

AI VIET NAM – COURSE 2023

Foundation of Prompt Engineering

Ngày 19 tháng 3 năm 2024

Phần I: Tổng quan về RAG

Phần II: 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").

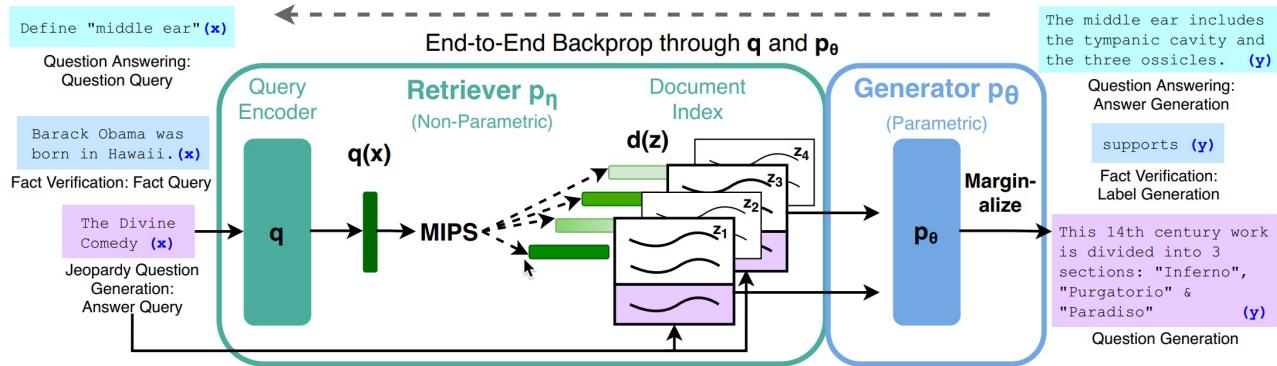
Bên cạnh đó, nhu cầu sử dụng mô hình ngôn ngữ để tương tác với dữ liệu riêng, dữ liệu doanh nghiệp cũng gặp nhiều khó khăn với việc các giải pháp fine-tuning, training LLM bởi chi phí lớn và yêu cầu kỹ thuật cao. RAG ra đời cung cấp giải pháp nhanh chóng, tiện lợi cho phép LLM sử dụng thông tin bổ sung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

1 Khái niệm RAG

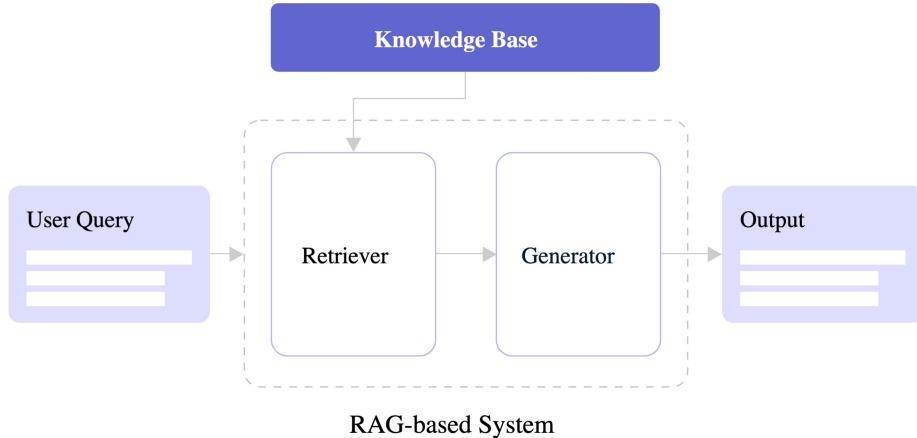
Retrieval Augmented Generation (RAG) lần đầu được giới thiệu bởi nhóm kỹ sư thuộc Meta AI là một kỹ thuật trong lĩnh vực xử lý ngôn ngữ tự nhiên (NLP) nhằm nâng cao độ chính xác và tin cậy của các mô hình tạo văn bản (Generative language models - LLMs). RAG kết hợp hai thành phần chính: cấu phần truy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

- **Truy xuất thông tin (Retriever):** 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 (Generator):** 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 1: 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.

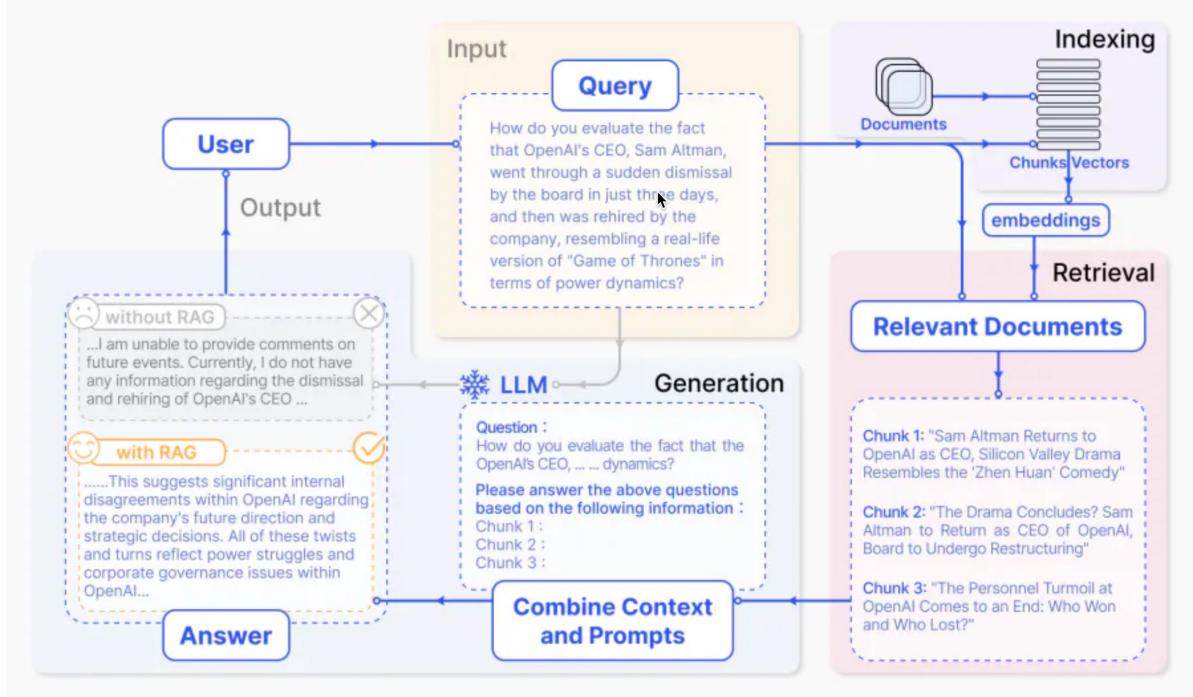


Hình 2: Hệ thống RAG cơ bản với các cấu phần: Nguồn thông tin tra cứu (Knowledge Base), Truy vấn (Retriever), Mô hình sinh (Generator), Truy vấn đầu vào và Kết quả đầu ra.

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.

2 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 thiết lập và xử lý trước.



Hình 3: Ví dụ chi tiết hoạt động của hệ thống RAG. Nguồn: Gao et al. (2024)

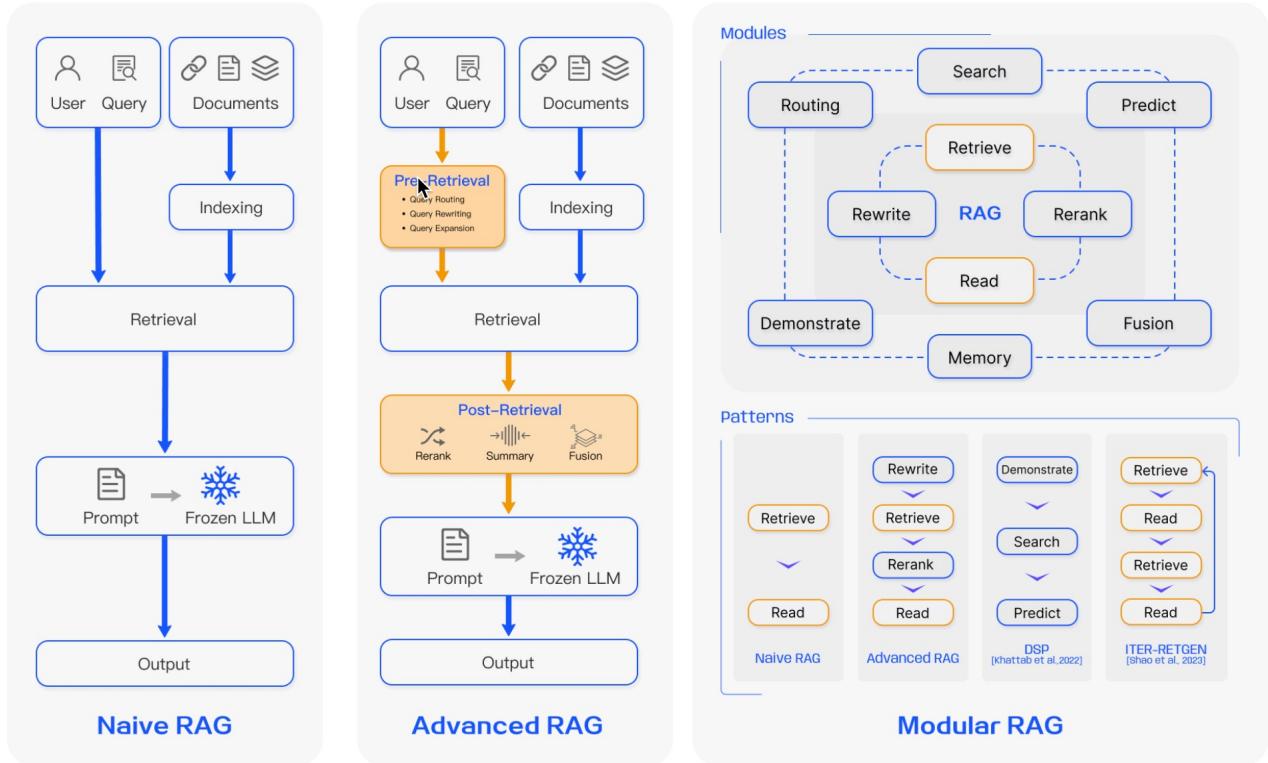
Các thành phần trong ví dụ có thể giải thích như sau:

- **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.

3 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 4: Phân loại hệ thống RAG. Nguồn: Gao et al. (2024)

- 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.
- 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 chùm 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.
- **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 hóa 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.

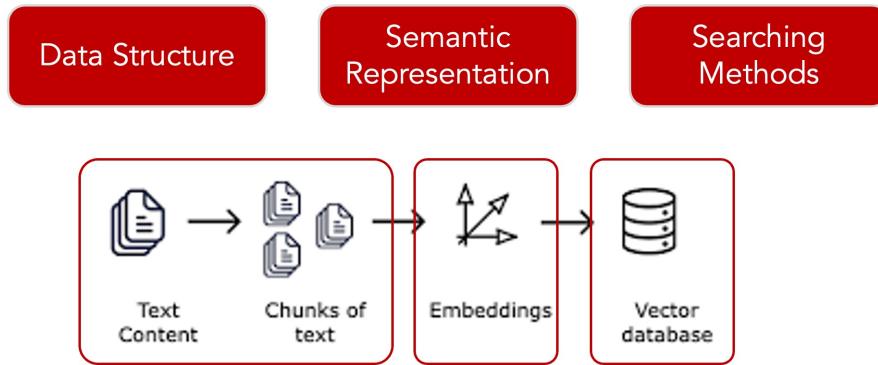
4 Cấu phần RAG

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.

