

Advanced RAG

<https://atekco.io/1713861441749-toan-canhan-cac-ky-thuat-advanced-rag/>

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

<https://medium.com/@krtarunsingh/advanced-rag-techniques-unlocking-the-next-level-040c205b95bc>

<https://pub.towardsai.net/advanced-rag-techniques-an-illustrated-overview-04d193d8fec6>

<https://towardsdatascience.com/advanced-retrieval-augmented-generation-from-theory-to-llmindex-implementation-4de1464a9930>

<https://arxiv.org/pdf/2404.01037>

<https://github.com/ianhojy/auto-hyde/tree/main>

<https://medium.com/m/global-identity-2?redirectTo=https%3A%2F%2Ftowardsdatascience.com%2Fautohyde-making-hyde-better-for-advanced-llm-rag-619e58cdbd8e>

<https://medium.com/m/global-identity-2?redirectTo=https%3A%2F%2Ftowardsdatascience.com%2Fautohyde-making-hyde-better-for-advanced-llm-rag-619e58cdbd8e>

I. Giới thiệu về RAG

1. Tổng quan về RAG

- Những hạn chế của LLMs: Các LLM chỉ có thể trả lời tốt ở các câu hỏi mà chúng ra được học trước đó. Còn đối với những câu hỏi mới (dữ liệu mới) chưa được học thì nó sẽ bị vấn đề là **hallucination**.
- Retrieval-Augmented Generation (RAG) là một kỹ thuật giúp nâng cao khả năng của mô hình sinh (language model generation) kết hợp với tri thức bên ngoài (external knowledge)
- Phương pháp này thực hiện bằng cách truy xuất thông tin liên quan từ kho tài liệu (tri thức) và sử dụng chúng cho quá trình sinh câu trả lời dựa trên LLMs.

2. Lợi ích mà RAG mang lại

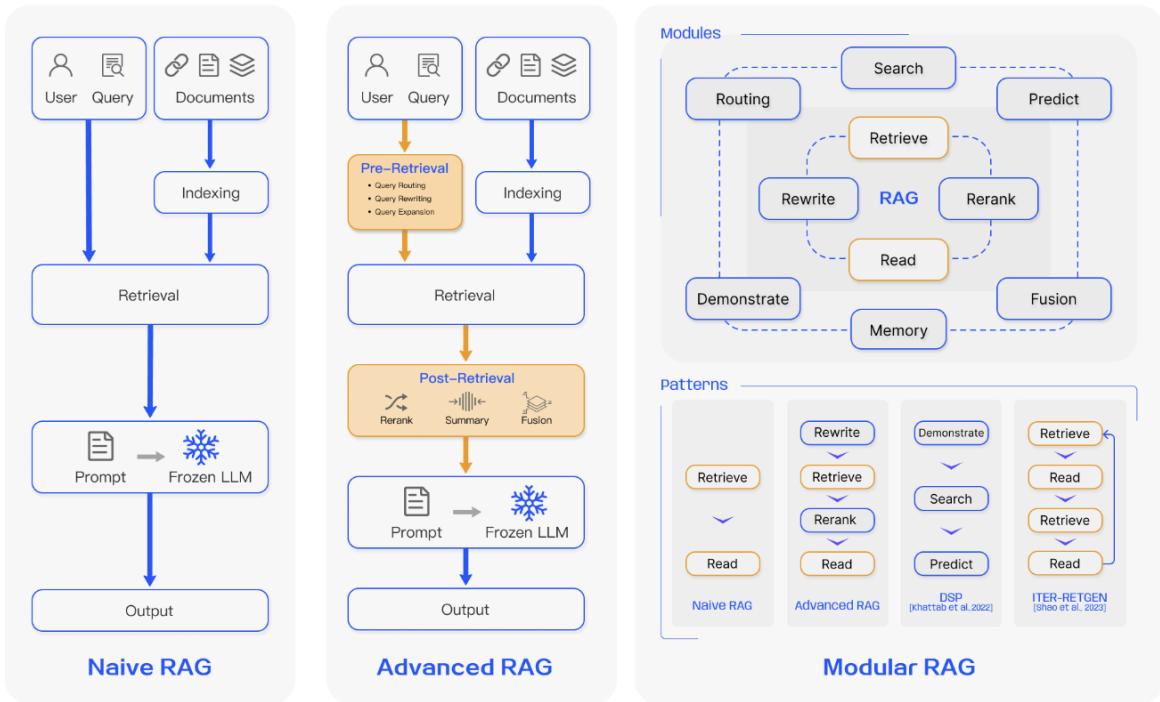
- Giảm thiểu chi phí
- Tiện lợi cho việc cung cấp các thông tin, dữ liệu mới cho Chatbot.
- Phù hợp với một domain cụ thể của doanh nghiệp.

