

Foundation of Prompt Engineering

Ngày 18 tháng 2 năm 2024

Phần I: Tổng quan về Prompt Engineering

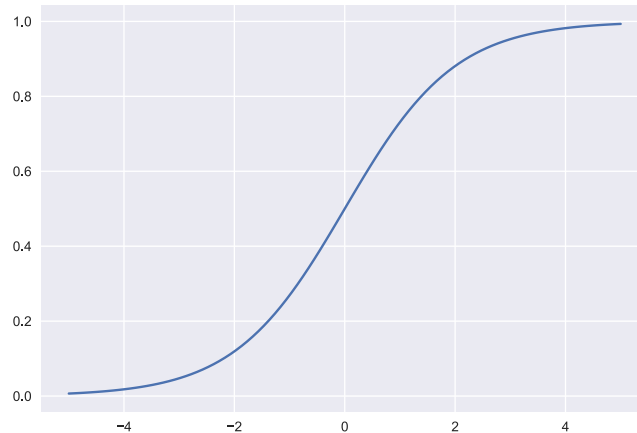
1 LLM Settings

Vài câu giới thiệu tổng quan

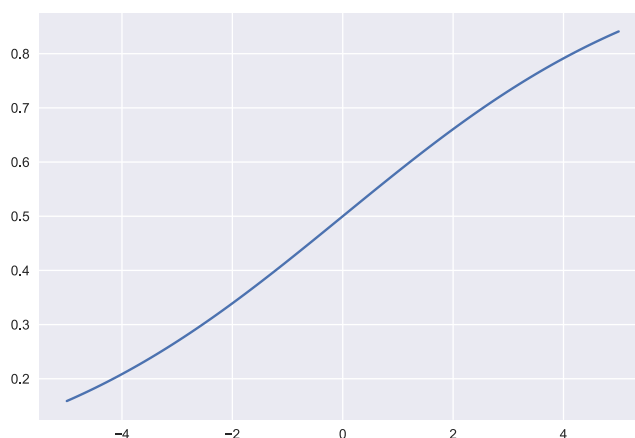
Temperature - Như chúng ta có thể đã biết thì Large Language Model (LLM) hoạt động theo cơ chế classify từ tiếp theo, và như bài toán classification khác thì output đầu ra của LLM sẽ áp dụng thêm một hàm softmax để tạo ra phân phối xác suất từ tiếp theo sao cho tổng xác suất cộng lại bằng 1. Temperature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

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

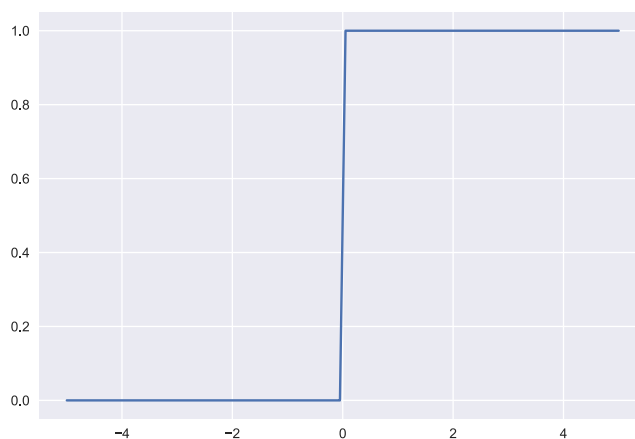
Công thức trên là công thức hàm softmax, trong đó tham số temperature được kí hiệu bằng chữ T , và logit_n kí hiệu output thứ n ở lớp cuối của mô hình LLM. Để cho dễ hình dung thì mình có một visualization nho nhỏ ở bên dưới với hàm sigmoid (hàm softmax cũng có tính chất tương tự).



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



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



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

Top P - là một kỹ thuật sampling với **temperature**, còn được gọi là **nucleus sampling**, giúp chúng ta có thể kiểm soát mức độ định hình của mô hình. Nếu chúng ta đang tìm kiếm câu trả lời chính xác và dựa trên sự thật thì giữ giá trị này ở mức thấp. Nếu chúng ta đang tìm kiếm các phản hồi đa dạng hơn, tăng lên một giá trị cao hơn. Nếu chúng ta sử dụng Top P, nghĩa là chỉ có các token tạo thành khối lượng xác suất top P được xem xét cho kết quả trả về, vì vậy một giá trị top P thấp sẽ cho phép mô hình xem xét nhiều từ có thể xảy ra hơn, bao gồm cả những từ ít có khả năng hơn, dẫn đến các đầu ra đa dạng hơn. Khuyến nghị chung là thay đổi temperature hoặc top P nhưng không phải cả

2.

Max Length - chúng ta có thể quản lý số lượng token mà mô hình tạo ra bằng cách điều chỉnh Max Length. Xác định một độ dài tối đa giúp chúng ta ngăn chặn các kết quả dài hoặc không liên quan và giúp kiểm soát chi phí.

Stop Sequences - là một chuỗi ngăn mô hình tạo ra token. Xác định Stop Sequences là cách khác để kiểm soát độ dài và cấu trúc của kết quả trả về từ mô hình. Ví dụ, chúng ta có thể bảo mô hình tạo ra các danh sách không có quá 10 items bằng cách thêm 11 làm Stop Sequences.

Frequency Penalty - áp dụng một cách phạt cho token tiếp theo tỷ lệ với số lần token đó đã xuất hiện trong kết quả và prompt. Frequency Penalty càng cao, một từ càng ít có khả năng xuất hiện lại. Thiết lập này giảm sự lặp lại của từ trong kết quả trả về từ mô hình bằng cách đưa ra lượng phạt cao hơn cho những token xuất hiện nhiều lần.

Presence Penalty - áp dụng phạt cho các token lặp lại nhưng không giống như frequency penalty, lượng phạt là như nhau cho tất cả các token lặp lại. một token xuất hiện hai lần và một token xuất hiện 10 lần được phạt như nhau. Thiết lập này ngăn mô hình lặp lại các cụm từ quá thường xuyên trong kết quả của nó. Nếu chúng ta muốn mô hình tạo ra văn bản đa dạng hoặc sáng tạo, chúng ta có thể sử dụng Presence Penalty cao hơn hoặc nếu chúng ta cần mô hình tập trung, thì sử dụng Presence Penalty thấp hơn.

Dưới đây là một số đề xuất của nhóm để khởi tạo các tham số này:

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

- **Presence Penalty**

- **Để tạo kết quả đa dạng:** Sử dụng giá trị cao (0.6 - 1.0).
- **Để duy trì độ chính xác của mô hình:** Sử dụng giá trị thấp (0.0 - 0.4).