4.1 Tăng cường (Augmented) và Truy vấn (Retrieval) dữ liệu

Tăng cường và truy vấn 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.

Trong các mô hình RAG, công tác tổ chức truy vấn bao gồm cả nhiệm vụ tăng cường (augmented) tổ chức cơ sở dữ liệu (xử lý dữ liệu thô, lưu trữ, duy trì ngữ nghĩa,...) và nhiệm vụ tìm kiếm tài liệu (tra cứu thông tin liên quan theo yêu cầu của người dùng). Dưới góc nhìn của người viết, ba cấu phần chính của công tác Truy vấn được phân loại bao gồm: Xử lý dữ liệu đầu vào (Data Structure), Xử lý lưu trữ và đại diện ngữ nghĩa (Semantic Representation), Truy vấn - tìm kiếm (Searching).



Hình 5: Các nhiệm vụ trong công tác Truy vấn - Retrieval

4.1.1 Xử lý dữ liệu

Các mô hình RAG tốt thường thích ứng với đa dạng các định dạng dữ liệu đầu vào. Tuy nhiên, một số bài toán cho dữ liệu riêng cần được thiết kế công cụ đọc và bóc tách dữ liệu hiệu quả.

Về phía các mô hình ngôn ngữ lớn (LLM) ở thời điểm hiện tại đều gặp giới hạn về độ dài ngữ cảnh (context) văn bản mà LLM có thể xử lý. Ví dụ: giới hạn 128k với GPT4 phiên bản mới nhất, 16k với GPT3.5, 32k với Gemini, 32k Llama-2, 8k Mixtral 8x7b,...

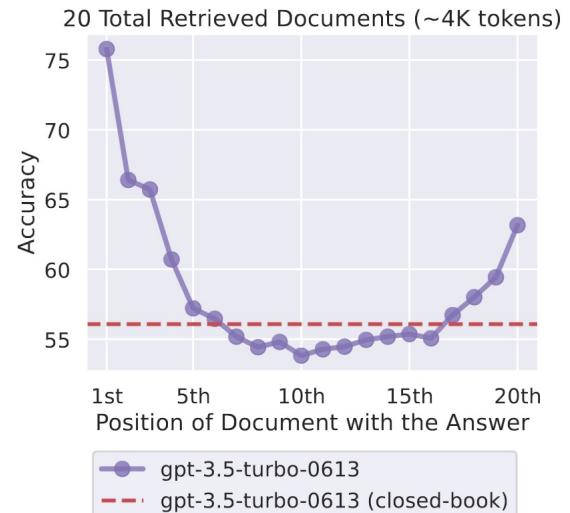
Ngoài ra, việc đưa lượng lớn dữ liệu vào ngữ cảnh cho LLM được chứng minh là không hiệu quả với một số vị trí. Liu et al. (2023) bằng cách thử nghiệm với các ngữ cảnh dài đã chỉ ra rằng việc sử dụng ngữ cảnh dài sẽ gặp khó khăn với các câu hỏi mà thông tin liên quan nằm giữa ngữ cảnh. Các câu hỏi sử dụng thông tin nằm ở đầu và cuối của ngữ cảnh thường cho kết quả tốt hơn.

Để giải quyết vấn đề này, việc chia nhỏ dữ liệu đầu vào với tham số độ dài và chiến lược phân chia (chunking) phù hợp là giải pháp vô cùng quan trọng. Đề bài đặt ra là xây dựng phép chia tài liệu mà ở đó kết quả sau khi phân chia, các thông tin nằm trong một đoạn phải có ý nghĩa liên quan với nhau và việc phân tách không được làm mất thông tin.

Đi từ giải pháp RAG cơ bản, việc phân đoạn đơn giản là phân chia đoạn theo độ dài (chunk size) với đơn vị là tokens, hay bổ sung phân đoạn chồng lấn (overlap chunk) nhằm giữ lại thông tin giữa các chunks, phân chia theo câu (Sentence Splitter).

Tuy nhiên các giải pháp phân chia cơ bản thường gặp nhiều vấn đề như không các đoạn chia (chunk) không chứa thông tin liên quan. Độ dài của các phân đoạn thường là cố định nên không phù hợp. Nếu độ dài quá lớn sẽ dẫn đến khó tìm kiếm với thông tin chi tiết. Nếu độ dài quá ngắn sẽ dẫn đến khó có được bối cảnh tổng quan.

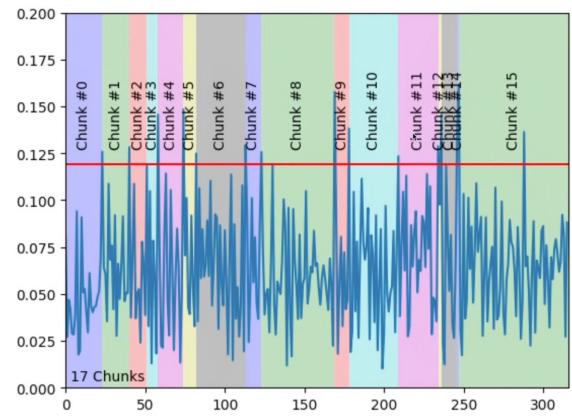
Với giải pháp RAG nâng cao, các hướng tiếp cận hiệu quả hơn được đề cập nhằm đưa ra các phương án chia nhỏ tối ưu để có thể vừa nâng cao khả năng tìm kiếm chi tiết mà vẫn có được thông tin bối



Hình 6: Hiện tượng Lost in the Middle chỉ ra các mô hình LLM thể hiện kém với việc hỏi đáp liên quan đến thông tin nằm ở giữa ngữ cảnh dài.

cảnh. Độ dài của các phân đoạn có thể linh hoạt ổn định cho phù hợp với nội dung ngữ nghĩa. Một số giải pháp có thể kể đến như sau:

- **Sentence Window:** là giải pháp phân chia theo câu kết hợp cửa sổ mở rộng. Dữ liệu được ngữ nghĩa hoá (word2vec) là đơn vị câu, tuy nhiên dữ liệu trả về là dữ liệu mở rộng cho câu lân cận. Tham số độ rộng cửa sổ được khai báo cho phép người dùng tùy chỉnh độ dài văn bản trả về.
- **Hierarchical Splitter:** là giải pháp chia nhỏ dữ liệu theo nhiều cấp. Trong đó, các cấp được phân loại theo độ dài và được ví như các đoạn mẹ (parent) và các đoạn con (child). Các đoạn mẹ có độ dài lớn hơn sẽ bao gồm nhiều đoạn con, đoạn mẹ mang thông tin về ngữ cảnh. Ngược lại, các đoạn con có độ dài ngắn hơn sẽ chứa thông tin chi tiết về các đối tượng, phù hợp hơn cho các phép tìm kiếm chi tiết. Giải pháp này cho phép việc truy vấn có thể tìm kiếm với thông tin chi tiết mà không bị mất đi dữ kiện bối cảnh.
- **Semantic Splitter:** Kamradt (2024) đã giới thiệu giải pháp phân đoạn thông minh nhằm tối ưu sự liên quan của các câu trong văn bản. Giải pháp sử dụng một mô hình embedding để đại diện ngữ nghĩa cho các phân đoạn. Tác giả coi mỗi câu văn như một đơn vị trong một bài toán chuỗi thời gian (Time Series). Một thuật toán chạy lặp qua các cửa sổ được thực hiện để tìm ra điểm ngắt của các phân đoạn không liên quan với nhau về mặt ngữ nghĩa. Theo cách tiếp cận này, việc phân chia văn bản được thực hiện tự động mà vẫn giữ được thông tin liên quan giữa các câu văn trong cùng một phân đoạn (chunk).



Hình 7: Giải pháp Semantic Splitter.

- **Agentic Splitter:** Cũng trong phần trình bày của mình Kamradt (2024) đã giới thiệu giải pháp chia nhỏ Agentic dựa trên bài báo Chen et al. (2023).
- **Enhancing Semantic Representations, Aligning Queries and Documents, Aligning Retriever and LLM**

Xử lý dữ liệu dạng bảng: Bên cạnh các giải pháp xử lý dữ liệu chữ, thông tin từ hệ thống bảng biểu cũng được cân nhắc. Bởi việc bóc tách xử lý chữ thông thường không giữ lại được cấu trúc và mối quan hệ trong các bảng biểu. Các phương pháp đơn giản sẽ chuyển đổi cấu trúc bảng thành các cấu trúc dễ hiểu hơn với máy như markdown hoặc html.

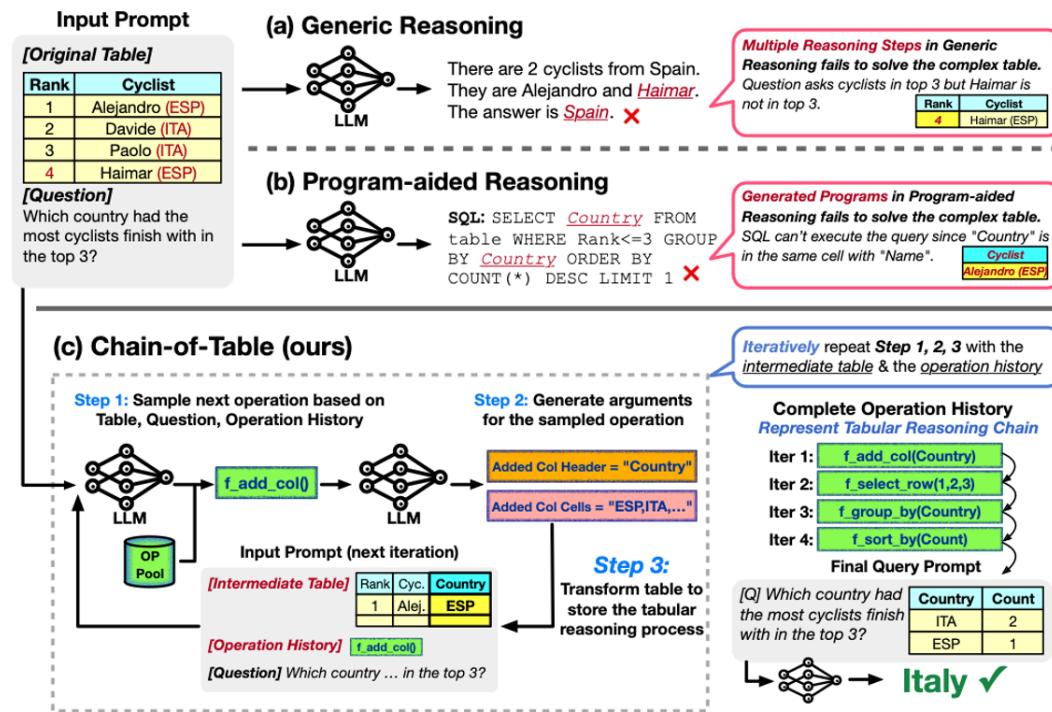
STT	Surveys	Diện tích	Vị trí
1	KS-001	443	112
2	KS-002	358	113
3	KS-003	382	105
4	KS-004	1.307	113
5	KS-004	1.177	114

Markdown			
## Bảng: Thống kê tài liệu			
STT Khảo sát Diện tích (km ²) Vị trí			
--- --- --- ---			
1 KS-001 443 112			
2 KS-002 358 113			
3 KS-003 382 105			
4 KS-004 1.307 113			
5 KS-005 1.177 114			

HTML			
<table><thead><tr><th>STT</th><th>Khảo sát</th><th>Diện tích (km ²)</th><th>Vị trí</th></tr></thead><tbody><tr><td>1</td><td>KS-001</td><td>443</td><td>112</td></tr><tr><td>2</td><td>KS-002</td><td>358</td><td>113</td></tr>...<tr><td>5</td><td>KS-005</td><td>1.177</td><td>114</td></tr></tbody></table>			
Raw-text Extraction			
STT	Surveys	Diện tích	Vị trí
1	KS-001	443	112
2	KS-002	358	113
...			
1.307			
5	KS-005	1.177	114

Hình 8: Xử lý thông tin dạng bảng với phép biến đổi markdown và html. Thông tin trích xuất trực tiếp với Raw-text cho thấy cấu trúc bảng không rõ ràng. Cấu trúc bảng của markdown và html cho phép các LLM dễ hiểu hơn nội dung trong bảng.

