell', 'Adam Clayton Powell (film)', 'Giancarlo Esposito'].

Thought: To find the documentary, I can search Adam Clayton Powell (film).

Action: Search[Adam Clayton Powell (film)]

Observation: Adam Clayton Powell is a 1989 American documentary film directed by Richard Kilberg. The film is about the rise and fall of influential African-American politician Adam Clayton Powell Jr.[3][4] It was later aired as part of the PBS series The American Experience.

Thought: Adam Clayton Powell (film) is a documentary about an African-American politician, not Finnish rock groups. So the documentary about Finnish rock groups must instead be The Saimaa Gesture.

Action: Finish[The Saimaa Gesture]

Question: What profession does Nicholas Ray and Elia Kazan have in common?

Thought: I need to search Nicholas Ray and Elia Kazan, find their professions, then find the profession they have in common.

Action: Search[Nicholas Ray]

Observation: Nicholas Ray (born Raymond Nicholas Kienzle Jr., August 7, 1911 – June 16, 1979) was an American film director, screenwriter, and actor best known for the 1955 film Rebel Without a Cause.

Thought: Professions of Nicholas Ray are director, screenwriter, and actor. I need to search Elia Kazan next and find his professions.

Action: Search[Elia Kazan]

Observation: Elia Kazan was an American film and theatre director, producer,

```

screenwriter and actor.
Thought: Professions of Elia Kazan are director, producer, screenwriter, and actor. So profession Nicholas Ray and Elia Kazan have in common is director, screenwriter, and actor.
Action: Finish[director, screenwriter, actor]

Question: Which magazine was started first Arthur's Magazine or First for Women?
Thought: I need to search Arthur's Magazine and First for Women, and find which was started first.
Action: Search[Arthur's Magazine]
Observation: Arthur's Magazine (1844-1846) was an American literary periodical published in Philadelphia in the 19th century.
Thought: Arthur's Magazine was started in 1844. I need to search First for Women next.
Action: Search[First for Women]
Observation: First for Women is a woman's magazine published by Bauer Media Group in the USA.[1] The magazine was started in 1989.
Thought: First for Women was started in 1989. 1844 (Arthur's Magazine) < 1989 (First for Women), so Arthur's Magazine was started first.
Action: Finish[Arthur's Magazine]

Question: Were Pavel Urysohn and Leonid Levin known for the same type of work?
Thought: I need to search Pavel Urysohn and Leonid Levin, find their types of work, then find if they are the same.
Action: Search[Pavel Urysohn]
Observation: Pavel Samuilovich Urysohn (February 3, 1898 - August 17, 1924) was a Soviet mathematician who is best known for his contributions in dimension theory.
Thought: Pavel Urysohn is a mathematician. I need to search Leonid Levin next and find its type of work.
Action: Search[Leonid Levin]
Observation: Leonid Anatolievich Levin is a Soviet-American mathematician and computer scientist.
Thought: Leonid Levin is a mathematician and computer scientist. So Pavel Urysohn and Leonid Levin have the same type of work.
Action: Finish[yes]

Question: {input}
(agent_scratchpad)

```

Tóm lại, ReAct prompting framework là:

- Thiết kế trực quan: ReAct đơn giản hóa việc tạo prompt, loại bỏ nhu cầu định dạng phức tạp và lựa chọn ví dụ. Nó giống như vẽ một bản đồ đơn giản trong khi điều hướng địa hình không quen thuộc.
- Chung và linh hoạt: ReAct thích ứng liền mạch với các nhiệm vụ khác nhau, phù hợp với các không gian hành động và nhu cầu lý luận riêng biệt. Đó là chiếc xe địa hình của bạn cho các ứng dụng AI.

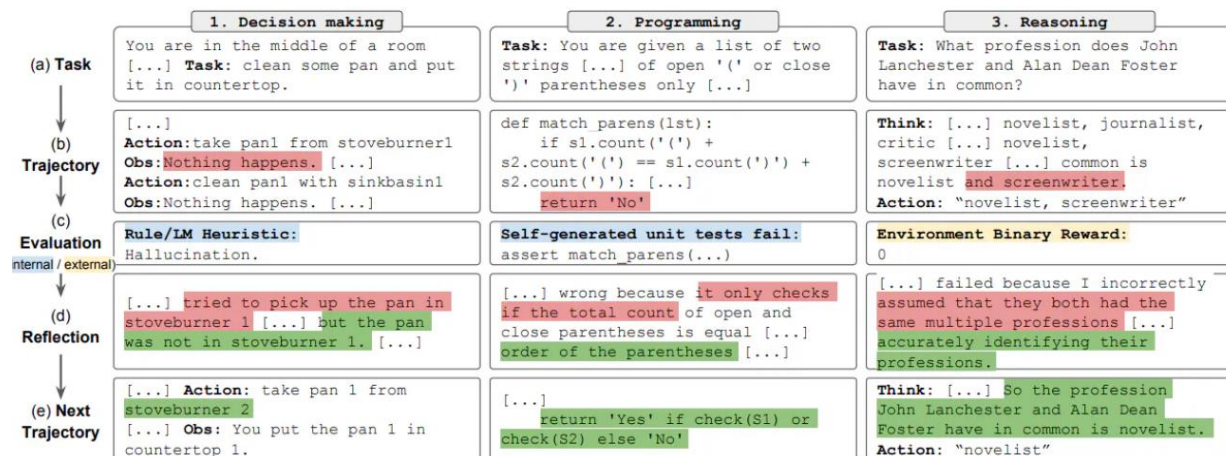
- Hiệu suất và mạnh mẽ: ReAct vượt trội dưới áp lực và học hỏi nhanh chóng từ các ví dụ tối thiểu. Nó luôn hoạt động tốt hơn các cách tiếp cận khác, ngay cả khi tinh chỉnh và vẫn mạnh mẽ để nhắc nhở các biến thể.
- Con người liên kết và có thể kiểm soát: ReAct cung cấp sự minh bạch trong việc ra quyết định và lý luận, cho phép con người kiểm tra và hướng dẫn quá trình. Hãy nghĩ về nó như thuyền trưởng của một con tàu AI, dễ dàng sửa chữa và hướng dẫn bởi các nhà điều hành con người.

Link documents:

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

<https://blog.gopenai.com/mastering-react-prompting-a-crucial-step-in-langchain-implementation-a-guided-example-for-agents-efdf1b756105>

3.16. Reflexion



Reflexion là kỹ thuật yêu cầu mô hình tự phản ánh và điều chỉnh câu trả lời dựa trên phản hồi trước đó. Điều này giúp cải thiện độ chính xác và chất lượng câu trả lời thông qua quá trình học từ sai lầm. Kỹ thuật này rất hữu ích trong các ứng dụng yêu cầu độ chính xác cao và liên tục cải thiện.

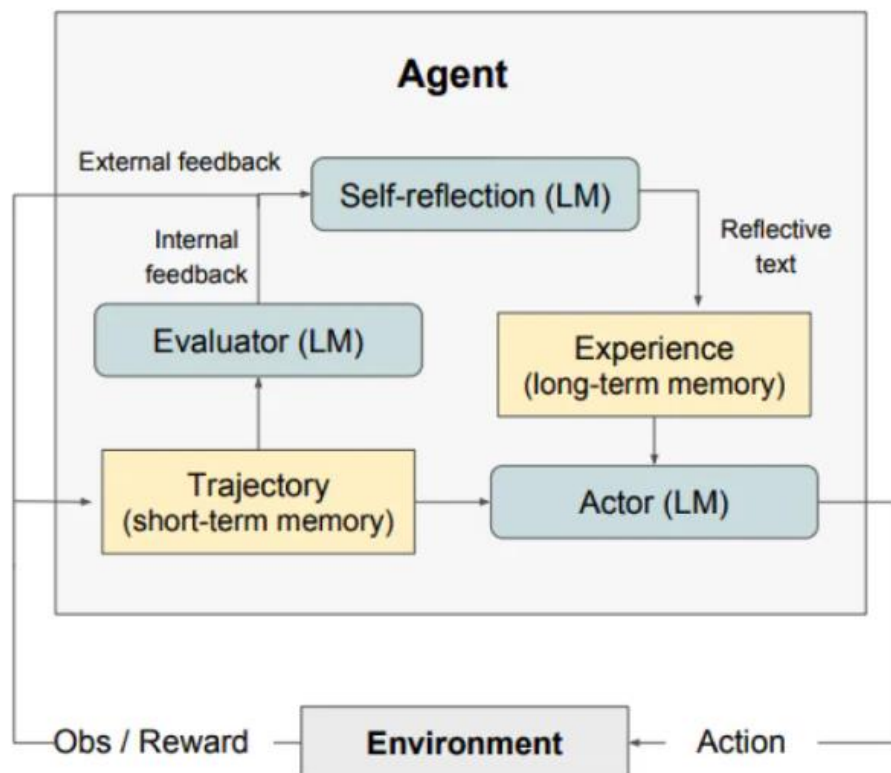
Reflexion là một phương pháp cho phép Mô hình ngôn ngữ (LM), chẳng hạn như GPT (Generative Pretrainer Transformer), sử dụng các vòng phản hồi để tự cải thiện. Khung này dựa

trên tiền đề rằng AI có thể đóng vai trò vừa là người tạo vừa là người đánh giá kết quả đầu ra của nó. Sau khi thực hiện một nhiệm vụ, AI sử dụng phản hồi tự tạo để tinh chỉnh phản hồi ban đầu, từ đó nâng cao quá trình học tập của nó.