3. Hạn chế của RAG

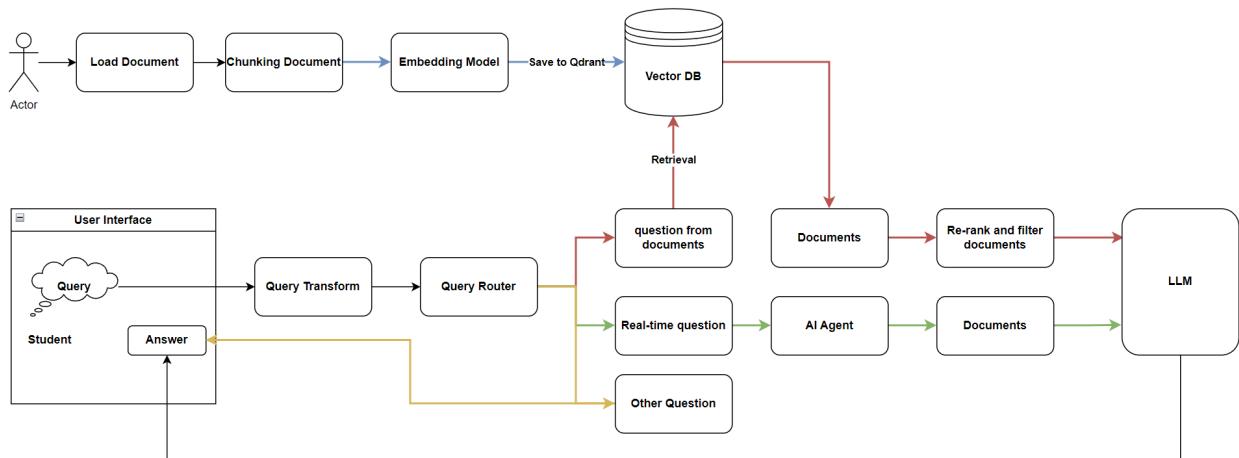
- Một số trường hợp vẫn chưa trả lời bằng với việc fine-tuning
- Tốc độ trả lời chậm hơn do phải qua nhiều bước xử lý.