Dữ liệu có cấu trúc: Một số giải pháp coi dữ liệu dạng bảng như một nguồn dữ liệu cung cấp các thông tin chính xác. Các giải pháp này tập trung vào việc nâng cao khả năng suy luận và truy vấn thông tin từ các cơ sở dữ liệu có cấu trúc. Wang et al. (2024) đề cập giải pháp có tên gọi Chain of Table, trong đó phương pháp thiết lập một chuỗi các lập luận với LLM để thực thi các truy vấn nhằm đạt được kết quả liên quan nhất với câu hỏi đầu vào.



Hình 9: So sánh giải pháp truy vấn thông tin bảng biểu (a) generic reasoning, (b) program-aided reasoning, and (c) CHAIN-OF-TABLE..

Knowledge Graph: Bên cạnh các kiến trúc lưu trữ dữ liệu truyền thống, Graph database hay

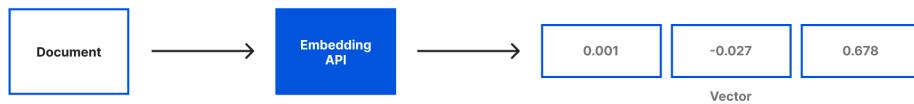
Knowledge Graph được cân nhắc như một giải pháp hiệu quả để quản lý kiến thức. Bởi sự phát triển mới quan hệ theo chiều ngang, việc thiết lập các lập luận (reasoning) được thực hiện dễ dàng hơn. Tuy nhiên, các mô hình Knowledge Graph thường tốn công sức để xây dựng hơn các kiến trúc dữ liệu khác.

4.1.2 Embedding Models

Trong các tác vụ xử lý ngôn ngữ, embeddings đóng vai trò chuyển đổi, đại diện thông tin từ các chủ thể như hình ảnh, âm thanh hay chữ. Trong giải pháp RAG, embeddings được sử dụng trong thuật toán tìm kiếm theo ngữ nghĩa (semantic search).

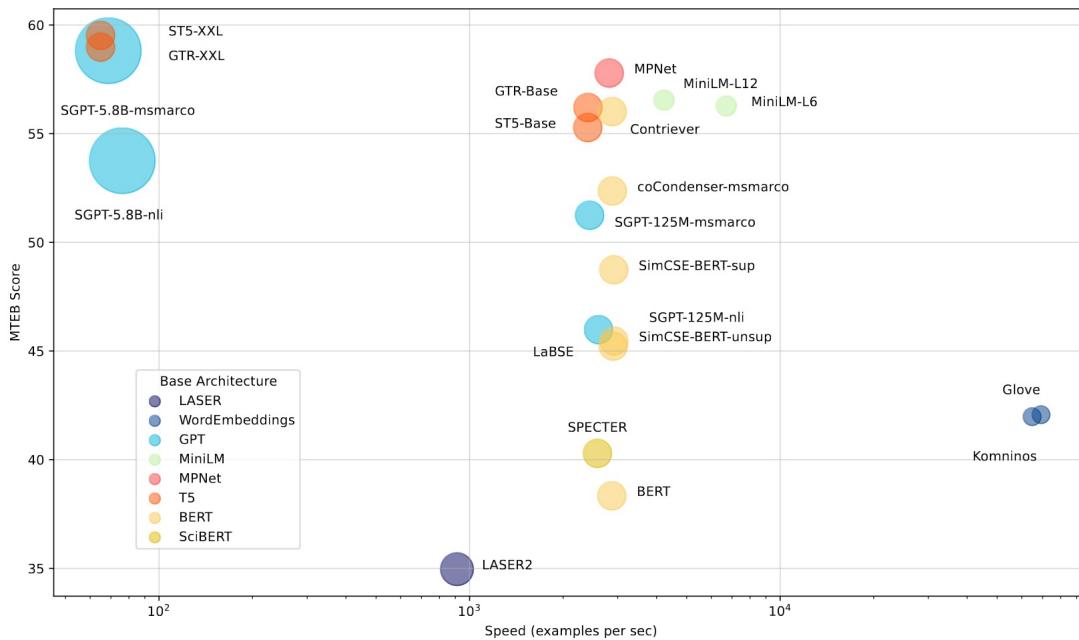
Về mặt toán học, kết quả embeddings được lưu trữ dưới dạng các vector. Mỗi vector là một chuỗi các số, trong đó mỗi số tương ứng với một chiều không gian. Tại các chiều không gian đó, thuật toán tìm kiếm tương đồng (similar) được thực thi để tìm ra các vector gần nhau trong cơ sở dữ liệu vector.

Như vậy, mục tiêu của embeddings là việc đại diện hóa thông tin từ ngữ cảnh (chữ) sang không gian vector sao cho các thông tin có nghĩa giống nhau sẽ có khoảng cách gần nhau.



Hình 10: Embedding đại diện thông tin văn bản dưới dạng vector.

Các mô hình embedding: Muennighoff et al. (2023) đã thực hiện đánh giá 08 tác vụ embedding trên 15 tập dataset, phủ 112 ngôn ngữ. Kết quả của đánh giá được thể hiện trên một bảng xếp hạng chất lượng của các mô hình embeddings.



Hình 11: Kết quả đánh giá mô hình embeddings dựa trên mức độ thể hiện, tốc độ, số chiều, kích thước dữ liệu. Dánh giá được thực hiện trên phần cứng Nvidia A100 80GB, CUDA 11.6.

Dựa trên kết quả đánh giá, một bảng xếp hạng các mô hình embedding đã được xây dựng dựa vào các tiêu chí.

The screenshot shows the 'Overall' tab of the MTEB English leaderboard. The table includes columns for Rank, Model, Model Size (GB), Embedding Dimensions, Max Tokens, Average (56 datasets), Classification Average (12 datasets), Clustering Average (11 datasets), and Pair Classification Average (3 datasets). The models listed are SFR-Embedding-Mistral, voyage-lite-02-instruct, GritLM-7B, e5-mistral-7b-instruct, GritLM-8x7B, echo-mistral-7b-instruct-last, mxbai-embed-large-v1, UAE-Large-V1, text-embedding-3-large, and voyage-lite-01-instruct.

Rank	Model	Model Size (GB)	Embedding Dimensions	Max Tokens	Average (56 datasets)	Classification Average (12 datasets)	Clustering Average (11 datasets)	Pair Classification Average (3 datasets)
1	SFR-Embedding-Mistral	14.22	4096	32768	67.56	78.33	51.67	88.54
2	voyage-lite-02-instruct		1024	4000	67.13	79.25	52.42	86.87
3	GritLM-7B	14.48	4096	32768	66.76	79.46	50.61	87.16
4	e5-mistral-7b-instruct	14.22	4096	32768	66.63	78.47	50.26	88.34
5	GritLM-8x7B	93.41	4096	32768	65.66	78.53	50.14	84.97
6	echo-mistral-7b-instruct-last	14.22	4096	32768	64.68	77.43	46.32	87.34
7	mxbai-embed-large-v1	0.67	1024	512	64.68	75.64	46.71	87.2
8	UAE-Large-V1	1.34	1024	512	64.64	75.58	46.73	87.25
9	text-embedding-3-large		3072	8191	64.59	75.45	49.01	85.72
10	voyage-lite-01-instruct		1024	4000	64.49	74.79	47.4	86.57

Hình 12: [Bảng xếp hạng](#) các mô hình embeddings dựa trên kết quả đánh giá của Muennighoff et al. (2023)

Hiệu chỉnh (fine-tuning) mô hình embedding: Trong các giải pháp RAG, hiệu chỉnh embedding là một cách được cân nhắc để cải thiện chất lượng truy vấn. Các mô hình embedding gốc không được huấn luyện với dữ liệu riêng của người dùng nên việc đại diện ngữ nghĩa có thể không hoàn toàn tốt. Công tác hiệu chỉnh mô hình embedding là quá trình cập nhật lại tham số của mô hình với nhằm đưa thêm thông tin dữ liệu riêng vào mô hình.

Các bước thực hiện hiệu chỉnh mô hình bao gồm:

- **Tạo dữ liệu huấn luyện:** Bên cạnh việc xây dựng các tập dữ liệu huấn luyện một cách thủ công, nhiều giải pháp sử dụng các mô hình ngôn ngữ lớn để tạo dataset được sử dụng. Trong đó, LLM tạo ra câu hỏi giả thuyết tương ứng với thông tin chứa trong đoạn văn bản để tạo ra bộ câu hỏi - đoạn chứa thông tin liên quan.
- **Hiệu chỉnh mô hình embedding:** Sử dụng tập dữ liệu được chuẩn bị trước và một mô hình embedding mã nguồn mở với thuật toán SentenceTransformers, việc huấn luyện hiệu chỉnh mô hình được thực hiện cùng các phép đánh giá đảm bảo mô hình đầu ra phù hợp với các tác vụ truy vấn. Code tham khảo cho [hiệu chỉnh embedding model](#) được viết bởi Jerry Liu.
- **Đánh giá mô hình:** Đánh giá được thực thi để kiểm tra chất lượng mô hình sau khi hiệu chỉnh. Thông thường, việc hiệu chỉnh tốt sẽ tăng cường chất lượng của mô hình hồi đáp từ 5-10 phần trăm.