Các thành phần cốt lõi của Reflexion

Reflexion bao gồm ba thành phần chính:

- Actor Model: Đây là trình tạo đầu ra ban đầu. Nó thực hiện các nhiệm vụ dựa trên dữ liệu đầu vào và tạo ra phản hồi ban đầu.
- Mô hình đánh giá: Thành phần này đánh giá đầu ra của Mô hình diễn viên. Nó có thể bao gồm các bài kiểm tra đơn vị cho mã hoặc kiểm tra chất lượng cho văn bản.
- Mô hình tự phản ánh: Sau khi đánh giá, mô hình này sử dụng phản hồi để đưa ra đánh giá về đầu ra, dùng để tinh chỉnh phản hồi trong lần lặp tiếp theo.



Ví dụ:


```

# Pseudocode for implementing Reflexion with LLMs

# Initialize the LLM (e.g., ChatGPT or another model)
initialize_llm(api_key)

# Define the task for the LLM
task_description = define_task()

# Define the Actor model which performs the task
def actor_model(task):
    # Generate initial output for the task
    output = llm_generate_output(task)
    return output

# Define the Evaluator model which critiques the output
def evaluator_model(output):
    # Evaluate the output's quality, efficiency, or correctness
    evaluation = evaluate_output(output)
    return evaluation

# Define the Self-Reflection model which generates feedback based on the
# evaluation
def self_reflection_model(evaluation):
    # Reflect on the evaluation and suggest improvements
    feedback = llm_generate_feedback(evaluation)
    return feedback

# Define the update function to improve the Actor model's task execution
def update_actor_model(task, feedback):
    # Update the task execution based on feedback
    new_task = apply_feedback_to_task(task, feedback)
    return new_task

# The Reflexion loop
def reflexion_loop(task):
    for iteration in range(max_iterations):
        # Actor model performs the task
        output = actor_model(task)

        # Evaluator model critiques the output
        evaluation = evaluator_model(output)

        # If the evaluation is satisfactory, exit the loop
        if check_satisfaction(evaluation):
            break

        # Self-Reflection model generates feedback
        feedback = self_reflection_model(evaluation)

        # Update the task based on feedback
        task = update_actor_model(task, feedback)

    return task, output, evaluation

# Run the Reflexion loop
final_task, final_output, final_evaluation = reflexion_loop(task_description)

```

```
# Output the results
print("Final Task:", final_task)
print("Final Output:", final_output)
print("Final Evaluation:", final_evaluation)
```

[Link documents:](#)

<https://levelup.gitconnected.com/what-is-the-reflexion-framework-and-how-to-use-it-in-chatgpt-and-other-llm-c16ca9fe3d57>

<https://www.promptingguide.ai/techniques/reflexion>

3.17. Multimodal CoT Prompting

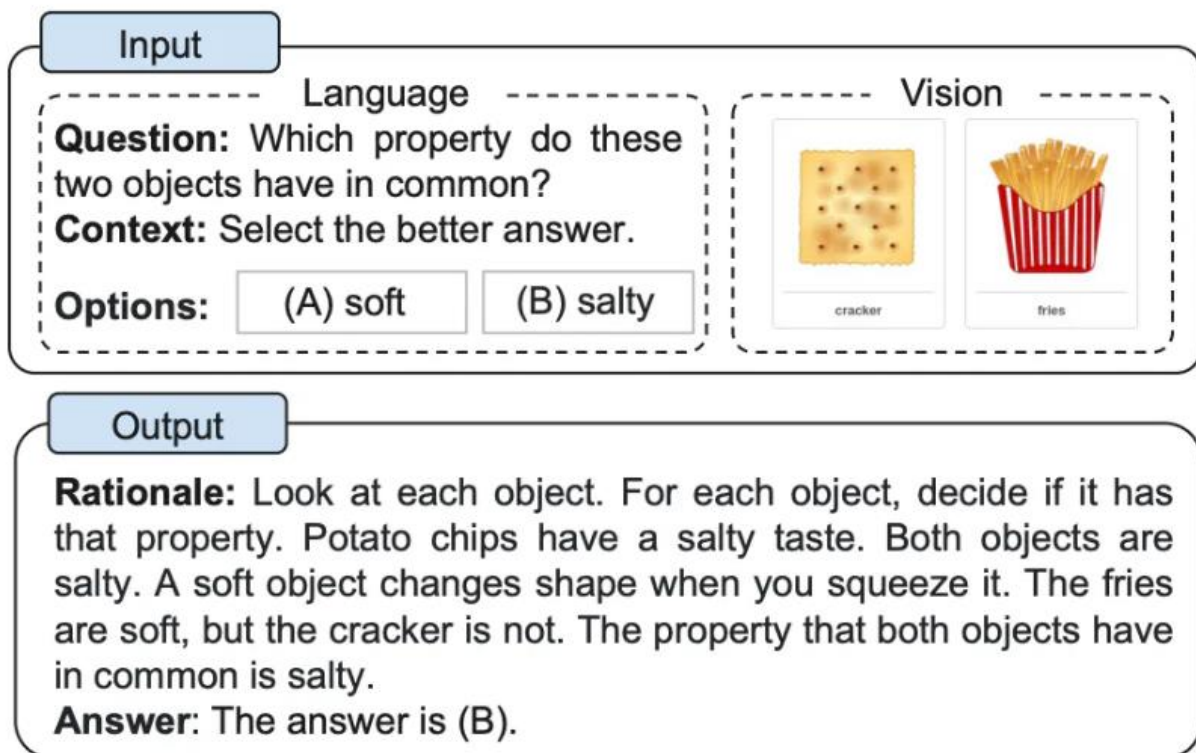


Figure 1. Example of the multimodal CoT task.

Multimodal CoT (Chain-of-Thought) Prompting kết hợp nhiều loại dữ liệu (văn bản, hình ảnh, âm thanh) trong quá trình suy nghĩ chuỗi. Đây là kỹ thuật quan trọng khi làm việc với các ứng dụng đa phương tiện và cần sự tích hợp giữa các dạng dữ liệu khác nhau.

Link documents:

<https://arxiv.org/pdf/2302.14045>

3.18. GraphPrompts

GraphPrompts sử dụng đồ thị để biểu diễn và điều hướng thông tin trong prompt, giúp mô hình hiểu và xử lý các mối quan hệ phức tạp. Kỹ thuật này rất quan trọng trong các ngữ cảnh yêu cầu xử lý dữ liệu có cấu trúc phức tạp, như trong các bài toán về mạng xã hội hoặc dữ liệu liên kết.

Link documents:

<https://arxiv.org/pdf/2302.08043>

4. Tổng kết

4.1. Zero-shot

Mô tả: Mô hình giải quyết các nhiệm vụ mới mà không cần ví dụ mẫu.

Ưu tiên: Cao

Trường hợp sử dụng: Hữu ích trong các tình huống cần phản hồi ngay lập tức mà không có dữ liệu huấn luyện cụ thể.

Ví dụ cụ thể:

- Chức năng: Trả lời các câu hỏi kiến thức chung.
- Ứng dụng: Chatbot trong một ứng dụng giáo dục có thể trả lời các câu hỏi về lịch sử, khoa học mà không cần được huấn luyện trước đó với các ví dụ cụ thể.

4.2. One-shot

- Mô tả: Mô hình cần một ví dụ mẫu để thực hiện nhiệm vụ.
- Ưu tiên: Trung bình

- Trường hợp sử dụng: Khi có ít dữ liệu huấn luyện.
- Ví dụ cụ thể:
 - Chức năng: Học cách trả lời các dạng câu hỏi cụ thể.
 - Ứng dụng: Chatbot học cách trả lời câu hỏi về chính sách bảo hành của sản phẩm từ một ví dụ duy nhất.

4.3. Few-shot

- Mô tả: Mô hình sử dụng vài ví dụ mẫu để giải quyết nhiệm vụ.
- Ưu tiên: Cao
- Trường hợp sử dụng: Khi có một số lượng nhỏ dữ liệu huấn luyện.
- Ví dụ cụ thể:
 - Chức năng: Trả lời câu hỏi về sản phẩm.
 - Ứng dụng: Chatbot của một công ty bán lẻ học cách trả lời các câu hỏi về giá cả, tính năng và chính sách bảo hành từ vài ví dụ cụ thể.