II. Hạn chế của Basic RAG

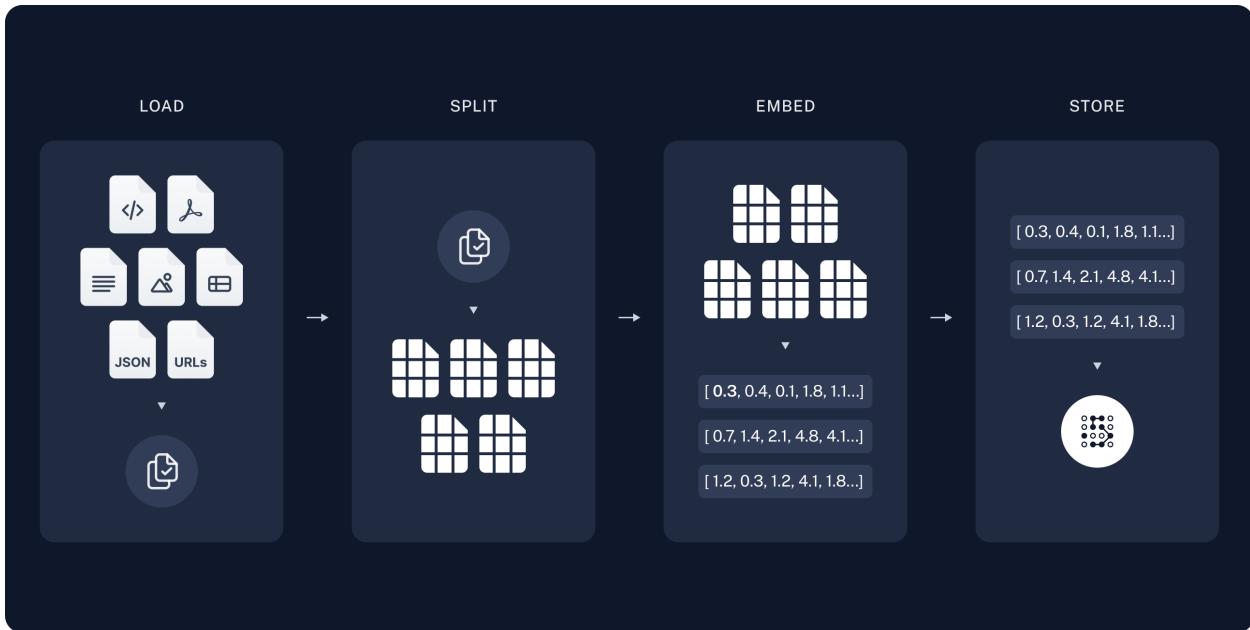
- Truy vấn: Thiếu độ chính xác trong việc lựa chọn được các thông tin liên quan đến câu hỏi.
- Phản hồi người dùng: Nếu truy vấn không tốt nguồn dữ liệu có thể bị sai khác dẫn đến câu trả lời không có chất lượng cao



III. Advanced RAG



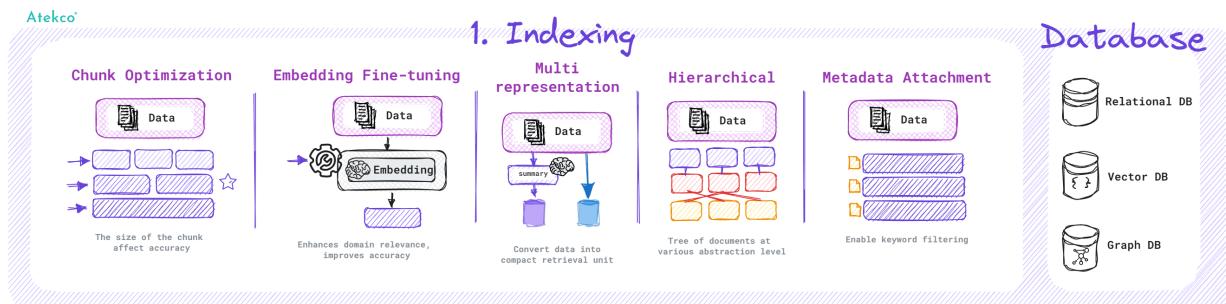
1. Indexing



1.1 Load Document

- Lấy nội dung Document từ các dữ liệu phi cấu trúc như file PDF, Docx, Latex, Markdown.

1.2 Indexing



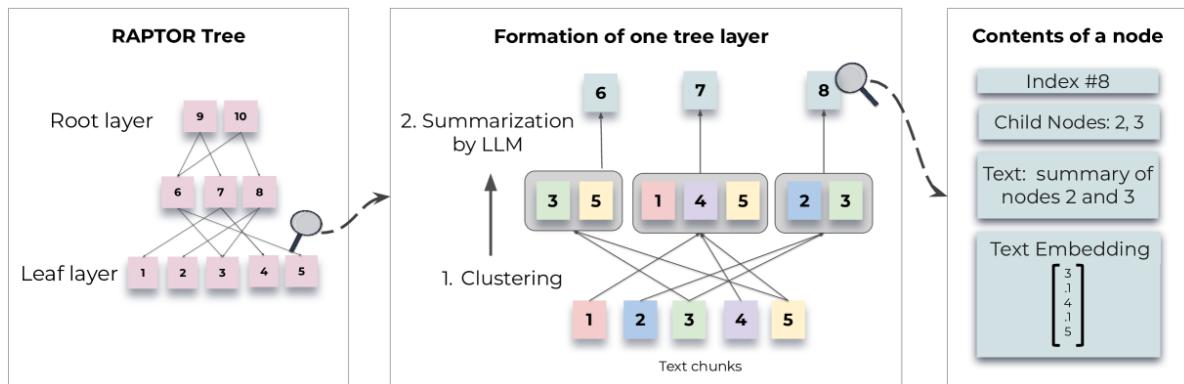
- Chunk Optimization:** Chia dữ liệu thành các chunk. Giúp tối ưu kích thước và nội dung của mỗi chunk để đảm bảo rằng duy trì được ngữ cảnh mà không vượt quá độ dài của LLM quy định.