2 Prompt Engineering

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

Phần II: Prompt Techniques

3 Zero-Shot Prompting

Zero-Shot Prompting là một phương pháp nhằm cải thiện zero-shot learning trong LLMs. Zero-shot learning cho phép mô hình hiểu và thực hiện các nhiệm vụ mà nó không nhìn thấy một cách rõ ràng trong quá trình huấn luyện. Phương pháp này bao gồm việc tinh chỉnh các LMs đã được huấn luyện trước trên một loạt các nhiệm vụ NLP được mô tả trong prompt bằng ngôn ngữ tự nhiên. Cách tiếp cận này đã được chứng minh là cải thiện đáng kể khả năng của mô hình trong việc thực hiện các nhiệm vụ chưa được nhìn thấy (unseen tasks). Hiệu quả của việc tinh chỉnh prompt được thể hiện qua sự cải thiện hiệu suất đáng kể so với các LMs, lợi ích càng trở nên rõ rệt khi số lượng nhiệm vụ tăng lên và khi được áp dụng với LLM. Trong bài báo "FINETUNED LANGUAGE MODELS ARE ZERO-SHOT LEARNERS" kết luận rằng việc tinh chỉnh prompt là một kỹ thuật triển vọng để cải thiện khả năng tổng quát hóa và hiệu suất của LLMs trong các tình huống zero-shot learning.

You

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

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

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

Trân trọng,

Mark."

ChatGPT

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

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

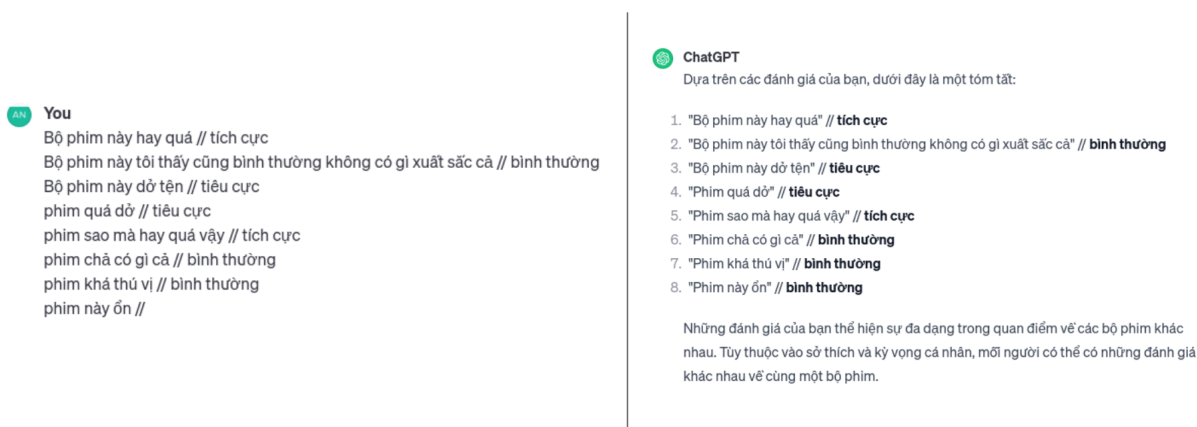
Tổng thể, thái độ của Mark qua email này là tích cực và chuyên nghiệp trong cách xử lý phản hồi từ khách hàng.

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

4 Few-Shot Prompting

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

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



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

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

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

5 Chain-of-Thought Prompting

Phương pháp "Chain of Thought" (COT) được giới thiệu này như một cách để cải thiện hiệu suất của các mô hình ngôn ngữ lớn (LLMs) trong việc giải quyết các bài toán đòi hỏi suy luận phức tạp. Phương

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

6 Zero-shot Chain-of-Thought Prompting

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

7 Automatic Chain-of-Thought Prompting

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

8 Self-Consistency

Self-Consistency được đề xuất trong paper "SELF-CONSISTENCY IMPROVES CHAIN OF THOUGHT REASONING IN LANGUAGE MODELS" giúp cải thiện khả năng suy luận của LLMs bằng cách sử dụng COT. Nó hoạt động bằng cách tạo ra nhiều hướng suy nghĩ đa dạng và sau đó chọn câu trả lời phổ biến nhất từ những hướng suy nghĩ đó. Điều này giúp tăng cường khả năng suy luận chính xác của mô hình, đồng thời cung cấp cái nhìn sâu sắc và dễ hiểu về quá trình suy nghĩ của mô hình, giúp người dùng có thể kiểm tra và hiểu rõ cách mô hình đưa ra kết quả.

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

- **Đa dạng hóa hướng suy nghĩ:** Self-Consistency sinh ra một tập hợp đa dạng các hướng suy nghĩ, thay vì chỉ chọn một suy nghĩ duy nhất. Điều này dựa trên quan điểm rằng một bài toán suy luận phức tạp thường có nhiều cách suy nghĩ khác nhau dẫn đến một câu trả lời đúng duy nhất.

- **Cải thiện hiệu suất suy luận:** Self-Consistency đã được chứng minh là cải thiện hiệu suất của COT một cách đáng kể trên các bài toán suy luận thông thường và toán học.
- **Khắc phục hạn chế của COT:** Nghiên cứu cho thấy rằng COT có thể làm giảm hiệu suất so với gợi ý thông thường trong một số trường hợp. Self-Consistency giúp khắc phục điểm yếu này, đặc biệt trong các tác vụ few-shot learning.
- **Ứng dụng rộng rãi:** Self-Consistency được ứng dụng trong nhiều tác vụ NLP, như trả lời câu hỏi không dựa trên nguồn thông tin (Closed-Book Question Answering) và suy luận ngôn ngữ tự nhiên (Natural Language Inference)