4.4. Chain-of-Thought (CoT)

- Mô tả: Mô hình giải thích quá trình suy luận để đạt đến kết quả.
- Ưu tiên: Cao
- Trường hợp sử dụng: Cần giải thích hoặc quá trình suy luận rõ ràng.
- Ví dụ cụ thể:
 - Chức năng: Giải bài tập toán.
 - Ứng dụng: Chatbot trong một ứng dụng học tập giải thích từng bước cách giải một bài toán phức tạp.

4.4.1. CoT Prompting

- Mô tả: Kết hợp các gợi ý trong quá trình suy luận chuỗi.
- Ưu tiên: Cao
- Trường hợp sử dụng: Nâng cao khả năng suy luận của chatbot.
- Ví dụ cụ thể:
 - Chức năng: Hướng dẫn nấu ăn.
 - Ứng dụng: Chatbot trong ứng dụng nấu ăn hướng dẫn từng bước chi tiết từ chuẩn bị nguyên liệu đến chế biến.

4.4.2. Zero-shot CoT Prompting

- Mô tả: Kết hợp phương pháp zero-shot với suy luận chuỗi.

- Ưu tiên: Trung bình
- Trường hợp sử dụng: Khi cần suy luận mà không có dữ liệu mẫu.
- Ví dụ cụ thể:
 - Chức năng: Giải thích hiện tượng khoa học.
 - Ứng dụng: Chatbot trong một ứng dụng khoa học giải thích các hiện tượng tự nhiên mà không cần ví dụ cụ thể.

4.4.3. Automatic Chain-of-Thought (Auto-CoT)

- Mô tả: Mô hình tự động tạo ra các chuỗi suy nghĩ.
- Ưu tiên: Cao
- Trường hợp sử dụng: Khi cần tự động hóa quá trình suy luận.
- Ví dụ cụ thể:
 - Chức năng: Phân tích xu hướng thị trường.
 - Ứng dụng: Chatbot trong lĩnh vực tài chính tự động phân tích và dự đoán xu hướng thị trường dựa trên các dữ liệu hiện có.

4.5. Self-Consistency

- Mô tả: Tăng tính nhất quán trong các câu trả lời của mô hình.
- Ưu tiên: Cao
- Trường hợp sử dụng: Cần đảm bảo các phản hồi của chatbot nhất quán.
- Ví dụ cụ thể:
 - Chức năng: Chẩn đoán y tế.
 - Ứng dụng: Chatbot tư vấn y tế đảm bảo các chẩn đoán và khuyến nghị nhất quán qua các cuộc hội thoại.

4.6. Generate Knowledge

- Mô tả: Tạo ra thông tin mới từ dữ liệu hiện có.
- Ưu tiên: Cao
- Trường hợp sử dụng: Khi cần mở rộng kiến thức của mô hình.
- Ví dụ cụ thể:
 - Chức năng: Cập nhật thông tin mới.
 - Ứng dụng: Chatbot tự động cập nhật và thông báo tin tức mới nhất từ các nguồn tin tức đáng tin cậy.

4.7. Prompt Chaining

- Mô tả: Kết hợp nhiều prompt để dẫn dắt mô hình.
- Ưu tiên: Cao
- Trường hợp sử dụng: Khi cần xây dựng các cuộc hội thoại phức tạp.
- Ví dụ cụ thể:
 - Chức năng: Phỏng vấn người dùng.
 - Ứng dụng: Chatbot trong ứng dụng tuyển dụng thực hiện các cuộc phỏng vấn với nhiều gợi ý liên tiếp để thu thập thông tin từ ứng viên.

4.8. Tree of Thoughts (ToT)

- Mô tả: Tạo ra các cây suy nghĩ để khám phá nhiều con đường suy luận.
- Ưu tiên: Trung bình
- Trường hợp sử dụng: Khi cần nhiều phương án suy luận.
- Ví dụ cụ thể:
 - Chức năng: Lập kế hoạch.
 - Ứng dụng: Chatbot lập kế hoạch cho một sự kiện, xem xét nhiều kịch bản và lựa chọn phương án tối ưu.

4.9. Retrieval Augmented Generation (RAG)

- Mô tả: Kết hợp truy xuất thông tin với quá trình tạo phản hồi.
- Ưu tiên: Cao
- Trường hợp sử dụng: Khi cần tìm kiếm thông tin từ cơ sở dữ liệu.
- Ví dụ cụ thể:
 - Chức năng: Trả lời câu hỏi chuyên sâu.
 - Ứng dụng: Chatbot hỗ trợ khách hàng truy xuất và cung cấp thông tin từ tài liệu hỗ trợ hoặc cơ sở dữ liệu nội bộ.

4.10. Automatic Reasoning and Tool-use (ART)

- Mô tả: Mô hình tự động suy luận và sử dụng các công cụ hỗ trợ.
- Ưu tiên: Cao
- Trường hợp sử dụng: Khi cần tích hợp khả năng suy luận và sử dụng công cụ.
- Ví dụ cụ thể:
 - Chức năng: Hỗ trợ kỹ thuật.

- Ứng dụng: Chatbot chẩn đoán sự cố kỹ thuật và sử dụng các công cụ để đưa ra giải pháp khắc phục.

4.11. Automatic Prompt Engineer (APE)

- Mô tả: Tự động tạo và điều chỉnh các prompt.
- Ưu tiên: Trung bình
- Trường hợp sử dụng: Khi cần tối ưu hóa prompt tự động.
- Ví dụ cụ thể:
 - Chức năng: Tối ưu hóa câu hỏi.
 - Ứng dụng: Chatbot trong ứng dụng học tập tự động điều chỉnh cách đặt câu hỏi để tối ưu hóa quá trình học tập của học sinh.

4.12. Active-Prompt

- Mô tả: Tạo ra các prompt chủ động để kích thích phản hồi cụ thể.
- Ưu tiên: Trung bình
- Trường hợp sử dụng: Khi cần phản hồi chính xác theo một định hướng cụ thể.
- Ví dụ cụ thể:
 - Chức năng: Hướng dẫn quy trình.
 - Ứng dụng: Chatbot hướng dẫn người dùng từng bước trong quá trình đăng ký tài khoản hoặc hoàn thành một quy trình phức tạp.

4.13. Directional Stimulus Prompting

- Mô tả: Hướng dẫn mô hình theo một hướng cụ thể bằng gợi ý.
- Ưu tiên: Trung bình
- Trường hợp sử dụng: Khi cần định hướng phản hồi của mô hình.
- Ví dụ cụ thể:
 - Chức năng: Hướng dẫn xử lý vấn đề.
 - Ứng dụng: Chatbot hướng dẫn người dùng về quy trình trả lại sản phẩm, với các bước cụ thể và chi tiết.

4.14. PAL (Program-Aided Language Models)

- Mô tả: Kết hợp mô hình ngôn ngữ với chương trình.
- Ưu tiên: Cao

- Trường hợp sử dụng: Khi cần các kết quả suy luận và tính toán phức tạp.
- Ví dụ cụ thể:
 - Chức năng: Phân tích tài chính.
 - Ứng dụng: Chatbot phân tích dữ liệu tài chính và đưa ra dự đoán hoặc khuyến nghị đầu tư.

4.15. ReAct Prompting

- Mô tả: Kết hợp phản ứng nhanh và hành động có kế hoạch.
- Ưu tiên: Cao
- Trường hợp sử dụng: Khi cần phản hồi tự nhiên và liên tục.
- Ví dụ cụ thể:
 - Chức năng: Hỗ trợ khách hàng.
 - Ứng dụng: Chatbot xử lý yêu cầu và vấn đề của khách hàng trong thời gian thực, với khả năng phản ứng linh hoạt.

4.16. Reflexion

- Mô tả: Mô hình tự phản ánh và điều chỉnh dựa trên phản hồi trước đó.
- Ưu tiên: Cao
- Trường hợp sử dụng: Khi cần cải thiện chất lượng phản hồi qua thời gian.
- Ví dụ cụ thể:
 - Chức năng: Học hỏi từ tương tác.
 - Ứng dụng: Chatbot học hỏi từ các cuộc hội thoại trước để cải thiện dịch vụ khách hàng, chẳng hạn như điều chỉnh cách trả lời để phù hợp hơn với nhu cầu của người dùng.

4.17. Multimodal CoT Prompting

- Mô tả: Kết hợp nhiều loại dữ liệu trong quá trình suy nghĩ chuỗi.
- Ưu tiên: Cao
- Trường hợp sử dụng: Khi làm việc với các ứng dụng đa phương tiện.
- Ví dụ cụ thể:
 - Chức năng: Hỗ trợ khách hàng đa phương tiện.
 - Ứng dụng: Chatbot sử dụng cả văn bản và hình ảnh để hướng dẫn khách hàng cách lắp ráp một sản phẩm hoặc xử lý một vấn đề kỹ thuật.