Chunk Optimization

- Multi-Representation:** Tóm tắt lại nội dung của Document. Giúp cải thiện tốc độ và tăng độ chính xác (chỉ phù hợp với yêu cầu

tóm tắt).

- **Hierarchical Indexing:** Áp dụng mô hình phân cấp RAPTOR để tổ chức dữ liệu thành các cấp độ tổng hợp khác nhau. Nó sẽ tăng ngữ cảnh cho câu được truy vấn từ việc lấy thêm thông tin từ nút lá.



- **Metadata Attachment:** Thêm metadata vào cho từng chunk giúp tăng khả năng phân loại thông tin, cho phép truy xuất thông tin một cách dễ dàng.

1.3 Embedding model

- Sentence-BERT (SBERT), một biến thể của mạng BERT được sử dụng để tạo ra các nhúng câu có ý nghĩa ngữ nghĩa. Các nhiệm vụ của SBERT làm tốt hơn BERT bao gồm so sánh tương đồng ngôn ngữ nghĩa trên quy mô lớn, phân cụm, và truy xuất thông tin thông qua tìm kiếm ngữ nghĩa.
- Model:
 - Mono: <https://huggingface.co/bkai-foundation-models/vietnamese-bi-encoder>
 - Multilingual: paraphrase-multilingual-mpnet-base-v2

1.4 Vector Store

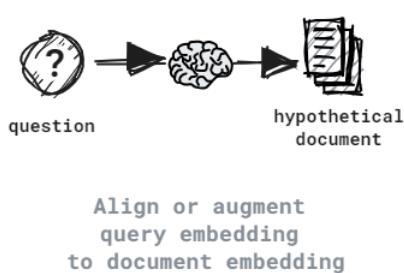
DB Attributes		Open-source & free to self-host	Managed Cloud Offering	Disk-based Index	Multi-tenancy Support	In-built Text Embeddings creation (Bring-your-own-model)	In-built Image Embedding creation	Metadata Filtering	Embeddable	Multiple vectors per point	Langchain integration	Llama index integration	Hybrid Search	BM25 support	Sparse Vector	Full-text search
1 Pinecone		✗	✓		✓ via I	✗		✓	✗	✓	✓	✓	✗	✗	✓	✗
2 Qdrant		✓	✓		✓ via C	✓ via FastEml	✓	✗	✓	✓	✓	✓	✗	✗	✗	✗
3 Weaviate		✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓ RRF	✓	✗	✗
4 pgvector		✓	✓	(supabas	✓	✗		✓	✗	✓	✓	✓	✓ https	✓	✗	✓ ht
5 Vespa		✓	✓	✓	https://doc	✓	✗	✓	✗	✓	✓	✓	✓	✓	✓	✓
6 Milvus		✓	✓	✓	✓ http	✗		✓	✗	✓	✓	✓	✓	✗	✗	✗
7 MongoDB Atlas		✗	✓		✓ via I	✗		✓	✗	✓	✓	✓	✓ https	✓	✗	✓
8 Marqo		✓	✓			✓	✓	✓	✓	✓	✓	✓	✓ via w	✓	✗	✗
9 Vectara		✗	✓		✓ via I	✗ [Note Vecta	✓	✗	✓	✓	✓	✓	✓ Only	✗	✗	✗
10 Elasticsearch		✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
11 OpenSearch		✓	✓	✓	✓	✓	✓ ht	✓	✗	✓	✓	✓	✓ Only	✓	✗	✓
12 Chroma		✓	✗	✗	✗	✓		✓	✓	✗	✓	✓	✗	✗	✗	✗

2. Query Transformation & Query Routing

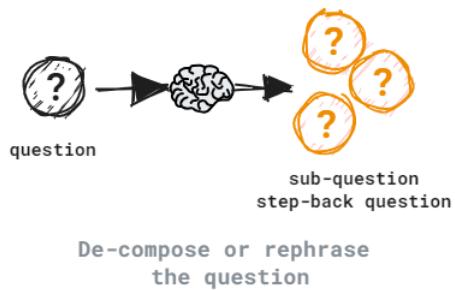
2.