9 Generated Knowledge Prompting

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

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

- **Bước Tạo Kiến Thức (Knowledge Generation):** Sử dụng mô hình ngôn ngữ để sinh ra các phát biểu kiến thức dựa trên câu hỏi $K_q = \{k_m : k_m \sim p_G(k|q), m = 1 \dots M\}$, với mỗi phát biểu kiến thức k_m là một chuỗi văn bản có độ dài biến đổi. Bản chất, mỗi phát biểu chứa thông tin hữu ích cho việc trả lời câu hỏi. Khi tạo kiến thức cho một câu hỏi mới, câu hỏi được chèn vào vị trí dành cho câu hỏi mới và lấy mẫu liên tục các tiếp nối của prompt này để thu được một tập hợp các phát biểu kiến thức $K|q$.
- **Bước Tích Hợp Kiến Thức (Knowledge Integration):** Trong bước này, kiến thức sinh ra được tích hợp vào quá trình đưa ra quyết định của mô hình ngôn ngữ sử dụng cho suy luận $\hat{a} = \arg \max_{a \in A_q} \max_{0 \leq m \leq M} p_I(a|q_m)$, Điều này được thực hiện bằng cách sử dụng từng phát biểu kiến thức để gợi ý cho mô hình, tạo ra M câu hỏi được bổ sung kiến thức: $q_0 = q, q_1 = [k_1|q], \dots, q_M = [k_M|q]$, với $[\cdot|\cdot]$ biểu thị sự kết hợp văn bản. Điểm tổng hợp cho mỗi lựa chọn trả lời được tính dựa trên câu hỏi bổ sung hỗ trợ tốt nhất lựa chọn đó dưới mô hình suy luận. Câu trả lời dự đoán cuối cùng là lựa chọn được hỗ trợ nhiều nhất từ một trong các phát biểu kiến thức. Mô hình suy luận có thể là một mô hình ngôn ngữ hiện có hoặc được tinh chỉnh cụ thể cho tác vụ. Không cần tinh chỉnh thêm với kỹ thuật gợi ý kiến thức.

10 Tree of Thoughts

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

ngôn ngữ trên ba nhiệm vụ mới đòi hỏi kế hoạch hoặc tìm kiếm không đơn giản: Game of 24, Creative Writing, and Mini Crosswords.

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

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

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

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

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

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

- **Đánh giá từng trạng thái độc lập:** Mỗi trạng thái được đánh giá độc lập để tạo ra một giá trị vô hướng (ví dụ từ 1 đến 10) hoặc một phân loại. Cách tiếp cận này sử dụng suy luận đánh giá để chuyển đổi một heuristic thành một giá trị. Sự suy luận đánh giá có thể thay đổi tùy thuộc vào vấn đề và các bước suy nghĩ.
- **Bầu chọn các trạng thái:** Đánh giá dựa trên việc so sánh và bầu chọn giữa các trạng thái khác nhau, nơi "trạng thái tốt" là trạng thái nhận được nhiều phiếu bầu nhất từ việc so sánh cẩn thận

giữa các trạng thái. Khi đánh giá bằng cách bầu chọn, LM sẽ xem xét các trạng thái khác nhau và bầu chọn cho trạng thái "ưu tú" nhất.

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

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

Một số lợi ích của TOT:

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

11 Automatic Prompt Engineer (APE)

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

Phương pháp APE hoạt động như sau:

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

Algorithm 1 Automatic Prompt Engineer (APE)

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

1: Use LLM to sample instruction proposals $\mathcal{U} \leftarrow \{\rho_1, \dots, \rho_m\}$. (See Section 3.1)

2: **while** not converged **do**

3: Choose a random training subset $\tilde{\mathcal{D}}_{\text{train}} \subset \mathcal{D}_{\text{train}}$.

4: **for all** ρ in \mathcal{U} **do**

5: Evaluate score on the subset $\tilde{s} \leftarrow f(\rho, \tilde{\mathcal{D}}_{\text{train}})$ (See Section 3.2)

6: **end for**

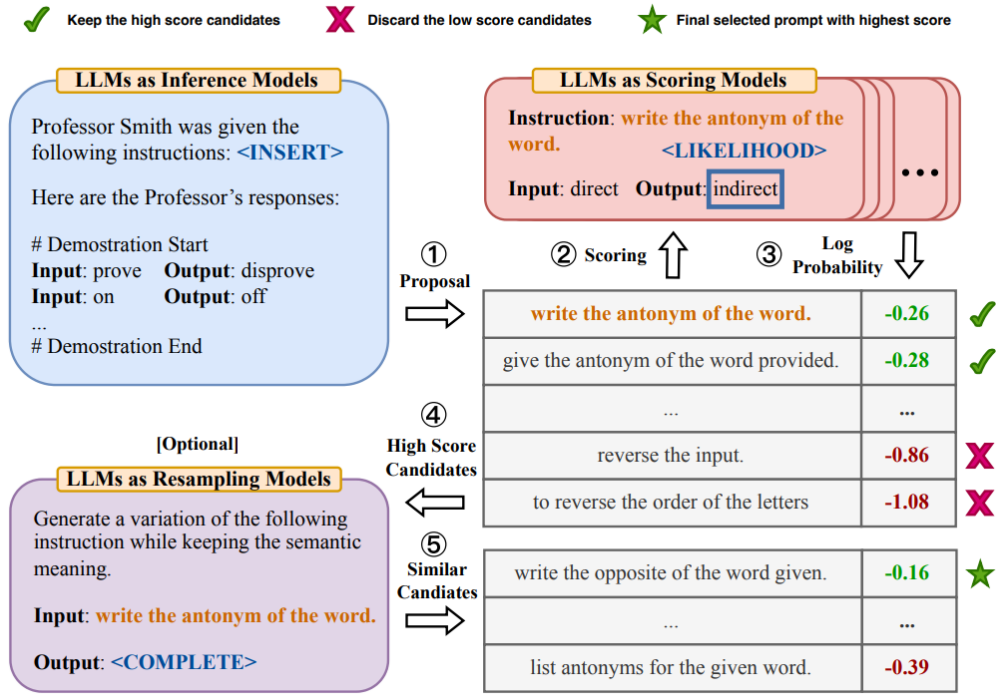
7: Filter the top k% of instructions with high scores $\mathcal{U}_k \subset \mathcal{U}$ using $\{\tilde{s}_1, \dots, \tilde{s}_m\}$

8: Update instructions $\mathcal{U} \leftarrow \mathcal{U}_k$ or use LLM to resample $\mathcal{U} \leftarrow \text{resample}(\mathcal{U}_k)$ (See Section 3.3)

9: **end while**

Return instruction with the highest score $\rho^* \leftarrow \arg \max_{\rho \in \mathcal{U}_k} f(\rho, \mathcal{D}_{\text{train}})$

Hình 6: Thuật toán của APE



Hình 7: Phương pháp APE, các bước được đánh số từ 1 đến 5

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

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

Hình 8: Prompt được tạo sinh bằng phương pháp APE và prompt được định nghĩa bởi con người.