4.18. GraphPrompts

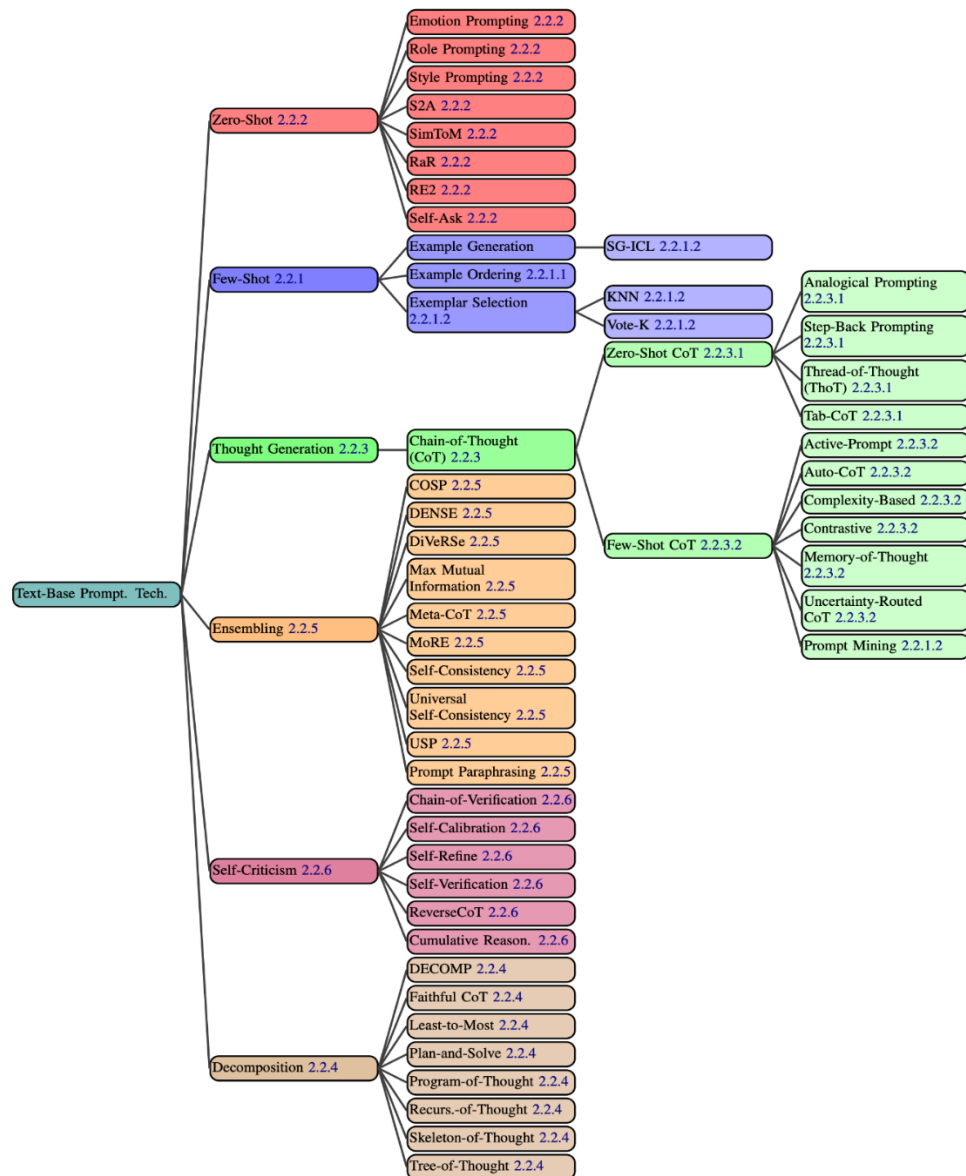
- Mô tả: Sử dụng đồ thị để điều hướng thông tin trong prompt.
- Ưu tiên: Trung bình
- Trường hợp sử dụng: Khi xử lý dữ liệu có cấu trúc phức tạp.
- Ví dụ cụ thể:
 - Chức năng: Phân tích mạng xã hội.
 - Ứng dụng: Chatbot phân tích và trả lời các câu hỏi liên quan đến mạng lưới quan hệ trên các nền tảng mạng xã hội, dựa trên cấu trúc đồ thị.

Tập trung vào các kỹ thuật nào?

Few-shot, CoT Prompting, RAG, ReAct Prompting, Reflexion, và Multimodal CoT Prompting là những kỹ thuật quan trọng và cần được tập trung khi phát triển chatbot và agent. Những kỹ thuật này giúp chatbot trả lời chính xác, cung cấp các phản hồi chi tiết, truy xuất thông tin từ cơ sở dữ liệu, phản ứng linh hoạt và học hỏi từ các tương tác trước.

5. Text-Based Techniques (Những kỹ thuật dựa trên văn bản)

Link documents: https://trigaten.github.io/Prompt_Survey_Site/



5.1. In-Context Learning (ICL)

5.1.1. Few-Shot Prompting

Mô Tả: Kỹ thuật Few-Shot Prompting sử dụng một vài ví dụ (exemplars) để dạy mô hình cách hoàn thành nhiệm vụ mà không cần cập nhật trọng số của nó. Ví dụ: để mô hình học cách phân loại cảm xúc, bạn có thể cung cấp một vài ví dụ về câu văn kèm theo nhãn cảm xúc tương ứng.

Cách Sử Dụng: Mô hình được cung cấp các ví dụ này trong prompt, và từ đó nó học cách áp dụng các mẫu này cho các nhiệm vụ mới mà không cần huấn luyện lại.

5.1.2. Example Generation

Mô Tả: Việc tự động tạo ra các ví dụ bằng cách sử dụng mô hình AI. Đây là một kỹ thuật hữu ích khi không có sẵn dữ liệu huấn luyện.

Cách Sử Dụng: Mô hình sẽ tạo ra các ví dụ dựa trên ngữ cảnh hiện tại và sử dụng chúng để giải quyết nhiệm vụ mới.

5.1.3. Example Ordering

Mô Tả: Thứ tự của các ví dụ được trình bày trong prompt có thể ảnh hưởng đến hiệu suất của mô hình.

Cách Sử Dụng: Thử nghiệm với các thứ tự khác nhau để tìm ra thứ tự tối ưu cho nhiệm vụ cụ thể của bạn.

5.1.4. Exemplar Selection

Mô Tả: Chọn các ví dụ tốt nhất để đưa vào prompt. Việc này có thể dựa trên các tiêu chí như sự tương tự ngữ nghĩa với nhiệm vụ hiện tại.

Cách Sử Dụng: Sử dụng các thuật toán như K-Nearest Neighbor (KNN) để chọn các ví dụ tương tự nhất từ dữ liệu huấn luyện.

5.1.5. Exemplar Label Distribution

Mô Tả: Phân phối nhãn của các ví dụ trong prompt. Một phân phối nhãn cân bằng có thể giúp mô hình tránh bị thiên vị.

Cách Sử Dụng: Đảm bảo rằng các nhãn trong các ví dụ được phân phối đồng đều để cải thiện độ chính xác và tính nhất quán của mô hình.

5.1.6. Exemplar Label Quality

Mô Tả: Độ chính xác của nhãn trong các ví dụ. Các nhãn chính xác thường giúp mô hình học tốt hơn, mặc dù một số nghiên cứu cho rằng mô hình vẫn có thể hoạt động tốt ngay cả khi có nhãn sai.

Cách Sử Dụng: Sử dụng các ví dụ có nhãn chính xác để cải thiện hiệu suất của mô hình.

5.1.7. Exemplar Format

Mô Tả: Định dạng của các ví dụ (ví dụ: Q: {input}, A: {label}). Các định dạng phổ biến trong dữ liệu huấn luyện thường hoạt động tốt hơn.

Cách Sử Dụng: Thử nghiệm với các định dạng khác nhau để xác định định dạng nào mang lại kết quả tốt nhất cho nhiệm vụ của bạn.

5.1.8. Exemplar Similarity

Mô Tả: Chọn các ví dụ tương tự với mẫu thử nghiệm. Các ví dụ tương tự thường giúp mô hình hoạt động tốt hơn.

Cách Sử Dụng: Chọn các ví dụ có nội dung và ngữ cảnh gần gũi với mẫu thử nghiệm để cải thiện hiệu suất của mô hình.

5.1.9. K-Nearest Neighbor (KNN)

Mô Tả: KNN sử dụng để chọn các ví dụ tương tự nhất với mẫu thử nghiệm từ dữ liệu huấn luyện.

Sử Dụng Tối Ưu: Kỹ thuật này giúp mô hình áp dụng các mẫu đã học từ các ví dụ tương tự, cải thiện hiệu suất.

5.1.10. Vote-K

Mô Tả: Vote-K là phương pháp chọn các ví dụ ứng viên có ích nhất để đưa vào prompt sau khi được gán nhãn bởi người gán nhãn.

Sử Dụng Tối Ưu: Kỹ thuật này giúp tăng tính đa dạng và cải thiện hiệu suất mô hình.