4.1.3 Giải pháp truy vấn (Retrieval)

Dữ liệu sau khi xử lý được lưu trữ trên các cơ sở dữ liệu như Vector database, Knowledge graph hay non-sql (MongoDB). Công việc tiếp theo là thực thi các truy vấn. Mục tiêu là truy vấn được các thông tin liên quan nhất với câu hỏi đầu vào. Một số giải pháp có thể đề cập đến như:

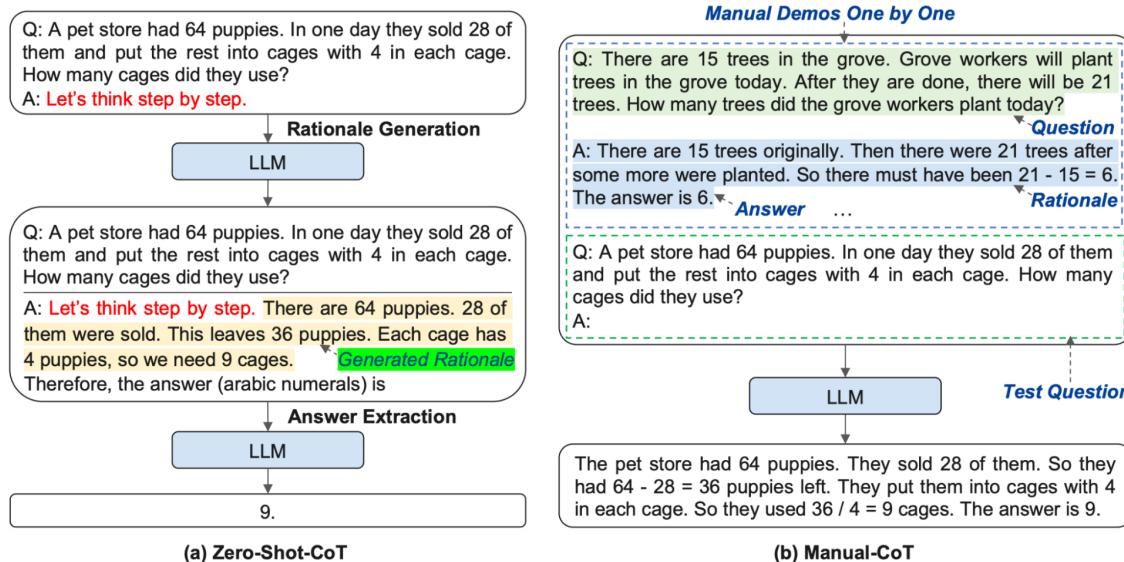
- **Viết lại câu hỏi:** Trong quá trình hỏi đáp, việc làm rõ câu hỏi đầu vào giúp cho việc tổ chức truy vấn hiệu quả. Việc sử dụng một LLM có hướng dẫn (instruction) để viết lại câu hỏi đầu vào của người dùng là giải pháp phù hợp với những câu hỏi quá đơn giản hoặc quá phức tạp.
- **Hypothetical Document Embedding:** Tạo ra câu hỏi giả thuyết để phục vụ embedding và tìm kiếm thông tin liên quan.
- **Hybrid Search and Reranker:** Áp dụng tìm kiếm với từ khoá và ngữ nghĩa sau đó thực hiện xếp hạng lại kết quả tìm kiếm giúp nâng cao kết quả so với việc tìm kiếm đơn thuần với một phương pháp.
- **Metadata Filter:** Giải pháp sử dụng thông tin bổ sung để tạo bộ lọc được sử dụng khi nấm rờ được cấu trúc của cơ sở dữ liệu cũng như nấm được các trường hợp truy vấn. Bộ lọc tìm kiếm giúp thu gọn phạm vi tìm kiếm, nâng cao chất lượng truy vấn.

4.2 Mô hình sinh ngôn ngữ (Generation)

Generation là công đoạn cuối trong chu trình RAG với nhiệm vụ chuyển đổi thông tin nhận được từ bối cảnh thành câu trả lời cho câu hỏi đầu vào của người dùng thông qua một mô hình ngôn ngữ lớn. Quá trình sinh câu trả lời cần đảm bảo câu trả lời đầu ra sử dụng thông tin từ ngữ cảnh được cung cấp, hạn chế câu trả lời sai (hallucination).

Với các yêu cầu về chất lượng, các giải pháp có thể tiếp cận để nâng cao kết quả bao gồm tối ưu hoá kết quả tìm kiếm từ khâu truy vấn (retrieval) bằng các giải pháp hậu truy vấn. Quá trình sinh văn bản có thể được tác động bằng các phương thức:

- **Prompt Engineering:** Prompt Engineering là giải pháp can thiệp vào cách trả lời của LLM mà không cần hiệu chỉnh lại bộ tham số gốc. Prompt Engineering đóng vai trò như một bộ lọc trong quá trình sinh ngôn ngữ đảm bảo kết quả đầu ra theo phong cách trình bày mong muốn. Ngoài ra, một số giải pháp như yêu cầu LLM không tạo ra câu trả lời nếu không có đủ thông tin sẽ giúp hạn chế việc hallucination. Trong trường hợp khác khi ngữ cảnh không đủ dữ kiện, hãy sử dụng dữ liệu được huấn luyện cũng là giải pháp tốt. Bên cạnh đó, trong các kiến trúc RAG kết hợp Multi-Agents, prompt engineering đóng vai trò cốt lõi trong việc điều hướng Agents để tạo ra sự mượt mà ở giải pháp cuối.



Hình 13: Kojima et al. (2023) sử dụng Zero-Shot-CoT để cải thiện chất lượng câu trả lời của LLM

- **Hiệu chỉnh mô hình ngôn ngữ lớn (Fine-tuning LLM):** Công tác hiệu chỉnh mô hình ngôn ngữ lớn thực hiện việc huấn luyện lại một phần tham số của mô hình LLM gốc. Các dữ liệu huấn luyện được chuẩn bị trước từ nguồn dữ liệu riêng, chưa được học bởi LLM. Công tác huấn luyện sẽ ghi đè một phần tham số trong bộ tham số gốc. Giải pháp Fine-tuning được đánh giá là giải pháp hiệu quả cho chất lượng kiểm soát tốt khi lượng dữ liệu huấn luyện đầy đủ, phủ khắp các trường hợp hỏi đáp. Điểm trừ của giải pháp là giải pháp yêu cầu nhiều công sức cho việc chuẩn bị dữ liệu, nhiều tài nguyên cho việc huấn luyện bổ sung cho mô hình ngôn ngữ lớn. Bên cạnh đó, người xây dựng cũng cần có kiến thức và kỹ thuật làm việc với mô hình tốt, tránh trường hợp kết quả sau khi fine-tuning tệ hơn trước khi fine-tuning.

4.3 Đánh giá hệ thống RAG

Giống với các loại mô hình khác, việc đánh giá mô hình luôn là yếu tố quan trọng để đánh giá chất lượng của giải pháp. Thông thường, hệ thống RAG được đánh giá bằng việc đo lường mức độ thể hiện trong các tác vụ cuối (hỏi đáp). Hệ thống đánh giá RAG đo lường hai yếu tố chính là khả năng truy vấn dữ liệu (retrieval) và khả năng sinh câu trả lời (generation).

Công tác đánh giá RAG tập trung vào ba yếu tố đo lường và bốn khả năng. Ba yếu tố đo lường gồm "context relevance" (sự chính xác của kết quả tìm kiếm), "answer faithfulness" (sự thành thực của câu trả lời với ngữ cảnh), "answer relevance" (sự liên quan giữa câu trả lời và câu hỏi). Bên cạnh đó, bốn khả năng được cân nhắc bao gồm: "noise robustness", "negative rejection", "information integration", và "counterfactual robustness".

Table 2: Summary of metrics applicable for evaluation aspects of RAG

	Context Relevance	Faithfulness	Answer Relevance	Noise Robustness	Negative Rejection	Information Integration	Counterfactual Robustness
Accuracy	✓	✓	✓	✓	✓	✓	✓
EM					✓		
Recall	✓						
Precision	✓			✓			
R-Rate							✓
Cosine Similarity			✓				
Hit Rate	✓						
MRR	✓						
NDCG	✓						

Table 3: Summary of evaluation frameworks

Evaluation Framework	Evaluation Targets	Evaluation Aspects	Quantitative Metrics
RGB [†]	Retrieval Quality	Noise Robustness	Accuracy
	Generation Quality	Negative Rejection	EM
		Information Integration	Accuracy
		Counterfactual Robustness	Accuracy
RECALL [†]	Generation Quality	Counterfactual Robustness	R-Rate (Reappearance Rate)
RAGAS [‡]	Retrieval Quality	Context Relevance	*
	Generation Quality	Faithfulness	*
		Answer Relevance	Cosine Similarity
ARES [‡]	Retrieval Quality	Context Relevance	Accuracy
	Generation Quality	Faithfulness	Accuracy
		Answer Relevance	Accuracy
TruLens [‡]	Retrieval Quality	Context Relevance	*
	Generation Quality	Faithfulness	*
		Answer Relevance	*

[†] represents a benchmark, and [‡] represents a tool. * denotes customized quantitative metrics, which deviate from traditional metrics. Readers are encouraged to consult pertinent literature for the specific quantification formulas associated with these metrics, as required.

Hình 14: Gao et al. (2024) đã hệ thống các giải pháp đánh giá hệ thống RAG và danh sách các công cụ sử dụng để đánh giá.

Việc tự động hoá công tác đánh giá các hệ thống RAG ngày càng phổ biến với sự xuất hiện của các công cụ như RAGAS, ARES, TruLens.

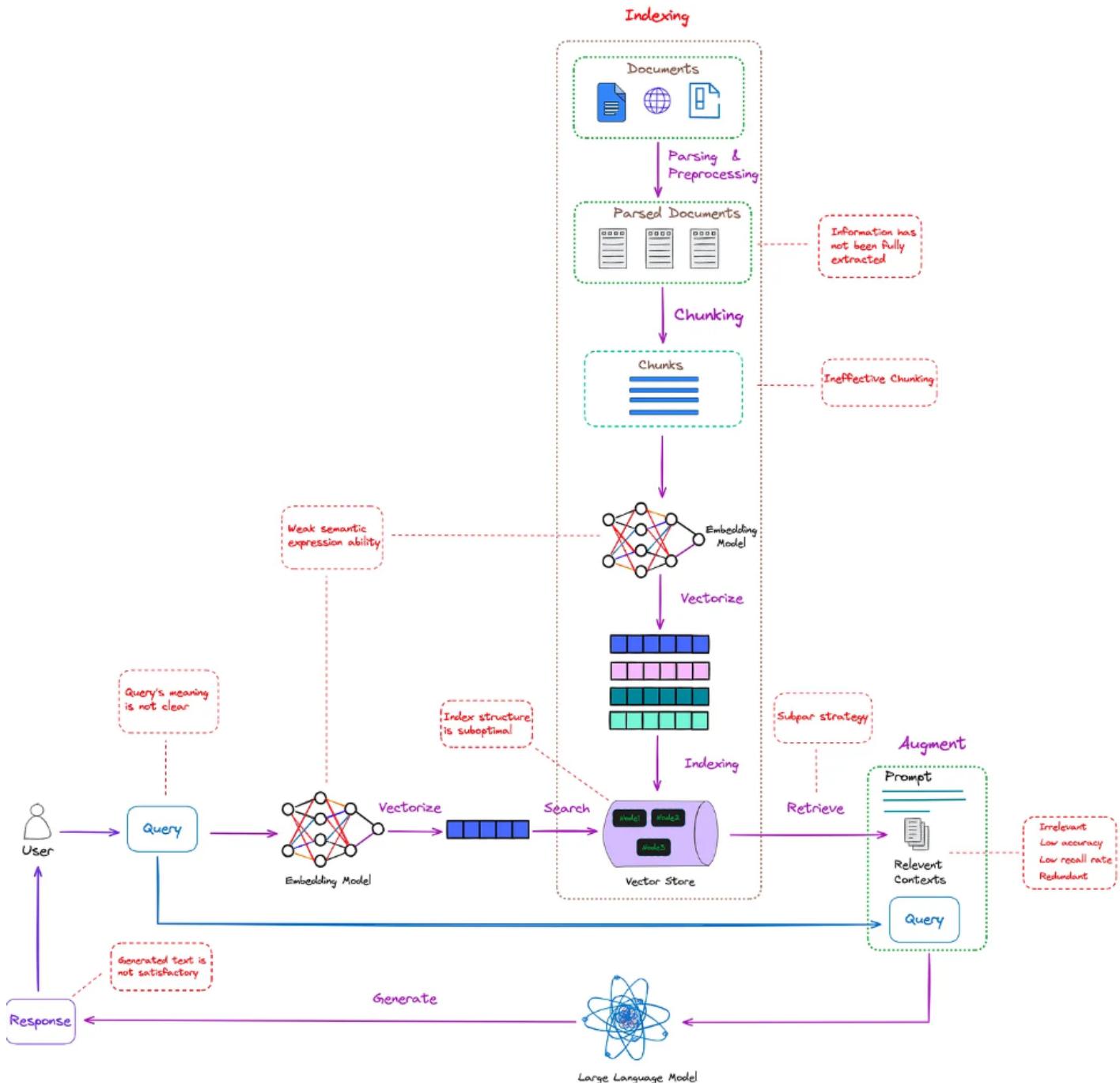
Phần III: Advanced RAG

Ở phần này chúng ta sẽ cùng tìm hiểu về vấn đề của Naive RAG và một số kỹ thuật RAG nâng cao nhằm nâng cao chất lượng cho một hệ thống sử dụng RAG.