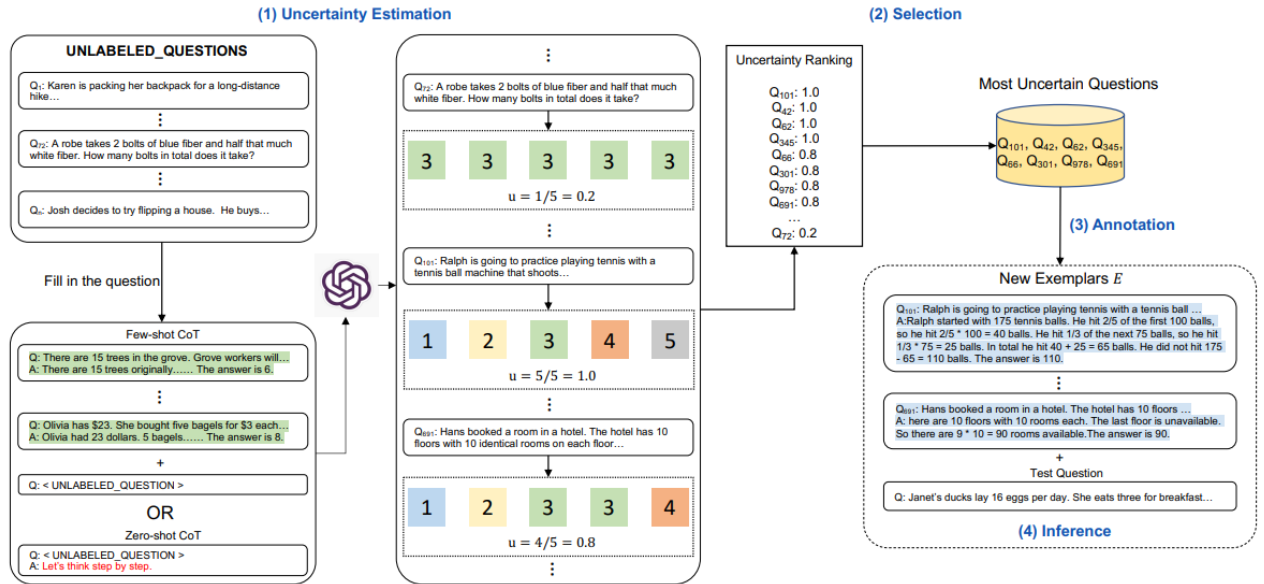
Lợi ích của việc sử dụng APE:

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

12 Active-Prompt

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

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



Hình 9: Các bước trong Active-Prompting

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

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

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

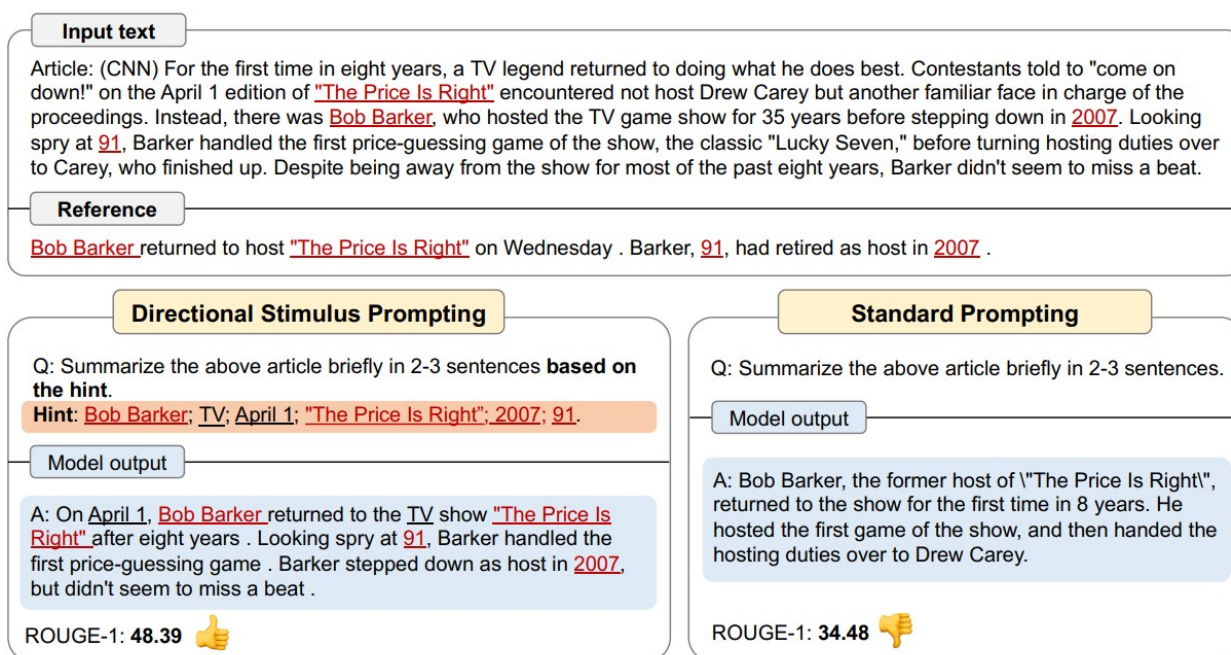
Trong đó $P_{\theta}(a_j | q_i)$ là tần số xuất hiện của một câu trả lời j nhất định trong tập tất cả các dự đoán. Entropy càng cao chứng tỏ độ bất định càng lớn.

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

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

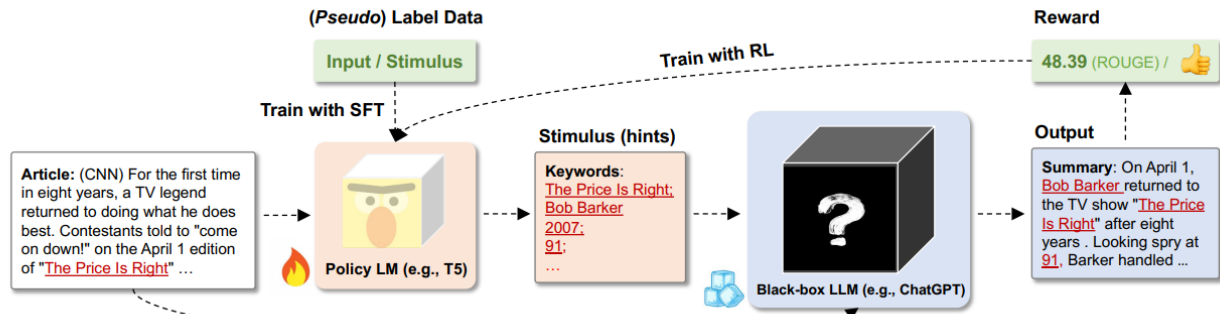
13 Directional Stimulus Prompting

Đây là một kỹ thuật prompting giúp LLM có thể tóm tắt văn bản theo ý muốn. Kỹ thuật này được giới thiệu trong paper "Guiding Large Language Models via Directional Stimulus Prompting" Li et al., 2023. DSP sử dụng các kích thích/chỉ dẫn hướng (directional stimulus, trong trường hợp này là các từ khóa), để cung cấp hướng dẫn cụ thể cho từng trường hợp cho các mô hình ngôn ngữ lớn (LLMs) trong việc tạo ra các bản tóm tắt phù hợp hơn với bản tóm tắt tham khảo mong muốn. Hình 10 minh họa cách hoạt động của DSP.