5.1.11. Self-Generated In-Context Learning (SG-ICL)

Mô Tả: SG-ICL sử dụng mô hình AI để tự động tạo ra các ví dụ khi không có dữ liệu huấn luyện sẵn có.

Sử Dụng Tối Ưu: Kỹ thuật này hữu ích trong các tình huống thiếu dữ liệu huấn luyện.

5.1.12. Prompt Mining

Mô Tả: Prompt Mining tìm ra các từ hoặc cụm từ tối ưu trong prompt thông qua phân tích một tập hợp dữ liệu lớn.

Sử Dụng Tối Ưu: Sử dụng các định dạng phổ biến từ dữ liệu huấn luyện để cải thiện hiệu suất.

5.2. Zero-Shot

5.2.1. Role Prompting

Mô Tả: Role Prompting, còn được gọi là persona prompting, là kỹ thuật yêu cầu mô hình đóng vai một người cụ thể, như "Madonna" hoặc "một nhà văn" hoặc "trợ lý". Điều này giúp tạo ra các phản hồi mang phong cách cụ thể và có thể cải thiện độ chính xác trong một số nhiệm vụ.

Sử Dụng Tối Ưu: Sử dụng khi cần các phản hồi có phong cách hoặc giọng điệu nhất định.

5.2.2. Style Prompting

Mô Tả: Style Prompting là kỹ thuật yêu cầu mô hình tạo ra phản hồi theo một phong cách, tông giọng, hoặc thể loại cụ thể. Ví dụ: "Viết một đoạn văn ngắn và rõ ràng về lạc đà."

Sử Dụng Tối Ưu: Sử dụng khi muốn điều chỉnh phong cách của phản hồi để phù hợp với nhu cầu cụ thể.

5.2.3. Emotion Prompting

Mô Tả: Emotion Prompting là kỹ thuật sử dụng các cụm từ liên quan đến cảm xúc con người (ví dụ: "Điều này rất quan trọng đối với sự nghiệp của tôi") để cải thiện hiệu suất của mô hình trong các tác vụ benchmark và tạo văn bản mở.

Sử Dụng Tối Ưu: Sử dụng khi muốn mô hình phản ánh các cảm xúc hoặc tông giọng cụ thể.

5.2.4. System 2 Attention (S2A)

Mô Tả: S2A đầu tiên yêu cầu mô hình viết lại prompt và loại bỏ bất kỳ thông tin không liên quan nào. Sau đó, prompt mới này được đưa vào mô hình để lấy phản hồi cuối cùng.

Sử Dụng Tối Ưu: Sử dụng khi muốn loại bỏ thông tin không cần thiết và tập trung vào câu hỏi chính.

5.2.5. SimToM

Mô Tả: SimToM giải quyết các câu hỏi phức tạp liên quan đến nhiều người hoặc đối tượng bằng cách xác định bộ thông tin mà một người biết và sau đó trả lời câu hỏi dựa trên bộ thông tin đó.

Sử Dụng Tối Ưu: Sử dụng khi câu hỏi liên quan đến nhiều đối tượng và cần loại bỏ thông tin không liên quan.

5.2.6. Rephrase and Respond (RaR)

Mô Tả: RaR yêu cầu mô hình viết lại và mở rộng câu hỏi trước khi đưa ra câu trả lời cuối cùng. Kỹ thuật này có thể được thực hiện trong một lượt hoặc chia ra làm hai lượt.

Sử Dụng Tối Ưu: Sử dụng khi câu hỏi ban đầu cần được làm rõ hoặc mở rộng để mô hình hiểu rõ hơn.

5.2.7. Re-reading (RE2)

Mô Tả: RE2 thêm cụm từ "Đọc lại câu hỏi:" vào prompt và lặp lại câu hỏi. Mặc dù đơn giản, kỹ thuật này đã cải thiện hiệu suất trong các benchmark về lý luận, đặc biệt là với các câu hỏi phức tạp.

Sử Dụng Tối Ưu: Sử dụng khi muốn mô hình tập trung lại vào câu hỏi chính để cải thiện độ chính xác.

5.2.8. Self-Ask

Mô Tả: Self-Ask yêu cầu mô hình trước tiên quyết định liệu nó có cần đặt thêm câu hỏi phụ cho một prompt nhất định hay không. Nếu cần, mô hình sẽ tạo ra các câu hỏi này, sau đó trả lời chúng và cuối cùng trả lời câu hỏi gốc.

Sử Dụng Tối Ưu: Sử dụng khi câu hỏi ban đầu có thể yêu cầu thông tin bổ sung để trả lời chính xác.

Tổng Quan

Các kỹ thuật Zero-Shot Prompting rất hữu ích khi không có ví dụ (exemplars) nào được cung cấp trong prompt. Chúng tận dụng khả năng của mô hình để hiểu và trả lời các câu hỏi dựa trên kiến thức đã được học từ dữ liệu huấn luyện mà không cần thêm ví dụ cụ thể nào. Các kỹ thuật này giúp tối ưu hóa hiệu suất của mô hình trong các tình huống khác nhau, từ việc tạo văn bản theo phong cách hoặc cảm xúc cụ thể đến việc xử lý các câu hỏi phức tạp và yêu cầu bổ sung thông tin.

5.3. Thought Generation (Thế hệ suy nghĩ)

5.3.1. Chain-of-Thought (CoT) Prompting

Mô Tả: Chain-of-Thought (CoT) Prompting là kỹ thuật khuyến khích mô hình trình bày quá trình suy nghĩ trước khi đưa ra câu trả lời cuối cùng. Kỹ thuật này thường sử dụng các ví dụ (exemplars) có chứa câu hỏi, quy trình suy luận và câu trả lời đúng.

Cách Sử Dụng: Kỹ thuật này đặc biệt hữu ích cho các nhiệm vụ yêu cầu suy luận phức tạp, như toán học và giải quyết vấn đề.

5.3.2. Zero-Shot CoT

Mô Tả: Zero-Shot CoT là phiên bản đơn giản nhất của CoT, không yêu cầu ví dụ. Kỹ thuật này chỉ cần thêm một cụm từ khuyến khích suy nghĩ như "Hãy suy nghĩ từng bước một" vào prompt.

Cách Sử Dụng: Sử dụng khi không có sẵn các ví dụ, để mô hình tự suy luận và đưa ra câu trả lời.

5.3.3. Step-Back Prompting

Mô Tả: Step-Back Prompting là kỹ thuật yêu cầu mô hình trước tiên trả lời một câu hỏi chung chung về các khái niệm hoặc thông tin liên quan, sau đó mới bắt đầu quá trình suy luận.

Cách Sử Dụng: Sử dụng để cải thiện hiệu suất trong các bài toán suy luận phức tạp.

5.3.4. Analogical Prompting

Mô Tả: Analogical Prompting là kỹ thuật tạo ra các ví dụ tương tự tự động, bao gồm cả quy trình suy luận. Kỹ thuật này đã chứng minh sự cải thiện trong các nhiệm vụ về toán học và lập trình.

Cách Sử Dụng: Sử dụng để mô hình học từ các ví dụ tương tự và áp dụng quy trình suy luận vào nhiệm vụ hiện tại.

5.3.5. Thread-of-Thought (ThoT) Prompting

Mô Tả: Thread-of-Thought (ThoT) Prompting sử dụng một cụm từ khuyến khích suy nghĩ cải tiến cho CoT, Thay vì "Hãy suy nghĩ từng bước", nó sử dụng "Hãy hướng dẫn tôi qua bối cảnh này theo từng phần có thể quản lý được, tóm tắt và phân tích khi chúng tôi thực hiện".

Cách Sử Dụng: Sử dụng trong các nhiệm vụ hỏi-đáp và truy xuất thông tin, đặc biệt khi xử lý các ngữ cảnh phức tạp và lớn.

5.3.6. Tabular Chain-of-Thought (Tab-CoT)

Mô Tả: Tabular Chain-of-Thought (Tab-CoT) yêu cầu mô hình xuất ra quá trình suy luận dưới dạng bảng markdown. Thiết kế bảng này giúp cải thiện cấu trúc và quá trình suy luận của mô hình.

Cách Sử Dụng: Sử dụng khi cần một cấu trúc rõ ràng và có tổ chức cho quá trình suy luận.

5.3.7. Few-Shot CoT

Mô Tả: Few-Shot CoT sử dụng nhiều ví dụ chứa các quy trình suy luận để giúp mô hình cải thiện hiệu suất. Kỹ thuật này thường được gọi là Manual-CoT hoặc Golden CoT.

Cách Sử Dụng: Sử dụng nhiều ví dụ có chất lượng cao để mô hình học cách suy luận từ các ví dụ đó.