Query Transformation

HyDE

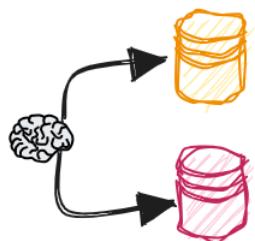


Multi-Step Query

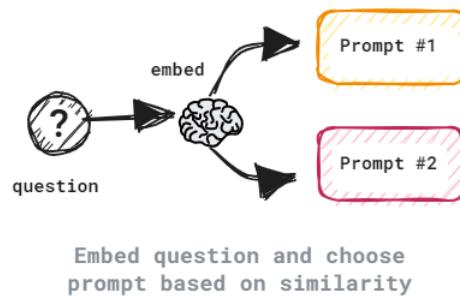


Query Routing

Logical Routing



Semantic Routing



2.1 Query Transformation

Định nghĩa: Query Transformation là kỹ thuật sử dụng LLM để lý luận giúp chỉnh sửa đầu vào của người dùng nhằm cải thiện khả năng truy xuất thông tin. LLM có thể chuyển đổi câu hỏi ban đầu thành câu hỏi rõ ràng hơn giúp tăng khả năng truy vấn.

Một số kỹ thuật Query Transformation:

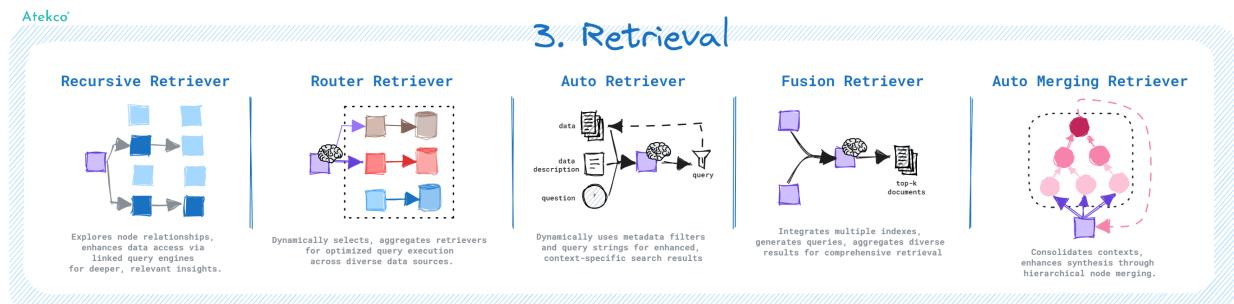
- **HyDE:** Là một kỹ thuật sử dụng LLM để sinh ra câu trả lời giả định dựa vào câu hỏi/câu query. Sau đó dựa vào vector của dữ liệu giả định vừa sinh ra + vector của câu query để truy xuất thông tin. Kỹ thuật này giúp nâng cao sự tương đồng về ngữ nghĩa giữa câu hỏi và nội dung được truy xuất ra từ DB.
- **Multi-Step Query:** Là phương pháp phân rã câu hỏi phức tạp thành các câu hỏi con đơn giản và thực hiện việc truy xuất dựa trên các câu hỏi con và kết quả của các query con sẽ được kết hợp.

2.2 Query Routing

Định nghĩa: Query Routing là bước sử dụng LLM để điều hướng truy vấn của người dùng. Các lựa chọn có thể là thực hiện truy vấn trên một tập dữ liệu cụ thể (giả sử RAG gồm nhiều loại dữ liệu khác nhau như địa lý, lịch sử, toán học). Hoặc có thể truy vấn trên nhiều tập khác nhau sau đó kết hợp lại để có kết quả cuối cùng

- **Logical Routing:** Sử dụng kỹ thuật Logic để định hướng truy vấn đến nguồn dữ liệu phù hợp. Bằng cách phân tích cấu trúc và mục đích của câu hỏi, router lựa chọn index nguồn dữ liệu phù hợp để thực hiện việc truy xuất.
- **Semantic Routing:** Sử dụng ngữ nghĩa của câu hỏi để định hướng. Router phân tích ngữ nghĩa của câu hỏi để trỏ tới index phù hợp

3. Retrieval



Định nghĩa: Quá trình Retrieval tập trung vào việc lấy dữ liệu từ các nguồn khác nhau để cung cấp ngữ cảnh.