Hình 10: So sánh giữa DSP và Prompting truyền thống cho tác vụ tóm tắt văn bản. DSP sử dụng các kích thích/chỉ dẫn hướng, trong trường hợp này là các keyword được đánh dấu bằng màu cam, để các mô hình LLM có thể sinh ra bản tóm tắt phù hợp với mong muốn.

Trong paper này, một mô hình language model nhỏ và có thể tune được (chẳng hạn như T5) được sử dụng để tạo sinh các kích thích/chỉ dẫn. Hướng tiếp cận này cho phép tránh việc phải fine-tune một mô hình ngôn ngữ lớn, thay vào đó ta có thể fine-tune một mô hình chỉ dẫn nhỏ hơn một cách dễ dàng hơn. Hướng tiếp cận này được mô tả trong hình 11.



Hình 11: DSP sử dụng một mô hình ngôn ngữ nhỏ hơn để tạo sinh ra các kích thích/chỉ dẫn định hướng, trong trường hợp này là keyword. Mô hình này có thể được huấn luyện bằng SFT (supervised fine-tuning) và/hoặc là RL.

Phần III: Retrieval Augmented Generation (RAG)

Trong bối cảnh các mô hình ngôn ngữ lớn (LLM) phát triển mạnh mẽ, sự xuất hiện của các mô hình GPT (OpenAI), Llama (Meta), Gemini (Google) đã thể hiện khả năng ấn tượng trong việc sinh ngôn ngữ, thực hiện tác vụ với ngôn ngữ tự nhiên. Cho dù vậy, các mô hình ngôn ngữ lớn vẫn cho thấy còn nhiều điểm yếu như dữ liệu thiếu tính cập nhật, thiếu dữ liệu chuyên môn cho các lĩnh vực cụ thể hay sinh ngôn ngữ thiếu chính xác (hay được biết đến với thuật ngữ "hallucination").

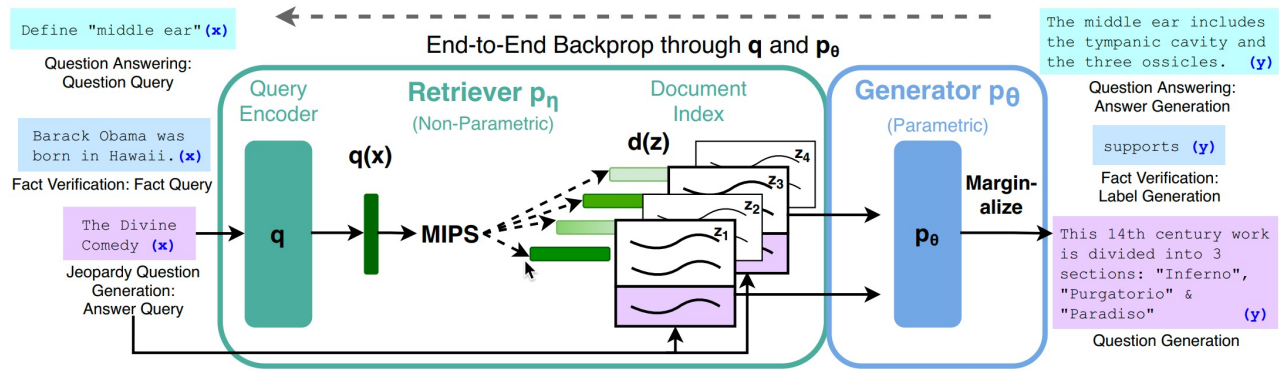
Bên cạnh đó, nhu cầu sử dụng mô hình ngôn ngữ để tương tác với dữ liệu riêng, dữ liệu doanh nghiệp cũng gặp nhiều khó khăn với việc các giải pháp fine-tuning, training LLM bởi chi phí lớn và yêu cầu kỹ thuật cao. RAG ra đời cung cấp giải pháp nhanh chóng, tiện lợi cho phép LLM sử dụng thông tin bổ sung để giao tiếp, ...

14 Khái niệm RAG

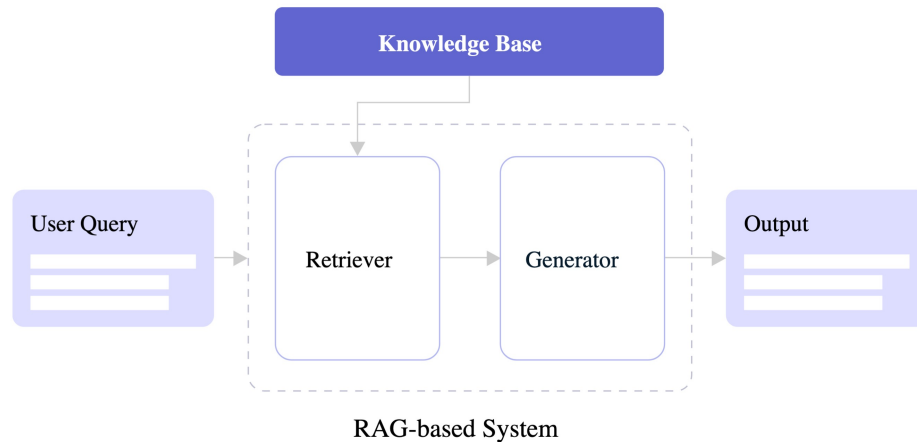
Retrieval Augmented Generation (RAG) lần đầu được giới thiệu bởi nhóm kỹ sư thuộc Meta AI là một kỹ thuật trong lĩnh vực xử lý ngôn ngữ tự nhiên (NLP) nhằm nâng cao độ chính xác và tin cậy của các mô hình tạo văn bản (Generative language models - LLMs). RAG kết hợp hai thành phần chính: cấu phần truy xuất thông tin (Retriever) và mô hình sinh ngôn ngữ (Generator):

- **Truy xuất thông tin (Retrieval):** RAG không chỉ dựa vào dữ liệu đào tạo ban đầu của LLM mà còn truy cập một nguồn kiến thức bên ngoài, thường là các văn bản được xác định trước và có độ tin cậy cao. Khi nhận được yêu cầu, RAG sẽ phân tích nó và tìm kiếm thông tin liên quan trong nguồn kiến thức bên ngoài.
- **Tạo văn bản (Generation):** Dựa trên thông tin đã truy xuất, LLM sẽ tạo ra văn bản phản hồi phù hợp với yêu cầu. Quá trình này có thể bao gồm tóm tắt, giải thích, trả lời câu hỏi, viết văn bản sáng tạo, v.v.