5 Vấn đề của Naive RAG

RAG đã nhanh chóng được ứng dụng rộng rãi trong nhiều sản phẩm và dịch vụ, nó đóng vai trò quan trọng trong việc cải thiện khả năng tương tác và trải nghiệm người dùng bằng cách cung cấp câu trả lời chính xác và tức thời cho các truy vấn phức tạp. Tuy nhiên, sự phổ biến và tính ứng dụng rộng rãi

của RAG cũng dẫn đến một loạt thách thức cần giải quyết. Vấn đề chính mà RAG đối mặt là việc tối ưu hóa hiệu suất của nó, cụ thể là làm thế nào để quá trình truy xuất thông tin diễn ra nhanh chóng hơn và kết quả thu được chính xác hơn. Bây giờ chúng ta sẽ tìm hiểu một số lý do mà một hệ thống Naive RAG sẽ gặp phải làm ảnh hưởng tới hiệu suất của nó.



Hình 15: Một số vấn đề của Naive RAG

Với hình trên chúng ta thấy rằng cả ba quá trình Indexing, Retrieval và Generation đều gặp phải những vấn đề riêng của chúng. Những vấn đề này có thể là độc lập hoặc liên quan với nhau để "cùng

nhau" làm giảm hiệu suất của hệ thống RAG.

Với quá trình Indexing:

- Quá trình Indexing còn chưa hoàn thiện, vì nó chưa xử lý hiệu quả thông tin hữu ích trong hình ảnh, biểu đồ và bảng biểu trong các tệp dữ liệu không cấu trúc như PDF.
- Quá trình chunking sử dụng chiến lược “one-size-fits-all” thay vì chọn lựa các chiến lược tối ưu dựa trên đặc điểm của các loại tệp khác nhau. Điều này đã dẫn đến việc mỗi phần chứa thông tin ngữ nghĩa chưa đầy đủ. Hơn nữa, nó không xem xét các chi tiết quan trọng, như các tiêu đề hiện có trong văn bản.
- Cấu trúc indexing chưa được tối ưu hóa đủ mức, dẫn đến chức năng truy xuất chưa hiệu quả.
- Khả năng biểu diễn ngữ nghĩa của embedding model chưa đủ mạnh.

Đối với quá trình Retrieval

- Mức độ liên quan của các ngữ cảnh được ghi nhớ không đủ và độ chính xác thấp.
- Recall Rate thấp ngăn cản việc truy vấn tất cả các đoạn văn bản liên quan, do đó cản trở khả năng của LLMs trong việc tạo ra các câu trả lời toàn diện.
- Truy vấn có thể không chính xác hoặc khả năng biểu diễn ngữ nghĩa của embedding model có thể yếu, dẫn đến khả năng không thể truy vấn thông tin có giá trị.
- Thuật toán truy vấn bị hạn chế vì nó không kết hợp các loại phương pháp hoặc thuật toán truy vấn khác nhau, như kết hợp truy vấn từ khóa, ngữ nghĩa và vector.
- Sự trùng lặp thông tin xảy ra khi nhiều ngữ cảnh được truy vấn chứa thông tin tương tự, dẫn đến nội dung lặp lại trong các câu trả lời được tạo ra.

Đối với quá trình Generation

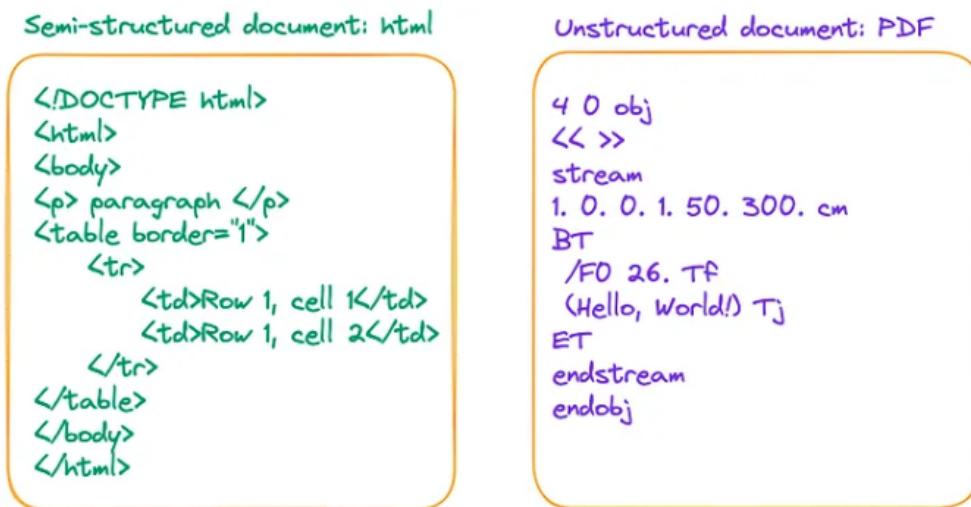
- Việc tích hợp hiệu quả giữa hai tác vụ retrieved context và generation có thể không khả thi, dẫn đến kết quả không nhất quán.
- Sự phụ thuộc quá mức vào thông tin được cải thiện trong quá trình generation mang lại rủi ro cao. Điều này có thể dẫn đến việc tạo ra những kết quả chỉ đơn giản lặp lại nội dung đã truy vấn mà không cung cấp thông tin có giá trị.
- LLM có thể tạo ra các phản hồi sai, không liên quan, có hại, hoặc thiên vị.

6 Unveiling PDF Parsing

Việc trích xuất thông tin từ tài liệu, đặc biệt là từ các tệp PDF không cấu trúc, đã trở thành một yếu tố không thể thiếu trong việc cải thiện chất lượng và hiệu suất của các hệ thống tự động như RAG. Quá trình này yêu cầu một sự chú ý đặc biệt với mục tiêu đảm bảo rằng nội dung được trích xuất một cách chính xác và hiệu quả, từ đó nâng cao giá trị của sản phẩm cuối cùng. Việc ứng dụng công nghệ phân tích cú pháp thông tin từ những tài liệu này là cực kỳ quan trọng, bởi vì dữ liệu không cấu trúc chiếm một tỷ lệ lớn trong dữ liệu được tạo ra và lưu trữ hằng ngày.

Tài liệu PDF, vốn là một trong những dạng phổ biến nhất của dữ liệu không cấu trúc, yêu cầu các phương pháp tiếp cận đặc thù để có thể khai thác trọn vẹn giá trị thông tin bên trong vì vậy việc trích xuất thông tin từ tài liệu PDF lại là một quá trình đầy thách thức. Thay vì là một định dạng dữ liệu, PDF chính xác hơn khi được mô tả là một tập hợp các hướng dẫn in ấn. Một tệp PDF bao gồm một

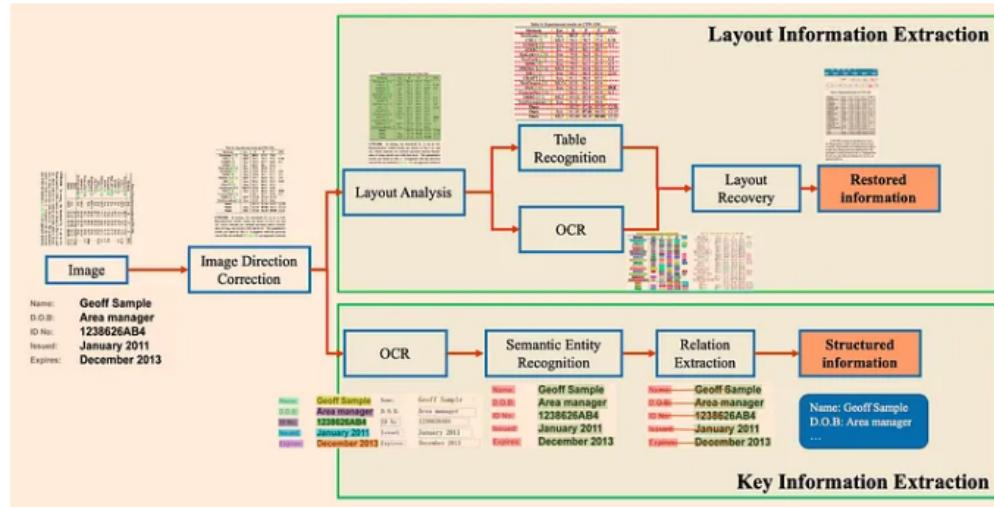
lôt hướng dẫn cho máy đọc hoặc máy in PDF biết cách hiển thị các ký tự trên màn hình hoặc giấy. Điều này trái ngược với các định dạng tệp như HTML và docx, sử dụng các thẻ như `<p>`, `<w:p>`, `<table>`, và `<w:tbl>` để tổ chức các cấu trúc logic khác nhau. Thách thức trong việc phân tích cú pháp tài liệu PDF nằm ở việc chính xác trích xuất bố cục của toàn bộ trang và dịch nội dung, bao gồm bảng, tiêu đề, đoạn văn, và hình ảnh, thành một biểu diễn văn bản của tài liệu. Quá trình này đòi hỏi phải xử lý những khía cạnh xác trong trích xuất văn bản, nhận dạng hình ảnh, và sự nhầm lẫn về mối quan hệ hàng-cột trong bảng.



Hình 16: HTML và PDF

Thông thường chúng ta sẽ có ba cách tiếp cận cơ bản để parse tài liệu PDF:

- Rule-based approach: nơi mà phong cách và nội dung của từng phần được xác định dựa trên đặc điểm tổ chức của tài liệu.
- Deep learning models: Sử dụng một số model kết hợp như Optical Character Recognition, Document Layout Analysis, Key Information Extraction,...
- Phân tích cấu trúc phức tạp hoặc trích xuất thông tin chính trong PDF dựa trên các multimodal large models.



Hình 17: Hình minh họa cho việc kết hợp nhiều Deep learning Model cho quá trình parse PDF

7 Re-ranking

Re-ranking đóng vai trò quan trọng trong RAG. Trong naive RAG, một số lượng lớn ngữ cảnh có thể được truy vấn, nhưng không phải tất cả đều cần thiết liên quan đến câu hỏi. Re-ranking cho phép sắp xếp lại và lọc các tài liệu, đặt những cái liên quan lên hàng đầu, từ đó nâng cao hiệu quả của RAG. Nhiệm vụ của re-ranking là đánh giá mức độ liên quan của các ngữ cảnh này và ưu tiên những cái có khả năng cung cấp câu trả lời chính xác và liên quan nhất. Điều này cho phép LLM ưu tiên những ngữ cảnh được xếp hạng cao này khi tạo ra câu trả lời, từ đó cải thiện độ chính xác và chất lượng của phản hồi.