<https://www.notion.so/Information-Retrieval-997b8f6431d44466b92e1b0b91b8eb71>

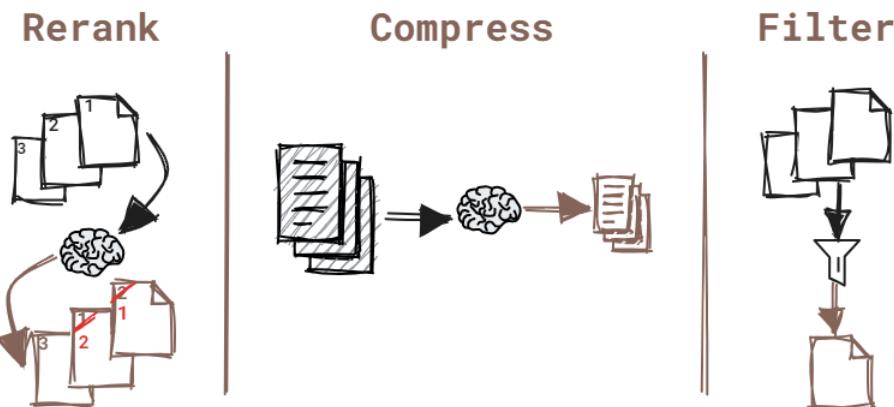
Một số kỹ thuật Retrieval:

- **Recursive Retriever:** là kỹ thuật cho phép truy xuất sâu vào dữ liệu liên quan và thực hiện việc truy xuất thêm dữ liệu dựa vào kết quả truy xuất trước đó. Kỹ thuật này hữu ích trong các tác vụ muốn khám phá sâu vào thông tin.
- **Fusion Retrieval:** Kết hợp kết quả từ nhiều truy vấn và index khác nhau, giúp tối ưu hóa việc truy xuất thông tin và đảm bảo kết quả thu được là toàn diện và không bị trùng lặp, mang lại cái nhìn đa chiều cho truy xuất.
- **Auto Merging Retrieval:** Khi sử dụng nhiều câu query con (Kỹ thuật Multi-step query) để truy vấn, kỹ thuật này sẽ chuyển chúng thành phân đoạn dữ liệu cha, tổng hợp ngữ cảnh nhỏ lẻ thành một ngữ cảnh lớn, hỗ trợ quá trình tổng hợp thông tin

4. Post Retrieval

Định nghĩa: Nó sẽ là nơi đánh giá và tinh chỉnh thông qua bộ lọc, sắp xếp lại hoặc biến đổi dữ liệu sau khi retrieval trước khi đưa vào mô hình. Mục đích nhằm nâng cao chất lượng của context trước khi đưa vào mô hình LLM.

4. Post-Retrieval



Rank / Filter / Compress documents based on relevance

Một số phương pháp:

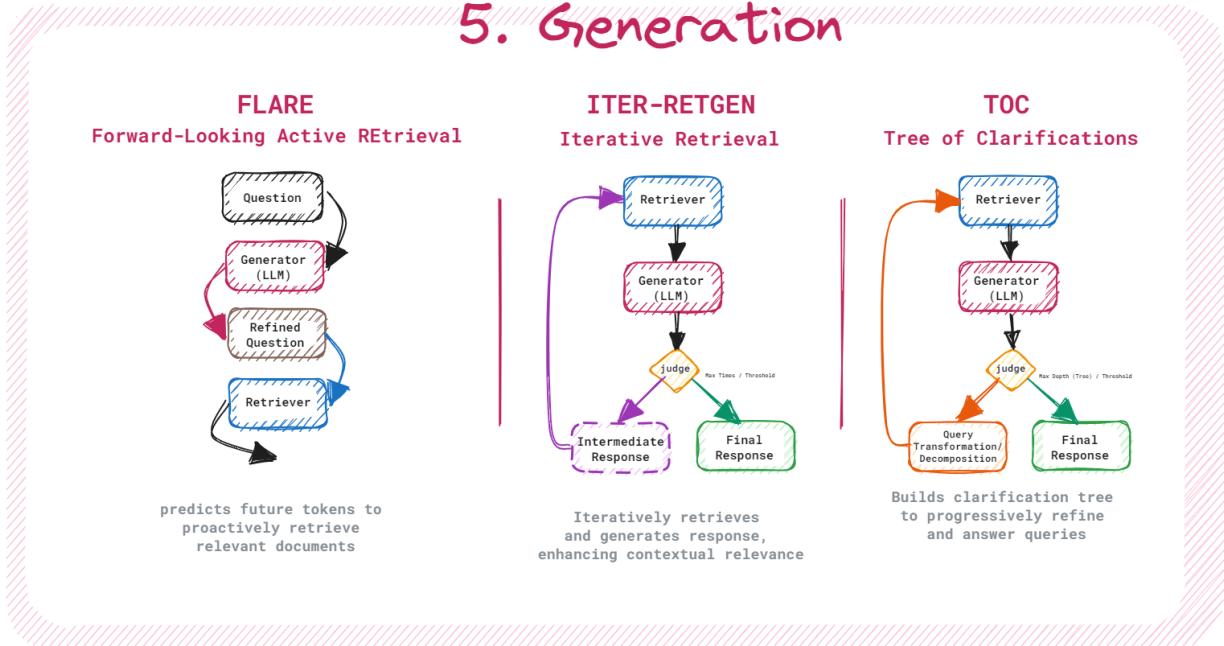
- **Rerank:** Không chỉ sắp xếp lại văn bản mà còn giúp lọc văn bản. Link đọc thêm: <https://www.notion.so/Reranking-Method-2b42a1c01fba4b64ba49cbe8c30edc47>.
- **Compress:** Giảm bớt ngữ cảnh dư thừa và không cần thiết, loại bỏ nhiễu.
- **Filter:** Chọn lọc nội dung trước khi đưa vào LLM, loại bỏ những tài liệu hoặc thông tin không liên quan hoặc có độ chính xác thấp.

5. Generation

Định nghĩa: Generation là quá trình LLM sinh ra câu trả lời dựa vào context và quá trình truy xuất đưa ra.

Một số phương pháp tăng chất lượng cho kết quả Generation:

5. Generation



- **FLARE:** Phương pháp này dựa vào kỹ thuật Prompt để đưa ra câu trả lời mà không dựa vào context được query. Context chỉ được đưa vào khi thông tin bị thiếu. Nhằm tránh việc thu thập dữ liệu không cần thiết. Quá trình này liên tục điều chỉnh câu hỏi và kiểm tra từ khoá có xác suất xuất hiện thấp thì từ đó sẽ truy xuất thêm dữ liệu liên quan để cải thiện và tinh chỉnh lại câu trả lời.
- **ITER-RETRIEVE:** Phương pháp này sẽ lặp lại quá trình generation dựa vào thông tin đã truy xuất. Mỗi lần lặp sẽ sử dụng kết quả làm ngữ cảnh để truy xuất nhằm nâng cao chất lượng hơn. Nó chỉ dừng lại khi max_time hoặc đến 1 ngưỡng chúng ta quy định.
- **ToC (Tree of Classification):**
 - ToC là phương pháp sử dụng đệ quy để làm rõ hơn câu hỏi ban đầu. Trong quá trình này mỗi bước hỏi đáp sẽ được đánh giá. Nếu câu trả lời chưa vượt ngưỡng thì sẽ sinh ra câu hỏi mới để làm rõ sự mơ hồ giúp tăng khả năng truy vấn và cho kết quả generation tốt hơn.
 - Quy trình đánh giá dựa trên ngưỡng hoặc thời gian.

- Việc làm rõ câu trả lời sẽ sử dụng phương pháp: viết lại câu hỏi hoặc là sẽ phân rã câu hỏi nhỏ hơn.