Trong bài báo của Lewis et al. (2021), khi lần đầu đề cập đến hệ thống RAG, Lewis đã sử dụng một mô hình seq2seq huấn luyện trước ...



Hình 12: Tổng quan giải pháp RAG cung cấp bởi Lewis (2020). Giải pháp kết hợp một mô hình truy vấn được huấn luyện trước (Query Encoder + Document Index) và một mô hình seq2seq huấn luyện trước (Generator). Với đầu vào là truy vấn x , Maximum Inner Product Search (MIPS) được sử dụng để tìm ra top- k tài liệu liên quan. Tài liệu tìm thấy được sử dụng trong phần dự báo của mô hình seq2seq.

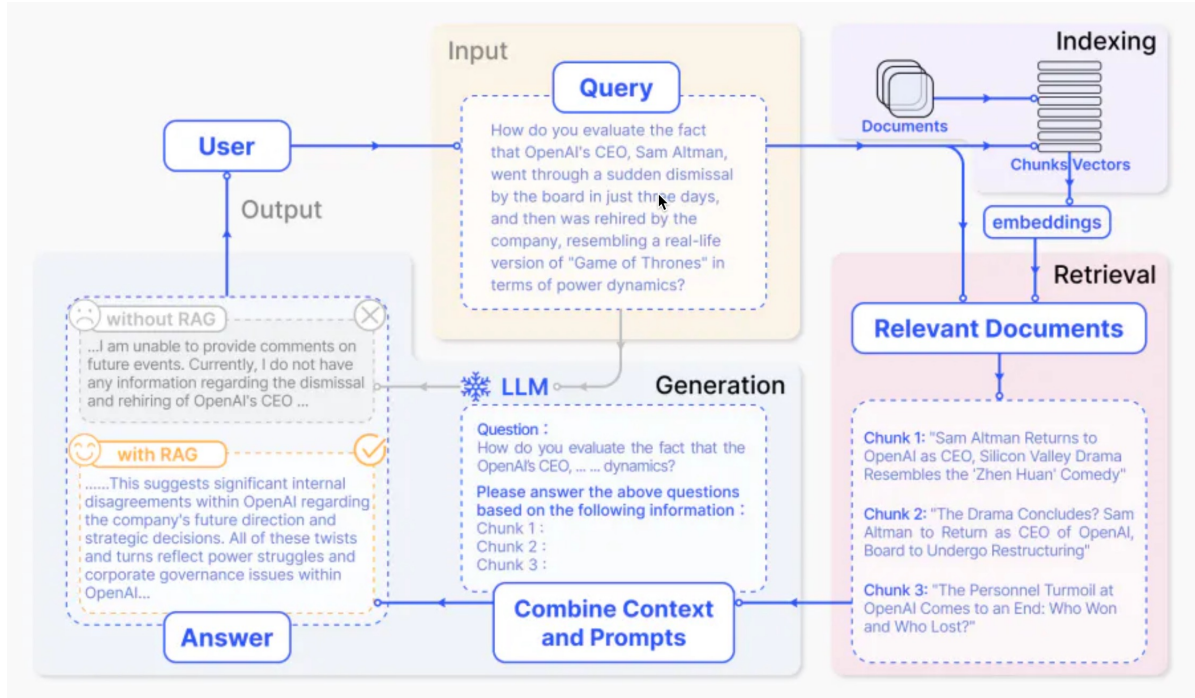


Hình 13: Hệ thống RAG cơ bản với các cấu phần: Nguồn thông tin tra cứu (Knowledge Base), Truy vấn (Retriever), Mô hình sinh (Generator), Truy vấn đầu vào và Kết quả đầu ra.

Hệ thống RAG cho phép tìm kiếm và nhận về các tài liệu liên quan đến câu hỏi truy vấn của người dùng. Các tài liệu này được sử dụng như nguồn thông tin kết hợp với câu hỏi truy vấn để tạo ra câu trả lời cuối cùng thông qua một mô hình sinh ngôn ngữ. Phương pháp này cho phép RAG thích ứng với các yêu cầu mà nguồn thông tin thay đổi theo thời gian thực, truy cập thông tin mới nhất, tạo ra kết quả đáng tin cậy mà không cần phải huấn luyện lại mô hình. Đây cũng là lý do giúp RAG trở thành giải pháp hữu hiệu giải quyết nhược điểm "hallucination" của các mô hình ngôn ngữ huấn luyện trước.

15 Cấu trúc tổng quan

Tương tự như các giải pháp LLM khác, hệ thống RAG tiếp nhận yêu cầu người dùng (User Query) và xử lý để đưa ra kết quả trả về (Output). Ngoài ra hai cấu phần chính của hệ thống RAG là bộ truy vấn (Retriever) và mô hình sinh (Generator). Thông thường, bên cạnh Retriever, một cơ sở dữ liệu phi cấu trúc (Knowledge Base) cần được được thiết lập và xử lý trước.