Các phương pháp re-ranking chủ yếu được chia thành hai loại:

- Re-ranking models: những mô hình này xem xét các đặc điểm tương tác giữa tài liệu và truy vấn để đánh giá mức độ liên quan một cách chính xác hơn.
- Large Language Model: sự xuất hiện của mô hình ngôn ngữ lớn đã mở ra những khả năng mới cho việc re-ranking. Bằng cách hiểu sâu toàn bộ tài liệu và truy vấn, có thể nắm bắt thông tin ngữ nghĩa một cách toàn diện hơn.

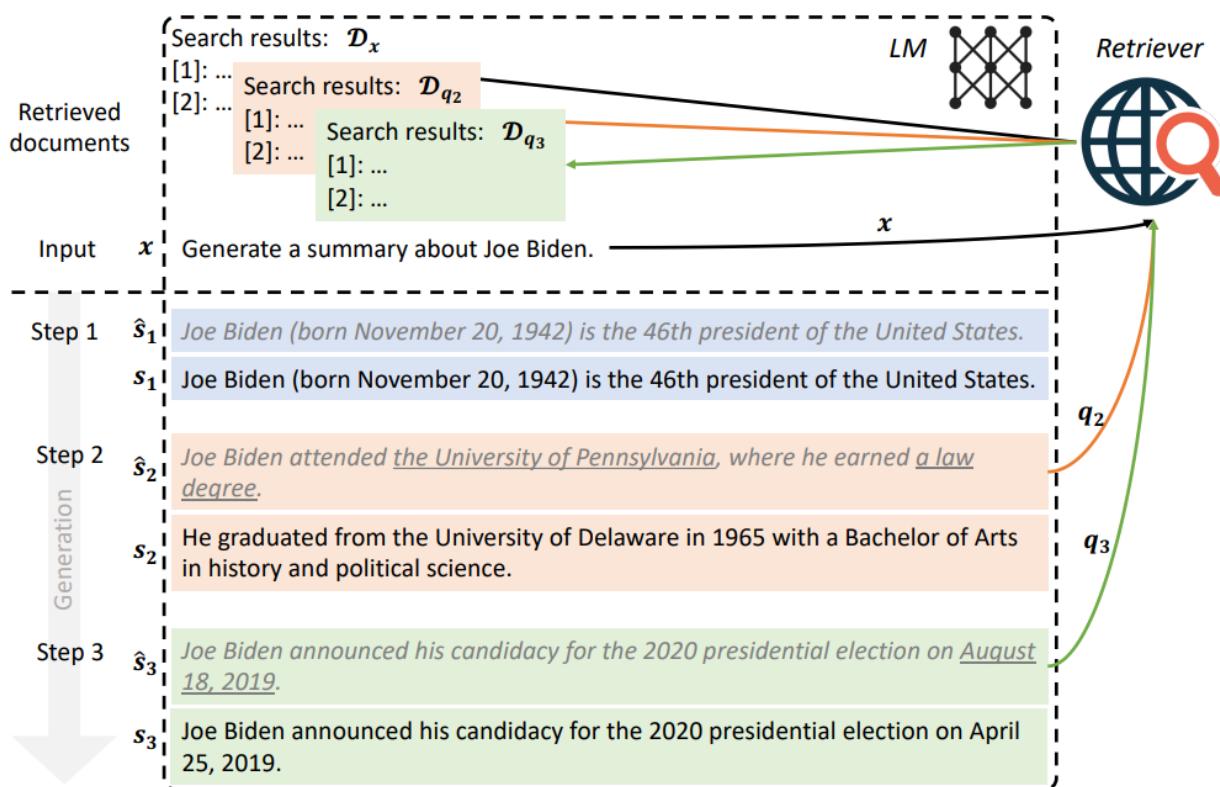
8 Exploring Semantic Chunking

9 Exploring Query Rewriting

Phần VI: Research in RAG

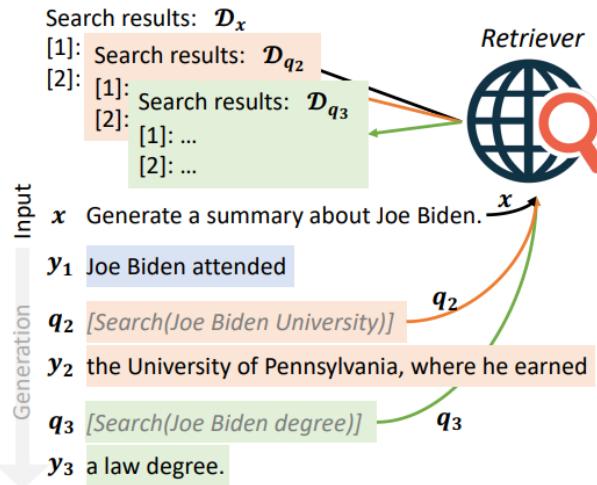
9.1 FLARE

<https://arxiv.org/pdf/2305.06983.pdf>



Trong bài báo "Active Retrieval Augmented Generation" Jiang et al., 2023, các tác giả giới thiệu phương pháp mới có tên FLARE, viết tắt của Forward-Looking Active Retrieval Augmented Generation. Phương pháp này cho phép các mô hình ngôn ngữ lớn tích cực lựa chọn thời điểm và loại thông tin cần truy vấn trong suốt quá trình tạo ra văn bản. FLARE hoạt động bằng cách tạo ra các câu tạm thời, và nếu nhận thấy có từ nào trong câu đó không chắc chắn, nó sẽ tìm kiếm thông tin liên quan để cải thiện câu tiếp theo. Điều này tiếp tục diễn ra cho đến khi hoàn thành văn bản. Một ưu điểm đáng chú ý là FLARE có thể áp dụng cho bất kỳ mô hình ngôn ngữ nào mà không cần phải huấn luyện lại từ đầu. Bài báo cũng trình bày hai cách thức truy vấn chủ động khác nhau: phương pháp đầu tiên, gọi là $FLARE_{instruct}$, tạo ra truy vấn dựa trên một chỉ dẫn đặc biệt để khuyến khích việc truy vấn thông tin, trong khi phương pháp thứ hai, $FLARE_{direct}$, sử dụng kết quả tạo sinh trực tiếp từ mô hình ngôn ngữ làm truy vấn, đặc biệt khi có từ không chắc chắn trong câu đó.

$FLARE_{instruct}$: Một phương pháp đơn giản để tạo ra thông tin truy vấn trong quá trình tạo nội dung là sử dụng token đặc biệt "[Search(query)]" mỗi khi cần thông tin bổ sung. Khi mô hình ngôn ngữ sinh ra token này, quá trình tạo nội dung tạm thời dừng lại, và câu truy vấn bên trong token sẽ được dùng để tìm kiếm thông tin liên quan. Phương pháp này được lấy cảm hứng từ nghiên cứu "Toolformer: Language Models Can Teach Themselves to Use Tools" Schick et al., 2023.

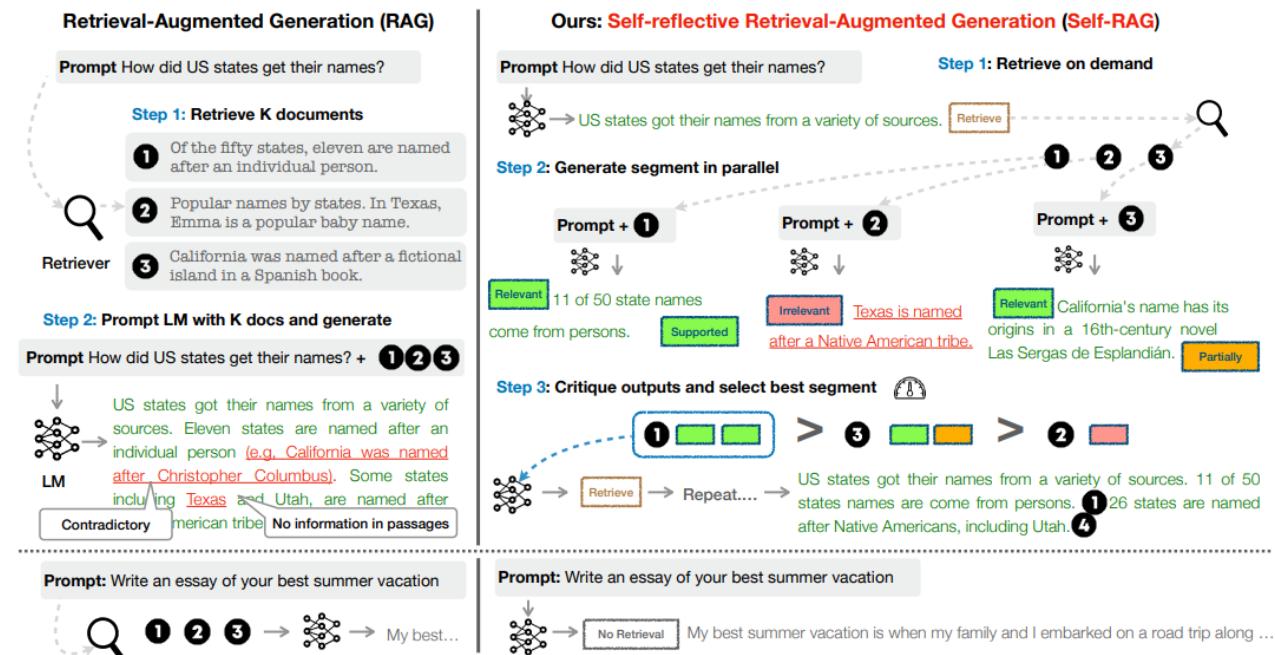


FLARE_{direct} Trong phương pháp *FLARE_{instruct}*, có lúc các câu truy vấn được tạo ra không chính xác, làm giảm độ tin cậy của quá trình truy vấn. Để giải quyết vấn đề này, bài báo đề xuất một cách tiếp cận truy vấn chủ động khác, dựa vào việc xem xét câu tiếp theo để quyết định thời điểm truy vấn. Cụ thể, mô hình ngôn ngữ sẽ trước tiên tạo ra một câu tiếp theo mà không dựa vào kết quả truy vấn. Nếu mô hình tự tin về câu trả lời đó, câu trả lời sẽ được chấp nhận mà không cần truy vấn thêm. Trong trường hợp mô hình không tự tin, một câu truy vấn sẽ được tạo dựa trên câu tiếp theo để tìm kiếm thông tin cần thiết, và sau đó câu trả lời sẽ được tạo ra lại. Cách tiếp cận này hiệu quả bởi vì thường xuyên từ có độ tự tin thấp thì liên quan đến việc thiếu thông tin.

9.2 Self RAG: Learning to retrieve, generate, and critique through self-reflection

<https://arxiv.org/pdf/2310.11511.pdf> <https://www.youtube.com/watch?v=i4V9iJcxzZ4>

Bài báo "Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection" giới thiệu một framework mới có tên là Self-Reflective Retrieval-Augmented Generation (Self-RAG). Framework này nhằm mục đích cải thiện chất lượng và độ chính xác về thông tin của các mô hình ngôn ngữ lớn (LLMs) bằng cách cho phép chúng tự động tìm kiếm thông tin và phản ánh về nội dung được trích xuất cũng như phản hồi của chính mình. Khác với các mô hình RAG truyền thống có thể trích xuất thông tin không liên quan, Self-RAG sử dụng các mã token đặc biệt gọi là token phản ánh để làm cho mô hình có thể kiểm soát và thích nghi với nhiều nhiệm vụ khác nhau. Các tác giả chứng minh rằng Self-RAG vượt trội hơn các mô hình LLM hiện tại và các mô hình tăng cường tìm kiếm trong các nhiệm vụ như QA miền mở, suy luận, xác minh sự thật và tạo ra văn bản dài, làm nổi bật những cải thiện đáng kể về tính chính xác và độ chính xác của trích dẫn.