5.3.8. Contrastive CoT Prompting

Mô Tả: Contrastive CoT Prompting thêm cả các ví dụ với giải thích sai và đúng vào prompt để cho mô hình thấy cách không nên suy luận. Phương pháp này đã cải thiện hiệu suất trong các nhiệm vụ về suy luận số học và hỏi-đáp thực tế.

Cách Sử Dụng: Sử dụng để dạy mô hình tránh các lỗi phổ biến trong quá trình suy luận.

5.3.9. Uncertainty-Routed CoT Prompting

Mô Tả: Uncertainty-Routed CoT Prompting lấy mẫu nhiều quy trình suy luận, sau đó chọn đáp án đúng nếu vượt ngưỡng nhất định. Phương pháp này đã cải thiện hiệu suất trên các benchmark MMLU cho các mô hình như GPT-4.

Cách Sử Dụng: Sử dụng khi cần xác định mức độ tự tin trong các đáp án của mô hình.

5.3.10. Complexity-Based Prompting

Mô Tả: Complexity-Based Prompting chọn các ví dụ phức tạp để mô hình học và sử dụng bỏ phiếu đa số cho các chuỗi suy luận dài hơn. Phương pháp này đã cải thiện hiệu suất trên các bộ dữ liệu về suy luận toán học.

Cách Sử Dụng: Sử dụng khi các nhiệm vụ yêu cầu suy luận phức tạp và chi tiết.

5.3.11. Active Prompting

Mô Tả: Active Prompting bắt đầu với một số câu hỏi huấn luyện, yêu cầu mô hình giải quyết chúng và sau đó tính toán sự không chắc chắn. Các ví dụ có mức độ không chắc chắn cao sẽ được chỉnh sửa bởi người gán nhãn.

Cách Sử Dụng: Sử dụng để cải thiện chất lượng ví dụ bằng cách loại bỏ những ví dụ không rõ ràng.

5.3.12. Memory-of-Thought Prompting

Mô Tả: Memory-of-Thought Prompting sử dụng các ví dụ huấn luyện không gán nhãn để xây dựng các prompt CoT tại thời điểm thử nghiệm. Kỹ thuật này đã cải thiện hiệu suất trên các benchmark về suy luận số học, kiến thức phổ thông và suy luận thực tế.

Cách Sử Dụng: Sử dụng các ví dụ huấn luyện để tạo ra các prompt CoT tương tự với mẫu thử nghiệm.

5.3.13. Automatic Chain-of-Thought (Auto-CoT)

Mô Tả: Auto-CoT sử dụng prompt Zero-Shot CoT để tự động tạo ra các chuỗi suy luận. Các chuỗi này sau đó được sử dụng để xây dựng một prompt Few-Shot CoT cho mẫu thử nghiệm.

Cách Sử Dụng: Sử dụng khi cần tạo ra các chuỗi suy luận tự động để cải thiện hiệu suất.

5.4. Decomposition

5.4.1. Least-to-Most Prompting

Mô Tả: Least-to-Most Prompting là kỹ thuật yêu cầu mô hình chia một vấn đề phức tạp thành các vấn đề con nhỏ hơn mà không giải quyết chúng. Sau đó, mô hình sẽ giải quyết các vấn đề con này theo trình tự và ghép các phản hồi để có được kết quả cuối cùng.

Cách Sử Dụng: Kỹ thuật này hữu ích cho các nhiệm vụ liên quan đến xử lý ký hiệu, tổng quát hóa cấu trúc và suy luận toán học.

5.4.2. Prompting (DECOMP)

Mô Tả: DECOMP là kỹ thuật Few-Shot yêu cầu mô hình học cách sử dụng một số chức năng nhất định. Các chức năng này có thể bao gồm các tác vụ như tách chuỗi hoặc tìm kiếm trên internet, thường được thực hiện dưới dạng các cuộc gọi mô hình riêng biệt.

Cách Sử Dụng: Kỹ thuật này cải thiện hiệu suất bằng cách cho phép mô hình chia nhỏ vấn đề thành các vấn đề con và giải quyết chúng từng bước.

5.4.3. Plan-and-Solve Prompting

Mô Tả: Plan-and-Solve Prompting là kỹ thuật Zero-Shot CoT cải tiến, yêu cầu mô hình trước tiên hiểu vấn đề và lập kế hoạch giải quyết. Sau đó, mô hình thực hiện kế hoạch và giải quyết vấn đề từng bước.

Cách Sử Dụng: Sử dụng để cải thiện quy trình suy luận và đảm bảo rằng mô hình có một kế hoạch rõ ràng trước khi giải quyết vấn đề.

5.4.4. Tree-of-Thought (ToT)

Mô Tả: Tree-of-Thought (ToT) tạo ra một cấu trúc tìm kiếm dạng cây bằng cách bắt đầu với một vấn đề ban đầu, sau đó tạo ra nhiều bước có thể dưới dạng các suy nghĩ. Mô hình đánh giá tiến trình của mỗi bước và quyết định các bước tiếp theo dựa trên các suy nghĩ này.

Cách Sử Dụng: Kỹ thuật này đặc biệt hiệu quả cho các nhiệm vụ yêu cầu tìm kiếm và lập kế hoạch.

5.4.5. Recursion-of-Thought

Mô Tả: Recursion-of-Thought là kỹ thuật tương tự như CoT, nhưng mỗi khi gặp một vấn đề phức tạp trong chuỗi suy luận, mô hình sẽ gửi vấn đề này vào một prompt hoặc cuộc gọi mô hình khác. Sau khi hoàn thành, câu trả lời sẽ được chèn vào prompt ban đầu.

Cách Sử Dụng: Sử dụng để giải quyết các vấn đề phức tạp và giúp mô hình vượt qua giới hạn ngữ cảnh tối đa.

5.4.6. Program-of-Thoughts

Mô Tả: Program-of-Thoughts sử dụng các mô hình LLM như Codex để tạo ra mã lập trình như các bước suy luận. Một trình thông dịch mã sẽ thực thi các bước này để lấy kết quả cuối cùng.

Cách Sử Dụng: Kỹ thuật này rất hiệu quả cho các nhiệm vụ liên quan đến toán học và lập trình, nhưng ít hiệu quả hơn cho các nhiệm vụ suy luận ngữ nghĩa.

5.4.7. Faithful Chain-of-Thought

Mô Tả: Faithful Chain-of-Thought tạo ra một chuỗi suy nghĩ có cả ngôn ngữ tự nhiên và ngôn ngữ ký hiệu (ví dụ: Python). Kỹ thuật này sử dụng nhiều loại ngôn ngữ ký hiệu khác nhau tùy thuộc vào nhiệm vụ.

Cách Sử Dụng: Sử dụng khi cần một quy trình suy luận kết hợp cả ngôn ngữ tự nhiên và ký hiệu.

5.4.8. Skeleton-of-Thought

Mô Tả: Skeleton-of-Thought tập trung vào việc tăng tốc độ trả lời thông qua việc xử lý song song. Mô hình tạo ra một khung sườn của câu trả lời dưới dạng các vấn đề con, sau đó gửi các câu hỏi này vào các cuộc gọi mô hình song song và kết hợp tất cả các phản hồi để có câu trả lời cuối cùng.

Cách Sử Dụng: Sử dụng khi cần tăng tốc độ xử lý và tối ưu hóa quy trình suy luận.

Tổng Quan

Các kỹ thuật Decomposition giúp mô hình AI chia nhỏ các vấn đề phức tạp thành các phần nhỏ hơn, dễ quản lý hơn và sau đó giải quyết từng phần một cách tuần tự. Điều này không chỉ cải thiện hiệu suất mà còn giúp mô hình có thể giải quyết các vấn đề phức tạp một cách có hệ thống và logic.

5.5. Ensembling

5.5.1. Demonstration Ensembling (DENSE)

Mô Tả: Demonstration Ensembling tạo ra nhiều prompt Few-Shot, mỗi prompt chứa một tập con khác nhau của các ví dụ từ tập huấn luyện. Sau đó, nó tổng hợp các đầu ra từ các prompt này để tạo ra câu trả lời cuối cùng.

Cách Sử Dụng: Sử dụng khi cần tổng hợp nhiều phản hồi từ các prompt khác nhau để cải thiện độ chính xác và độ tin cậy.

5.5.2. Mixture of Reasoning Experts (MoRE)

Mô Tả: MoRE tạo ra một tập hợp các chuyên gia suy luận bằng cách sử dụng các prompt chuyên biệt khác nhau cho các loại suy luận khác nhau (ví dụ: prompt bổ sung thông tin cho suy luận

thực tế, Chain-of-Thought cho suy luận đa bước và toán học, và prompt tạo kiến thức cho suy luận thông thường). Đầu ra tốt nhất từ tất cả các chuyên gia được chọn dựa trên điểm số đồng thuận.

Cách Sử Dụng: Sử dụng khi nhiệm vụ yêu cầu nhiều loại suy luận khác nhau và cần chọn ra phản hồi tốt nhất từ các chuyên gia khác nhau.