Hình 14: Ví dụ chi tiết hoạt động của hệ thống RAG. Nguồn: Gao et al. (2024)

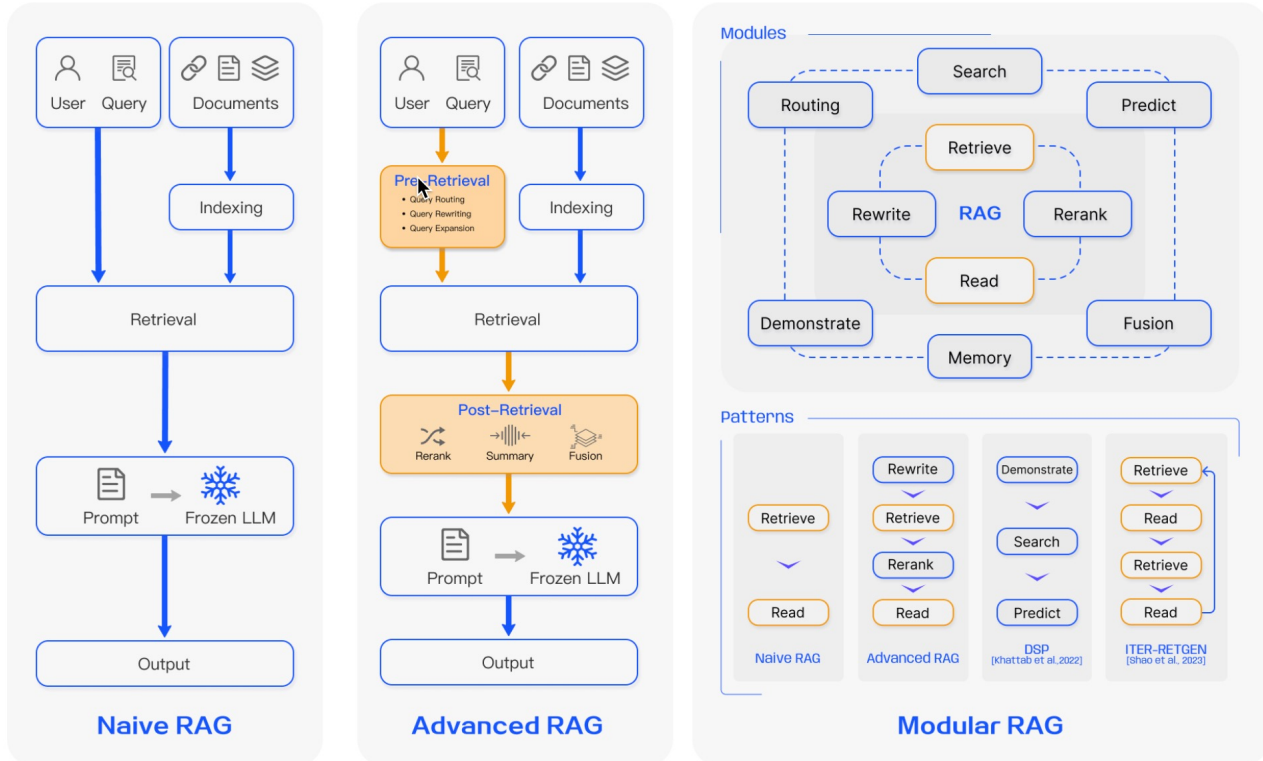
Các thành phần trong ví dụ có thể giải thích như sau:

- **Input:** Câu hỏi đầu vào, thường được cung cấp/yêu cầu bởi người dùng nhằm truy vấn và tương tác với mô hình ngôn ngữ.
- **Indexing:** Trong hệ thống RAG, để xây dựng Knowledge Base, các văn bản/tài liệu cần được xử lý (indexing) và lưu trữ dưới các định dạng dễ dàng truy vấn như vector. Các thao tác cần xử lý có thể kể đến như phân đoạn (chunking), vector hoá (embedding), lưu trữ trên cơ sở dữ liệu vector (vector store database). Trong quá trình truy vấn, câu hỏi người dùng sẽ được embedded để sẵn sàng cho các thuật toán tìm kiếm.
- **Retrieval:** Quá trình tìm kiếm các đoạn tài liệu (chunks) liên quan đến câu hỏi truy vấn. Công tác tìm kiếm thường được thực hiện thông qua phép tìm kiếm mức độ tương đồng (similarity), so sánh vector truy vấn và vector đã được xử lý (index) trong cơ sở dữ liệu vector. Các kết quả có similarity score cao thể hiện mức độ tương quan tốt với câu hỏi truy vấn. Giá trị top-k cũng được thiết lập tại bước này nhằm thể hiện số lượng k tài liệu có score cao nhất được lấy ra từ bước này.
- **Generation:** Khái niệm thể hiện cho các mô hình sinh ngôn ngữ, có thể biết đến như các mô hình ngôn ngữ lớn (từ các mô hình thương mại như GPT - OpenAI, Gemini - Google, Claude - Anthropic cho đến các mô hình mã nguồn mở như Llama - MetaAI, BLOOM, BERT, Falcon, Mixtral, Vicuna, PhoGPT,...). Trong hệ thống RAG, các tài liệu liên quan được đưa vào mô hình ngôn ngữ cùng câu hỏi dưới dạng prompt (ngữ cảnh bổ sung thông tin). Kết quả từ mô hình ngôn ngữ được đưa về người dùng như câu trả lời.

16 Phân loại RAG

Theo Yunfan Gao et al, một khảo sát tổng hợp chi tiết về sự phát triển của các hệ thống RAG đã được thực hiện và ra công bố vào những ngày đầu năm dương lịch 2024 Gao et al. (2024). Theo đó, các hệ

thống RAG có thể được phân loại theo các bước phát triển của mô hình từ RAG cơ bản (Naive) đến RAG nâng cao (Advanced) và Mô-đun RAG (Modular). Các mô hình có sự cập nhật và giải quyết được các giới hạn của mô hình trước về các mặt mức độ thể hiện (performance), thời gian và hiệu quả (cost and efficiency).



Hình 15: Phân loại hệ thống RAG. Nguồn: Gao et al. (2024)

- (a) **RAG cơ bản** bao gồm các bước thực hiện truyền thống từ: xử lý dữ liệu nguồn (indexing), truy vấn (retrieval) và sinh câu trả lời (generation). RAG cơ bản tồn tại các giới hạn như chỉ lệ chính xác chưa cao đến từ việc truy vấn thiếu chính xác các phân đoạn tài liệu (chunks) liên quan. Điều này cũng là nguyên nhân khiến các vấn đề về "tự cho mình đúng" (hallucination) từ các mô hình LLM còn tồn tại.
- (b) **RAG nâng cao** cung cấp các giải pháp giải quyết nhược điểm của RAG cơ bản như cải thiện chất lượng truy vấn bằng cách thực thi các giải pháp tiền truy vấn, trong truy vấn và sau truy vấn.
- **Giải pháp tiền truy vấn:** bao gồm tối ưu quá trình xử lý dữ liệu nguồn như tối ưu cấu trúc tài liệu đã xử lý bằng các phương pháp tách tài liệu (chunking khác nhau: sentence splitter, sentence window, semantic splitter, hierarchical,...), bổ sung metadata, tối ưu mô hình embedding bằng embedding fine-tuning,...
 - **Giải pháp truy vấn:** Quá trình truy vấn có thể được cải thiện bằng các giải pháp viết lại câu hỏi truy vấn (Sub-queries), truy vấn mở rộng (sử dụng bộ lọc metadata), truy vấn với các phương pháp khác nhau (full-text search, semantic-search, hybrid-search).
 - **Giải pháp sau truy vấn:** Các kết quả sau khi truy vấn với nhiều phương pháp được xếp hạng lại nhằm đánh giá lại mức độ tương quan trước khi chọn ra top-k tài liệu liên quan nhất.