Trong nghiên cứu này, các tác giả huấn luyện mô hình LLM để tự sinh ra 4 loại token phải ứng (reflection token) để hỗ trợ cho quá trình trả lời, bao gồm:

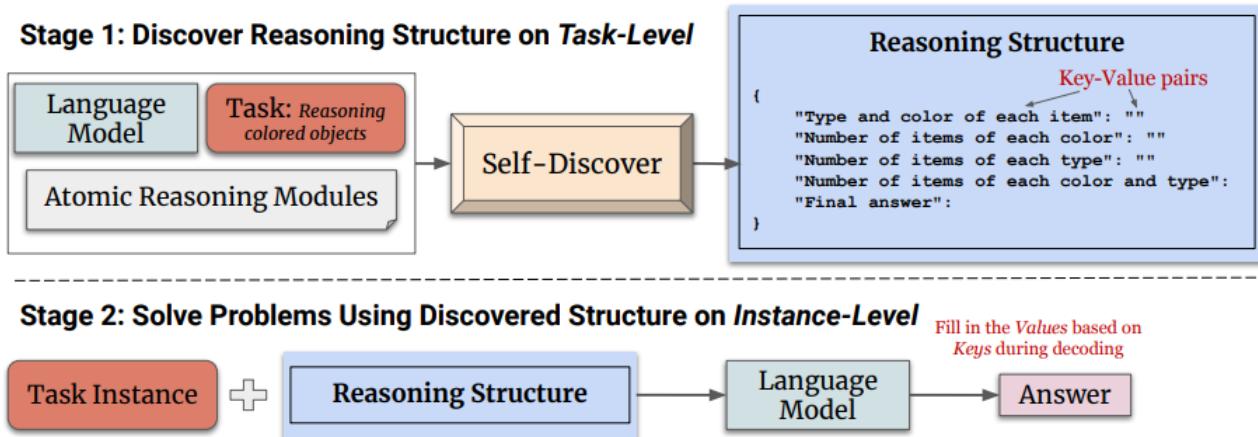
- Retrieve: Cho mô hình biết có nên truy vấn/tiếp tục truy vấn hay không.
- IsRel: Dánh giá xem kết quả truy vấn có cung cấp thông tin hữu dụng cho quá trình trả lời câu hỏi hay không.
- IsSup: Cho biết mức độ kết quả truy vấn cung cấp dẩn chứng hỗ trợ cho câu trả, bao gồm: support hoàn toàn, support một phần, và hoàn toàn không support cho câu trả lời.
- IsUse: Dánh giá xem mức độ hữu dụng của câu trả lời, trên thang điểm từ 1-5.

Ở quá trình inference, thuật toán Self-RAG hoạt động như sau:

- Đầu tiên ta có một câu hỏi prompt từ người dùng và câu trả lời từ bước trước đó (nếu có). Model sẽ tự tạo ra token Retrieve để dự đoán xem có cần truy vấn thông tin từ cơ sở dữ liệu hay không.
- Nếu cần Retrieve thì:
 - Truy vấn thông tin cần thiết từ cơ sở dữ liệu.
 - Mô hình dự đoán token IsRel để dự đoán kết quả truy vấn có cung cấp thông tin cần thiết.
 - Mô hình dự đoán token IsSup và IsUse để đánh giá từng kết quả truy vấn.
 - Rerank lại kết quả truy vấn dựa trên 3 token IsRel, IsSup và IsUse.
- Nếu không cần Retrieve, mô hình sẽ trả lời mà không cần retrieve, sau đó dự đoán độ hữu ích của câu trả lời qua token IsUse.

9.3 SELF-DISCOVER: Large Language Models Self-Compose Reasoning Structures

<https://arxiv.org/pdf/2402.03620.pdf> 52 y Trong các nghiên cứu, các phương pháp prompting khác nhau thường được đề xuất để giải quyết một vấn đề hết sức cụ thể, chẳng hạn như giải toán, suy luận theo từng bước, hay suy luận số học. Tuy nhiên các phương pháp prompting này lại không thật sự phù hợp với tất cả các task, chẳng hạn những phương pháp prompting chuyên về giải toán sẽ không tốt bằng những phương pháp khác khi được ứng dụng trong tác vụ trả lời câu hỏi suy luận logic. Vì vậy nghiên cứu "SELF-DISCOVER: Large Language Models Self-Compose Reasoning Structures" Zhou et al., 2024 nhằm đến tự tìm kiếm những cấu trúc suy luận phù hợp nhất cho từng task, trong khi vẫn giữ tính hiệu quả về mặt tính toán.



Phương pháp này được lấy cảm hứng từ cách con người chúng ta suy luận một vấn đề. Phương pháp này được chia thành 2 giai đoạn. Ở giai đoạn đầu tiên, phương pháp self-discover sẽ tìm ra những cấu trúc prompt suy luận phù hợp nhất cho từng task. Sau đó các cấu trúc này sẽ được format về dạng JSON, bởi cấu trúc này được chứng minh là tăng cường khả năng suy luận của mô hình Zhou et al., 2023. Bước này có mục tiêu là tìm những cấu trúc suy luận hợp lý nhất cho từng tác vụ, và không chỉ một tác vụ trong một lần. Ở giai đoạn thứ 2, phương pháp này sử dụng những cấu trúc suy luận ở bước trước và tìm ra cấu trúc phù hợp nhất với mỗi câu hỏi của người dùng.

Giai 1 bao gồm 3 hành động chính:

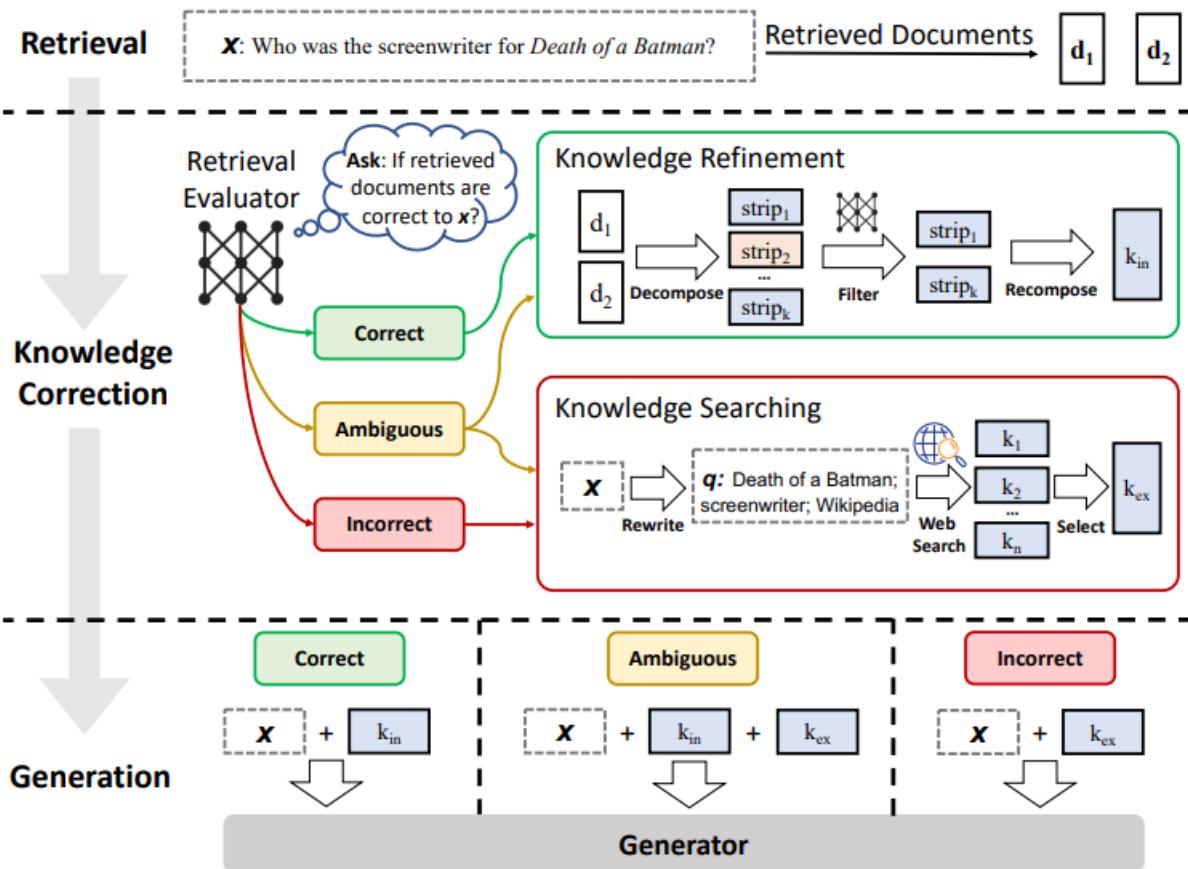
- SELECT: Những cấu trúc suy luận có thể cần thiết để giải quyết tác vụ được chọn từ một tập mô tả các cấu trúc suy luận.
- ADAPT: Cấu trúc suy luận ở bước trước sẽ được viết lại để phù hợp hơn với tác vụ mà mô hình đang muốn giải quyết.
- IMPLEMENT: cấu trúc vừa được tạo ra sẽ được tiến hành thành một cấu trúc từng bước các hành động để giải quyết tác vụ.

Sau 3 bước ở giai đoạn 1, ta đã tìm được cấu trúc phù hợp để giải 1 tác vụ cụ thể. Sau đó ta sẽ prompt mô hình ngôn ngữ lớn để làm theo cấu trúc đó, từ đó tạo ra câu trả lời chính xác cho người dùng.

9.4 Corrective RAG

<https://arxiv.org/abs/2401.15884> <https://www.youtube.com/watch?v=pbAd8O1Lvm4>

Trong các ứng dụng RAG thông thường, thông tin ngữ cảnh truy vấn sẽ được đưa vào mô hình để sinh ra kết quả, bất kể rằng thông tin đó có liên quan hay không. Điều này làm cho kết quả sinh ra bởi mô hình trở nên nhiễu và làm chậm tốc độ tạo sinh đi rất nhiều. Vì vậy nghiên cứu "Corrective Retrieval Augmented Generation" Yan et al., 2024 được đưa ra nhằm giải quyết vấn đề này. Bài báo này đi vào nghiên cứu những trường hợp khi những thông tin truy vấn được không liên quan hoặc không chính xác, đồng thời đề xuất 1 cơ chế tự động điều chỉnh và tối ưu những văn bản truy vấn.



Cụ thể, phương pháp này hoạt động như sau. Cho một câu query và nhiều văn bản được truy vấn. Một mô hình đánh giá gọn nhẹ sẽ được sử dụng để đánh giá độ phù hợp của các văn bản với câu query đầu vào, quy thành 3 mức độ: Đúng, Không đúng và không rõ ràng, tương ứng với những hành động khác nhau. Nếu những văn bản truy vấn được là Đúng, những văn bản đó sẽ được chắt lọc lại thành những thông tin liên quan. Quá trình này bao gồm: phân rã kiến thức, chắt lọc và sắp xếp lại kiến thức. Nếu những văn bản truy vấn được là không chính xác, những văn bản này sẽ được bỏ đi, thay vào đó là bước tìm kiếm trên internet. Khi những thông tin truy vấn là không rõ ràng, cả 2 quá trình chắt lọc và tìm kiếm trên web sẽ đồng thời được diễn ra.