5.5.3. Max Mutual Information Method

Mô Tả: Phương pháp Max Mutual Information tạo ra nhiều mẫu prompt với các phong cách và ví dụ khác nhau, sau đó chọn mẫu tối ưu là mẫu tối đa hóa thông tin tương hỗ giữa prompt và các đầu ra của mô hình.

Cách Sử Dụng: Sử dụng khi cần tìm prompt tối ưu bằng cách thử nghiệm nhiều mẫu khác nhau và chọn mẫu có hiệu suất tốt nhất.

5.5.4. Self-Consistency

Mô Tả: Self-Consistency dựa trên trực giác rằng nhiều con đường suy luận khác nhau có thể dẫn đến cùng một câu trả lời. Phương pháp này đầu tiên yêu cầu mô hình thực hiện Chain-of-Thought nhiều lần với nhiệt độ khác không để thu được các con đường suy luận đa dạng, sau đó sử dụng bỏ phiếu đa số trên tất cả các đầu ra để chọn ra phản hồi cuối cùng.

Cách Sử Dụng: Sử dụng khi cần cải thiện độ chính xác bằng cách tổng hợp các phản hồi từ nhiều con đường suy luận khác nhau.

5.5.5. Universal Self-Consistency

Mô Tả: Universal Self-Consistency tương tự như Self-Consistency, nhưng thay vì chọn câu trả lời đa số bằng cách đếm chương trình, nó chèn tất cả các đầu ra vào một mẫu prompt để chọn ra câu trả lời đa số. Phương pháp này hữu ích cho việc tạo văn bản tự do và các trường hợp mà cùng một câu trả lời có thể được diễn đạt khác nhau.

Cách Sử Dụng: Sử dụng khi cần tổng hợp các phản hồi từ nhiều phiên bản của một câu trả lời để chọn ra phiên bản tốt nhất.

5.5.6. Meta-Reasoning over Multiple CoTs

Mô Tả: Meta-Reasoning over Multiple CoTs tương tự như Universal Self-Consistency; nó đầu tiên tạo ra nhiều chuỗi suy luận (không nhất thiết là câu trả lời cuối cùng) cho một vấn đề. Sau đó, nó chèn tất cả các chuỗi này vào một mẫu prompt và tạo ra câu trả lời cuối cùng từ chúng.

Cách Sử Dụng: Sử dụng khi cần tổng hợp các chuỗi suy luận để tạo ra câu trả lời cuối cùng.

5.5.7. DiVeRSe

Mô Tả: DiVeRSe tạo ra nhiều prompt cho một vấn đề, sau đó thực hiện Self-Consistency cho từng prompt, tạo ra nhiều con đường suy luận. Chúng đánh giá từng con đường suy luận dựa trên từng bước và chọn ra phản hồi cuối cùng.

Cách Sử Dụng: Sử dụng khi cần đánh giá và tổng hợp các con đường suy luận đa dạng để chọn ra câu trả lời chính xác nhất.

5.5.8. Consistency-based Self-adaptive Prompting (COSP)

Mô Tả: COSP xây dựng các prompt Few-Shot CoT bằng cách chạy Zero-Shot CoT với Self-Consistency trên một tập hợp các ví dụ và chọn ra một tập con đầu ra có độ đồng thuận cao để bao gồm vào prompt cuối cùng. Sau đó, nó thực hiện Self-Consistency với prompt cuối cùng này.

Cách Sử Dụng: Sử dụng khi cần cải thiện độ chính xác bằng cách tạo ra các prompt từ các đầu ra có độ đồng thuận cao.

5.5.9. Universal Self-Adaptive Prompting (USP)

Mô Tả: USP dựa trên thành công của COSP, nhằm làm cho nó trở nên phổ biến với tất cả các nhiệm vụ. USP sử dụng dữ liệu không gán nhãn để tạo ra các ví dụ và một hàm tính điểm phức tạp hơn để chọn ra các ví dụ. Ngoài ra, USP không sử dụng Self-Consistency.

Cách Sử Dụng: Sử dụng khi cần một phương pháp phổ biến và linh hoạt để tạo ra các prompt tối ưu từ dữ liệu không gán nhãn.

5.5.10. Prompt Paraphrasing

Mô Tả: Prompt Paraphrasing chuyển đổi prompt ban đầu bằng cách thay đổi một số từ ngữ mà vẫn giữ nguyên ý nghĩa tổng thể. Đây là một kỹ thuật tăng cường dữ liệu có thể được sử dụng để tạo ra các prompt cho một ensemble.

Cách Sử Dụng: Sử dụng khi cần tạo ra nhiều phiên bản khác nhau của một prompt để tạo ra một ensemble.

Tổng Quan

Các kỹ thuật Ensembling giúp giảm thiểu sự dao động của đầu ra của mô hình và thường cải thiện độ chính xác bằng cách tổng hợp nhiều phản hồi từ các prompt khác nhau. Chúng đặc biệt hữu ích khi cần tăng cường độ tin cậy và độ chính xác của mô hình AI trong các nhiệm vụ phức tạp và đa dạng.

5.6. Self-Criticism (Tự đánh giá)

5.6.1. Self-Criticism

Mô Tả: Self-Calibration là kỹ thuật yêu cầu mô hình tự đánh giá lại câu trả lời của mình để kiểm tra tính chính xác. Ban đầu, mô hình đưa ra một câu trả lời, sau đó tạo một prompt mới bao gồm câu hỏi ban đầu, câu trả lời của mô hình, và một yêu cầu đánh giá câu trả lời đó.

Cách Sử Dụng: Sử dụng khi cần đánh giá độ tin cậy và độ chính xác của câu trả lời, đặc biệt trong các tình huống mà độ chính xác là rất quan trọng.

5.6.2. Self-Refine

Mô Tả: Self-Refine là một khung làm việc lặp lại, trong đó mô hình cung cấp phản hồi về câu trả lời của chính mình và sau đó cải thiện câu trả lời dựa trên phản hồi đó. Quá trình này tiếp tục cho đến khi đạt được điều kiện dừng (ví dụ: đạt số lần lặp tối đa).

Cách Sử Dụng: Sử dụng khi cần cải thiện chất lượng và độ chính xác của câu trả lời qua nhiều lần lặp lại.

5.6.3. Reversing Chain-of-Thought (RCoT)

Mô Tả: RCoT yêu cầu mô hình tái dựng lại vấn đề dựa trên câu trả lời đã tạo ra. Sau đó, mô hình tạo ra các so sánh chi tiết giữa vấn đề gốc và vấn đề tái dựng để kiểm tra tính nhất quán. Những điểm không nhất quán sau đó được chuyển thành phản hồi để mô hình cải thiện câu trả lời ban đầu.

Cách Sử Dụng: Sử dụng khi cần kiểm tra tính nhất quán và logic của các câu trả lời, đặc biệt trong các nhiệm vụ suy luận phức tạp.

5.6.4. Self-Verification

Mô Tả: Self-Verification tạo ra nhiều câu trả lời khả thi bằng Chain-of-Thought (CoT). Sau đó, nó đánh giá từng giải pháp bằng cách che một số phần của câu hỏi ban đầu và yêu cầu mô hình dự đoán chúng dựa trên phần còn lại của câu hỏi và giải pháp đã tạo ra.

Cách Sử Dụng: Sử dụng khi cần xác minh tính chính xác của các giải pháp bằng cách so sánh các phần bị che và phần dự đoán.

5.6.5. Chain-of-Verification (COVE)

Mô Tả: COVE đầu tiên sử dụng mô hình để tạo ra một câu trả lời cho câu hỏi ban đầu. Sau đó, nó tạo ra một danh sách các câu hỏi liên quan để kiểm tra tính chính xác của câu trả lời. Mỗi câu hỏi được trả lời bởi mô hình, sau đó tất cả thông tin này được cung cấp cho mô hình để tạo ra câu trả lời cuối cùng đã được xác minh.

Cách Sử Dụng: Sử dụng khi cần kiểm tra và xác minh tính chính xác của câu trả lời bằng cách đặt các câu hỏi liên quan và tổng hợp các phản hồi.

5.6.6. Cumulative Reasoning

Mô Tả: Cumulative Reasoning đầu tiên tạo ra một số bước khả thi trong việc trả lời câu hỏi. Sau đó, mô hình đánh giá từng bước này và quyết định chấp nhận hoặc từ chối chúng. Cuối cùng, nó kiểm tra xem có đạt được câu trả lời cuối cùng hay không. Nếu không, quá trình này sẽ lặp lại.

Cách Sử Dụng: Sử dụng khi cần kiểm tra và tổng hợp từng bước suy luận để đảm bảo tính chính xác của câu trả lời cuối cùng.

Tổng Quan

Các kỹ thuật Self-Criticism giúp mô hình AI tự đánh giá và cải thiện các câu trả lời của mình. Bằng cách sử dụng các kỹ thuật này, mô hình có thể tự phát hiện và sửa lỗi, cải thiện độ chính xác và độ tin cậy của các phản hồi. Điều này đặc biệt hữu ích trong các nhiệm vụ yêu cầu suy luận phức tạp và độ chính xác cao.