- (c) **Mô-đun RAG:** Một cách hệ thống, giải pháp RAG bao gồm nhiều cấu phần, mỗi cấu phần thực hiện các chức năng khác nhau. Mô-đun RAG được thiết kế với các mô-đun chức năng nhằm cải thiện chất lượng của các cấu phần thuộc hệ thống RAG. Một cách tổng quan, RAG cơ bản là một giải pháp thuộc RAG nâng cao, RAG nâng cao là một trường hợp của Mô-đun RAG với các chức năng cố định. Các mô-đun RAG bao gồm tìm kiếm (Search), bộ nhớ (Memory), kết hợp (Fusion), (Routing), (Predict), (Task Adaptable Module). Các mô-đun này được sắp xếp để giải quyết các vấn đề cụ thể.

Nhằm tăng tính linh hoạt của hệ thống RAG, một số kỹ thuật có thể áp dụng như:

- **Hybrid Search Exploration:** Là giải pháp kết hợp hai giải pháp tìm kiếm là tìm kiếm với từ khoá (keyword) và tìm kiếm theo ngữ nghĩa (semantic). Giải pháp này tối ưu được kết quả chính xác từ việc tìm theo từ khoá và việc linh hoạt tìm kiếm với các từ cùng ngữ cảnh. Kết hợp với giải pháp này thường là các thuật toán xếp hạng (reranker) để phân hạng lại các kết quả có được từ hai phương pháp tìm kiếm.
- **Recursive Retrieval and Query Engine:** Giải pháp cung cấp phương pháp phân chia tài liệu hiệu quả với các cấp phân chia. Cấp nhỏ để phân chia thành các đoạn tài liệu ngắn nhằm gia tăng khả năng tìm kiếm các nội dung tương đồng. Sau đó, cấp lớn chứa đoạn tài liệu dài hơn, bao trùm ngữ cảnh tốt hơn (chứa đoạn nhỏ) được trả về như kết quả tìm kiếm. Giải pháp cho phép tìm kiếm chính xác hơn mà vẫn giữ được ngữ cảnh của tài liệu.
- **StepBack-prompt:** [Zheng et al. (2023)]
- **Sub-Queries:** Trong bối cảnh thực tế, rất nhiều loại truy vấn được thực hiện. Nhằm tối ưu việc truy vấn, các giải pháp về truy vấn được đề xuất, từ việc đơn giản hoá truy vấn, truy vấn nhiều cấp hay xác định chủ đích truy vấn. Các giải pháp này nhằm tăng cường mức độ hoàn thiện của đầu vào truy vấn, cũng như nâng cao mức độ chính xác của kết quả truy vấn.
- **Hypothetical Document Embeddings:** Gao et al. (2022) đã đưa ra giải pháp mới với quan điểm tìm kiếm dựa trên một câu trả lời giả thuyết sẽ cho kết quả tốt hơn tìm kiếm trực tiếp với câu hỏi truy vấn. Giải pháp HyDE sử dụng một LLM để tạo ra một câu trả lời giả thuyết cho câu hỏi truy vấn. Sau đó câu trả lời này được sử dụng làm đầu vào cho việc tìm kiếm thông tin liên quan. Kết quả tìm kiếm sẽ được đưa trở lại LLM cùng câu hỏi truy vấn để tạo ra câu trả lời cuối cùng. Trên thực tế, giải pháp này không phải lúc nào cũng hiệu quả do phụ thuộc hoàn toàn vào câu trả lời giả thuyết. Khi chủ đề của câu hỏi không có sẵn trong dữ liệu được huấn luyện trước của LLM, kết quả nhận về thường không được như mong đợi.

17 Cấu phần RAG

Trong nội dung này, chi tiết các giải pháp cho từng cấu phần của hệ thống RAG: Retrieval, Augmented, Generation sẽ được đề cập. Mỗi giải pháp có điểm mạnh riêng và phù hợp với các trường hợp cụ thể.

17.1 Truy vấn - Retrieval

Truy vấn là cấu phần đầu tiên, cũng là cấu phần quan trọng trong việc tổ chức và tìm kiếm tài liệu liên quan với yêu cầu người dùng (user query). Việc xây dựng một quy trình truy vấn hiệu quả luôn là vấn đề then chốt trong mọi hệ thống RAG. Các giải pháp nâng cao chất lượng truy vấn có thể kể đến như: Enhancing Semantic Representations, Aligning Queries and Documents, Aligning Retriever and LLM.

17.2 Augmented

17.3 Generation

17.4 Evaluation

18 Challenges and Future

19 Công cụ RAG

20 Tùy biến RAG

References

- Zhou, Y., Muresanu, A. I., Han, Z., Paster, K., Pitis, S., Chan, H., & Ba, J. (2022). Large language models are human-level prompt engineers.
- Kojima, T., Gu, S. S., Reid, M., Matsuo, Y., & Iwasawa, Y. Large language models are zero-shot reasoners. In: *Advances in neural information processing systems*. 35. 2022, 22199–22213.
- Diao, S., Wang, P., Lin, Y., & Zhang, T. (2023). Active prompting with chain-of-thought for large language models.
- Li, Z., Peng, B., He, P., Galley, M., Gao, J., & Yan, X. (2023). Guiding large language models via directional stimulus prompting. *arXiv preprint arXiv:2302.11520*.
- Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., tau Yih, W., Rocktäschel, T., Riedel, S., & Kiela, D. (2021). Retrieval-augmented generation for knowledge-intensive nlp tasks.
- Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., Dai, Y., Sun, J., Guo, Q., Wang, M., & Wang, H. (2024). Retrieval-augmented generation for large language models: A survey.
- Zheng, H. S., Mishra, S., Chen, X., Cheng, H.-T., Chi, E. H., Le, Q. V., & Zhou, D. (2023). Take a step back: Evoking reasoning via abstraction in large language models.
- Gao, L., Ma, X., Lin, J., & Callan, J. (2022). Precise zero-shot dense retrieval without relevance labels.

- Hết -