Một trong những thành phần quan trọng nhất trong phương pháp này là mô hình đánh giá văn bản truy vấn. Trong nghiên cứu này, mô hình T5-Large được sử dụng để fine-tune cho tác vụ này. Với mỗi câu query, sẽ có 10 văn bản được truy vấn. Từng câu sẽ kết hợp với câu query để thành input đầu vào đưa vào mô hình T5, từ đó đưa ra một thang điểm độ liên quan. Dựa trên thang điểm này, ta sẽ phân loại văn bản đầu vào thành 3 loại: Đúng, Không Đúng, hoặc không liên quan.

Khi những văn bản truy vấn được là Đúng, quá trình chắt lọc kiến thức sẽ được bắt đầu. Đầu tiên, những văn bản này sẽ được cắt ra thành những đoạn kiến thức. Lúc này mô hình đánh giá truy vấn ở bước trên sẽ được sử dụng lại độ liên quan của những dải kiến thức này, và những dải không quá liên

quan sẽ được loại bỏ đi, trong khi những dài có liên quan sẽ được giữ lại.

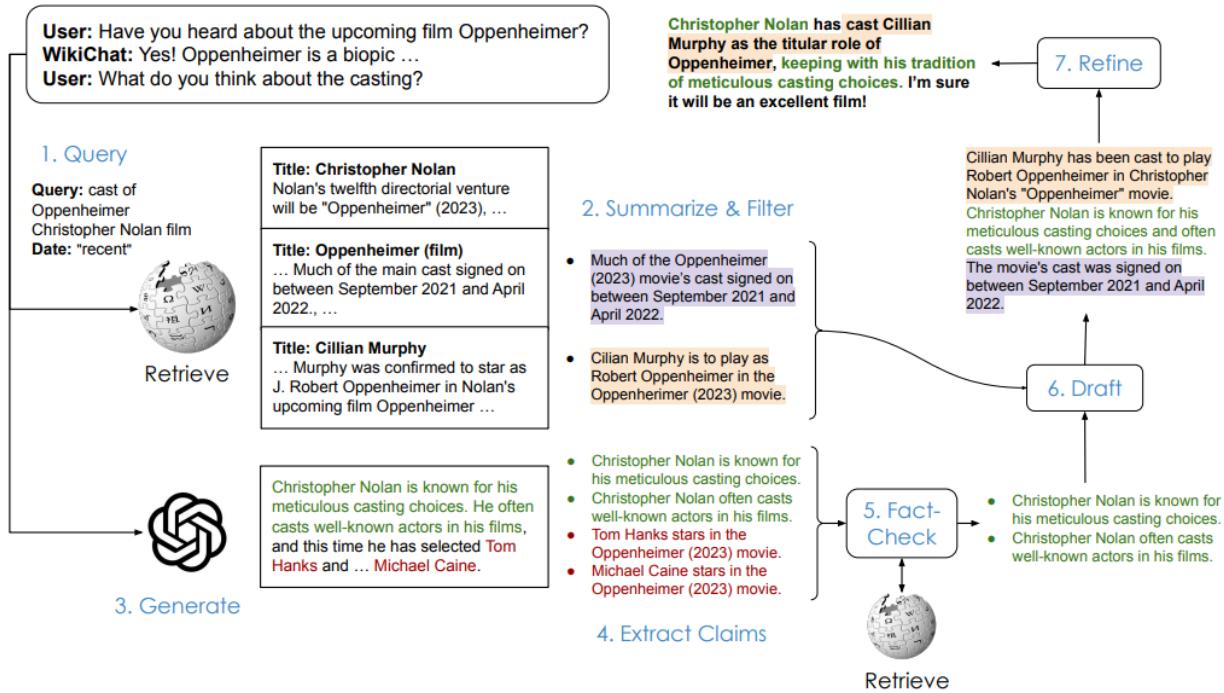
Khi những văn bản truy vấn là Không Đúng, quá trình tìm kiếm trên web sẽ được bắt đầu. Cụ thể câu đầu vào sẽ được viết lại thành những keyword để mô phỏng quá trình tìm kiếm trên web. Sau đó quá trình chắt lọc thông tin tương tự như bước trên sẽ được sử dụng để đưa ra những dài kiến thức liên quan nhất.

9.5 WikiChat

<https://arxiv.org/pdf/2305.14292v2.pdf> Đây là một phương pháp được nghiên cứu trong paper "WikiChat: Stopping the Hallucination of Large Language Model Chatbots by Few-Shot Grounding on Wikipedia" Semnani et al., 2023, nhằm giải quyết vấn đề cần câu trả lời đưa ra bởi mô hình ngôn ngữ lớn có độ trung thực không cao và có tỉ lệ tưởng tượng (hallucination) lớn. Phương pháp này bao gồm 7 bước chính, với mỗi bước sử dụng few-shot prompting để hướng mô hình trả lời theo ý muốn của người dùng. Nhìn tổng quan, WikiChat bao gồm 2 giai đoạn chính, giai đoạn đầu là thu thập thông tin dựa trên đoạn chat với người dùng, và giai đoạn hai đưa ra câu trả lời dựa trên thông tin thu thập được. 5 bước đầu của phương pháp này thuộc về giai đoạn 1, và 2 bước sau thuộc về giai đoạn 2, sau đây chúng ta đi chi tiết từng giai đoạn.

Giai đoạn 1: Thu thập thông tin Trong quá trình giao tiếp với người dùng, WikiChat sẽ tự nhận biết lúc nào nên thu thập thông tin, (khi người dùng đưa ra câu hỏi trực tiếp). Giai đoạn này bao gồm 5 bước chính

- Bước 1: WikiChat sẽ tự tạo một câu query có thể bao hàm được ý muốn của người dùng, cộng thêm một mốc thời gian chẳng hạn như "recent" nếu câu trả lời cần cập nhật nhất có thể, "year=xxxx" cho một năm cụ thể, hoặc là none nếu thời gian là không quan trọng.
- Bước 2: WikiChat trích xuất những thông tin liên quan từ những văn bản truy vấn được thành dạng những gạch đầu dòng tóm tắt, và những thông tin không liên quan sẽ được lọc đi.
- Bước 3: Mô hình LLM sẽ được sử dụng để tạo ra câu trả lời, câu trả lời này thường sẽ đề cập những thông tin liên quan và thú vị, tuy nhiên thường không đáng tin cậy.
- Bước 4: Câu trả lời của mô hình sẽ được chia nhỏ ra thành từng câu khẳng định một. Sau đó một hệ thống truy vấn thông tin sẽ được sử dụng để truy vấn những bằng chứng cho khẳng định đó.
- Bước 5: Sử dụng một mô hình LLM khác, từng câu khẳng định sẽ được chia thành 3 nhóm chính, những bằng chứng truy vấn được ủng hộ khẳng định, phủ định, hoặc không đủ thông tin để kết luận. Chỉ những khẳng định được ủng hộ bởi bằng chứng mới được giữ lại.



Giai đoạn 2: Tạo ra câu trả lời Bước tiếp theo là dùng những thông tin đã thu thập được tạo ra câu trả lời chính xác. Nghiên cứu này chỉ ra rằng việc trực tiếp tạo ra câu trả lời sẽ gây khó khăn trong việc duy trì tính hội thoại cho câu trả lời của mô hình. Vì vậy, giai đoạn này sẽ chia thành 2 bước nhỏ.

- Bước 6: WikiChat tạo ra một câu trả lời tạm dựa trên những gạch đầu dòng được truy vấn từ bước trước.
- Bước 7: Sau đó WikiChat sẽ tạo ra nhận xét và chỉnh sửa lại câu trả lời dựa trên các yếu tố tính phù hợp, tính tự nhiên, tính không lặp lại và tính đúng đắn về mặt thời gian.

Chắt lọc kiến thức sáng các mô hình nhỏ hơn Để giảm độ trễ, chi phí và đảm bảo tính riêng tư, nghiên cứu này đề xuất phương pháp để chắt lọc kiến thức từ một mô hình lớn và mã nguồn đóng (chẳng hạn như GPT-4) sang một mô hình nhỏ hơn và mã nguồn mở (chẳng hạn như Llama-7B). Một mô hình ngôn ngữ giả lập người dùng sẽ được sử dụng để hội thoại với WikiChat về những chủ đề được lấy từ Wikipedia, và ghi lại những điều vào và điều ra của mô hình. Dữ liệu này sẽ được giữ lại để fine-tune mô hình Llama nhỏ. Kết quả thu được cho thấy mô hình nhỏ này có khả năng trả lời tiệm cận so với mô hình lớn, trong khi giảm đi độ trễ, chi phí và vẫn đảm bảo được tính riêng tư cho dữ liệu người dùng.

References

- 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.
- Gao, L., Ma, X., Lin, J., & Callan, J. (2022). Precise zero-shot dense retrieval without relevance labels.
- Liu, N. F., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., Petroni, F., & Liang, P. (2023). Lost in the middle: How language models use long contexts.

- Kamradt, G. (2024). *The 5 levels of text splitting for retrieval*. <https://youtu.be/8OJC21T2SL4>
- Chen, T., Wang, H., Chen, S., Yu, W., Ma, K., Zhao, X., Zhang, H., & Yu, D. (2023). Dense x retrieval: What retrieval granularity should we use?
- Wang, Z., Zhang, H., Li, C.-L., Eisenschlos, J. M., Perot, V., Wang, Z., Miculicich, L., Fujii, Y., Shang, J., Lee, C.-Y., & Pfister, T. (2024). Chain-of-table: Evolving tables in the reasoning chain for table understanding.
- Muennighoff, N., Tazi, N., Magne, L., & Reimers, N. (2023). Mteb: Massive text embedding benchmark.
- Kojima, T., Gu, S. S., Reid, M., Matsuo, Y., & Iwasawa, Y. (2023). Large language models are zero-shot reasoners.
- Jiang, Z., Xu, F. F., Gao, L., Sun, Z., Liu, Q., Dwivedi-Yu, J., Yang, Y., Callan, J., & Neubig, G. (2023). Active retrieval augmented generation.
- Schick, T., Dwivedi-Yu, J., Dessi, R., Raileanu, R., Lomeli, M., Zettlemoyer, L., Cancedda, N., & Scialom, T. Toolformer: Language models can teach themselves to use tools. In: 2023.
- Zhou, P., Pujara, J., Ren, X., Chen, X., Cheng, H.-T., Le, Q. V., Chi, E. H., Zhou, D., Mishra, S., & Zheng, H. S. (2024). Self-discover: Large language models self-compose reasoning structures.
- Zhou, Y., Muresanu, A. I., Han, Z., Paster, K., Pitis, S., Chan, H., & Ba, J. (2023). Large language models are human-level prompt engineers.
- Yan, S.-Q., Gu, J.-C., Zhu, Y., & Ling, Z.-H. (2024). Corrective retrieval augmented generation.
- Semnani, S. J., Yao, V. Z., Zhang, H. C., & Lam, M. S. (2023). Wikichat: Stopping the hallucination of large language model chatbots by few-shot grounding on wikipedia.

- *Hết* -