5.7. Prompt Engineering

5.7.1. Meta Prompting

Mô Tả: Meta Prompting là quá trình yêu cầu mô hình AI tạo ra hoặc cải thiện một prompt hoặc mẫu prompt. Điều này có thể bao gồm việc mô hình tự tạo ra các hướng dẫn mới hoặc cải tiến các hướng dẫn hiện có dựa trên các phản hồi trước đó.

Cách Sử Dụng: Sử dụng khi cần tạo ra các prompt chất lượng cao mà không cần sự can thiệp của con người.

5.7.2. AutoPrompt

Mô Tả: AutoPrompt sử dụng một mô hình LLM cố định cùng với một mẫu prompt chứa một số "trigger tokens" mà giá trị của chúng được cập nhật thông qua backpropagation trong quá trình huấn luyện. Đây là một phiên bản của soft-prompting.

Cách Sử Dụng: Sử dụng khi cần tối ưu hóa prompt thông qua các cập nhật gradient để cải thiện hiệu suất trên các nhiệm vụ cụ thể.

5.7.3. Automatic Prompt Engineer (APE)

Mô Tả: APE sử dụng một tập hợp các ví dụ để tạo ra một prompt hướng dẫn zero-shot. Nó tạo ra nhiều prompt có thể, đánh giá chúng, sau đó tạo ra các biến thể của các prompt tốt nhất. Quá trình này lặp lại cho đến khi đạt được các tiêu chí mong muốn.

Cách Sử Dụng: Sử dụng khi cần tạo và tối ưu hóa prompt một cách tự động dựa trên một tập hợp các ví dụ.

5.7.4. Gradientfree Instructional Prompt Search (GrIPS)

Mô Tả: GrIPS tương tự như APE, nhưng sử dụng một tập hợp các thao tác phức tạp hơn bao gồm xóa, thêm, hoán đổi và diễn đạt lại để tạo ra các biến thể của một prompt ban đầu.

Cách Sử Dụng: Sử dụng khi cần tối ưu hóa prompt mà không cần sử dụng các kỹ thuật dựa trên gradient.

5.7.5. Prompt Optimization with Textual Gradients (ProTeGi)

Mô Tả: ProTeGi là một phương pháp tối ưu hóa prompt cải tiến dựa trên các gradient văn bản. Đầu tiên, nó cho một lô các đầu vào qua template, sau đó đưa ra đầu ra, ground truth, và prompt vào một prompt khác để phê bình prompt ban đầu. Nó tạo ra các prompt mới từ các phê bình này và sử dụng một thuật toán bandit để chọn một prompt tối ưu.

Cách Sử Dụng: Sử dụng khi cần cải thiện prompt bằng cách sử dụng phản hồi từ mô hình và một thuật toán tối ưu hóa.

5.7.6. RLPrompt

Mô Tả: RLPrompt sử dụng một mô hình LLM cố định với một mô-đun không cố định được thêm vào. Nó sử dụng mô hình này để tạo ra các template prompt, đánh giá chúng trên một tập dữ liệu, và cập nhật mô-đun không cố định bằng Soft Q-Learning. Phương pháp này thường chọn các đoạn văn bản không có ngữ pháp làm template tối ưu.

Cách Sử Dụng: Sử dụng khi cần tối ưu hóa prompt bằng cách sử dụng học tăng cường và một mô-đun không cố định.

5.7.7. Dialogue-comprised Policy-gradient-based Discrete Prompt Optimization (DP2O)

Mô Tả: DP2O là một kỹ thuật tối ưu hóa prompt phức tạp nhất, bao gồm học tăng cường, một hàm tính điểm prompt tùy chỉnh, và các cuộc hội thoại với mô hình để tạo ra prompt.

Cách Sử Dụng: Sử dụng khi cần một phương pháp tối ưu hóa prompt phức tạp và tùy chỉnh.

Tổng Quan

Các kỹ thuật Prompt Engineering giúp tự động tối ưu hóa và cải thiện các prompt để đạt hiệu suất tốt hơn từ các mô hình AI. Bằng cách sử dụng các kỹ thuật này, bạn có thể tạo ra các prompt hiệu quả hơn, giúp mô hình AI cung cấp phản hồi chính xác và hữu ích hơn.

5.8. Answer Engineering

5.8.1. Answer Shape

Mô Tả: Answer Shape đề cập đến việc định dạng của câu trả lời. Điều này bao gồm việc xác định câu trả lời nên là một token, một chuỗi token, hay một định dạng khác.

Cách Sử Dụng: Chọn định dạng phù hợp với yêu cầu của nhiệm vụ để đảm bảo rằng câu trả lời được trình bày một cách rõ ràng và chính xác.

5.8.2. Answer Space

Mô Tả: Answer Space là không gian giá trị mà câu trả lời có thể chứa. Đây có thể là một tập hợp các lựa chọn cụ thể hoặc một phạm vi giá trị liên tục.

Cách Sử Dụng: Xác định không gian câu trả lời để mô hình có thể chọn câu trả lời chính xác trong phạm vi giá trị cho phép.

5.8.3. Answer Extractor

Mô Tả: Answer Extractor là quy tắc hoặc hàm được sử dụng để trích xuất câu trả lời cuối cùng từ đầu ra của mô hình. Điều này thường liên quan đến việc xử lý đầu ra để lấy được phần thông tin cần thiết.

Cách Sử Dụng: Sử dụng các quy tắc hoặc hàm trích xuất để đảm bảo rằng câu trả lời cuối cùng được lấy ra một cách chính xác và hiệu quả.

5.8.4. Các Phương Pháp Tối Ưu Hóa Answer Engineering

1. Tailored Answer Extraction

Mô Tả: Tailored Answer Extraction sử dụng các kỹ thuật tùy chỉnh để trích xuất câu trả lời từ các đầu ra của mô hình. Điều này có thể bao gồm việc sử dụng các biểu thức chính quy, các hàm xử lý ngôn ngữ tự nhiên, hoặc các mô hình học máy khác để xác định và trích xuất câu trả lời.

Cách Sử Dụng: Áp dụng các kỹ thuật tùy chỉnh phù hợp với đặc điểm của nhiệm vụ và dữ liệu để trích xuất câu trả lời một cách chính xác.

2. Confidence Scoring

Mô Tả: Confidence Scoring là phương pháp đánh giá độ tin cậy của câu trả lời dựa trên các yếu tố như điểm xác suất, độ tương tự ngữ nghĩa, và các chỉ số đánh giá khác.

Cách Sử Dụng: Sử dụng các phương pháp đánh giá độ tin cậy để xác định mức độ tin cậy của các câu trả lời và chọn câu trả lời có độ tin cậy cao nhất.

3. Ensemble Methods

Mô Tả: Ensemble Methods sử dụng nhiều mô hình hoặc nhiều phiên bản của cùng một mô hình để tạo ra câu trả lời cuối cùng. Điều này có thể bao gồm việc kết hợp các câu trả lời từ các mô hình khác nhau hoặc sử dụng các kỹ thuật bỏ phiếu để chọn câu trả lời cuối cùng.

Cách Sử Dụng: Sử dụng các phương pháp ensemble để kết hợp các câu trả lời từ nhiều nguồn khác nhau nhằm cải thiện độ chính xác và độ tin cậy của câu trả lời.

4. Contextual Answer Validation

Mô Tả: Contextual Answer Validation sử dụng ngữ cảnh xung quanh câu trả lời để xác minh tính chính xác của nó. Điều này có thể bao gồm việc kiểm tra câu trả lời trong ngữ cảnh của đoạn văn hoặc câu hỏi ban đầu.

Cách Sử Dụng: Sử dụng ngữ cảnh để xác minh và điều chỉnh câu trả lời, đảm bảo rằng câu trả lời phù hợp và chính xác trong ngữ cảnh của nó.

5. Multi-step Reasoning

Mô Tả: Multi-step Reasoning là phương pháp yêu cầu mô hình thực hiện nhiều bước suy luận để đưa ra câu trả lời cuối cùng. Điều này thường liên quan đến việc chia nhỏ vấn đề thành các bước nhỏ hơn và giải quyết từng bước một.

Cách Sử Dụng: Áp dụng các bước suy luận đa bước để giải quyết các vấn đề phức tạp và đảm bảo rằng câu trả lời cuối cùng là kết quả của quá trình suy luận kỹ lưỡng.

Tổng Quan

Các kỹ thuật Answer Engineering giúp tối ưu hóa và cải thiện chất lượng của các câu trả lời mà mô hình AI cung cấp. Bằng cách sử dụng các kỹ thuật này, bạn có thể đảm bảo rằng các câu trả lời được trích xuất và trình bày một cách chính xác, đáng tin cậy và phù hợp với yêu cầu của nhiệm vụ.