
**Tiểu luận cuối kì Data Mining**

# NỘI DUNG: Xây dựng Agent Ai và chatbot AI sử dụng RAG

**Giảng viên:** Lê Hoàng Sơn

**Sinh viên thực hiện:** Hoàng Việt Hưng

**Mã Sinh Viên:** 20002057

**Hà Nội, 2025**

# Mục lục

# 1: Giới thiệu đề tài

Trong những năm gần đây, sự phát triển mạnh mẽ của các mô hình ngôn ngữ lớn (Large Language Models – LLMs) như GPT, Llama hay Claude đã tạo ra bước đột phá trong nhiều lĩnh vực như xử lý ngôn ngữ tự nhiên, phân tích dữ liệu, trợ lý ảo và tự động hóa quy trình. Mặc dù có khả năng sinh ngôn ngữ mạnh mẽ, các mô hình LLM truyền thống vẫn tồn tại nhiều hạn chế, bao gồm hiện tượng *hallucination*, không thể tự truy cập hoặc cập nhật dữ liệu mới, và khả năng suy luận không ổn định khi thiếu thông tin ngữ cảnh.

Retrieval-Augmented Generation (RAG) xuất hiện như một giải pháp hiệu quả nhằm khắc phục các nhược điểm này bằng cách kết hợp cơ chế truy xuất tài liệu (*retrieval*) với mô hình sinh (*generation*). Nhờ khả năng sử dụng dữ liệu ngoài mô hình, RAG giúp tạo ra câu trả lời chính xác hơn, cập nhật theo thông tin thực tế và giảm thiểu sai lệch.

Bên cạnh RAG truyền thống, nhiều biến thể nâng cao như Self-RAG, Reflexion Agentic RAG, RAG với BM25 Re-ranking hay Adaptive RAG Routing cũng được phát triển nhằm xây dựng các AI Agent thông minh, có khả năng tự phản hồi, tự sửa lỗi, tự truy xuất và tự điều hướng nhiệm vụ. Do đó, đề tài **"Agent AI sử dụng RAG"** là hướng nghiên cứu hiện đại, mang tính ứng dụng cao và phù hợp với xu thế phát triển của kỹ thuật AI thế hệ mới.

## 1.1 Mục tiêu nghiên cứu

Mục tiêu của đề tài là nghiên cứu và xây dựng một mô hình Agent AI ứng dụng kỹ thuật Retrieval-Augmented Generation (RAG) nhằm nâng cao độ chính xác, tính nhất quán và khả năng suy luận của các mô hình ngôn ngữ lớn. Cụ thể, đề tài hướng đến các mục tiêu sau:

1. **Tìm hiểu tổng quan về RAG và vai trò của nó trong cải thiện chất lượng sinh ngôn ngữ.** Làm rõ cách kết hợp giữa mô hình truy xuất tài liệu và mô hình sinh nhằm bổ sung thông tin chính xác, cập nhật và giảm thiểu hiện tượng *hallucination*.

2. **Khảo sát các biến thể RAG nâng cao** như Self-RAG, Reflexion Agentic RAG, BM25 Re-ranking, Adaptive RAG Routing. Phân tích ưu điểm, hạn chế và khả năng ứng dụng của từng phương pháp trong quá trình xây dựng Agent.

3. **Xây dựng một Agent AI có khả năng tự truy xuất, tự phản hồi và tự cải thiện câu trả lời.** Agent được kỳ vọng có khả năng:

   - xác định khi nào cần tìm thêm thông tin,
   - truy xuất tài liệu phù hợp,
   - đánh giá lại chất lượng phản hồi,
   - tự điều chỉnh để tạo ra câu trả lời tốt hơn.

4. **Thực nghiệm và đánh giá mô hình Agent sử dụng RAG.** Thử nghiệm nhằm đánh giá:

   - độ chính xác,
   - mức độ tin cậy,
   - khả năng suy luận,
   - sự ổn định khi thực hiện các nhiệm vụ khác nhau.

5. **Đề xuất hướng phát triển hệ thống Agent AI dựa trên RAG cho các ứng dụng thực tế.** Bao gồm các lĩnh vực như pháp lý, tài chính, hỗ trợ khách hàng, phân tích dữ liệu và trợ lý ảo.

## 1.2 Phạm vi và đối tượng nghiên cứu

**Phạm vi nghiên cứu:**

Phạm vi nghiên cứu của đề tài tập trung vào các kỹ thuật RAG hiện đại và các biến thể nâng cao nhằm xây dựng Agent AI thông minh, cụ thể bao gồm:

- **Self-RAG:** Hệ thống RAG có khả năng tự phản hồi, tự đánh giá và tự cải thiện câu trả lời thông qua cơ chế đánh giá mức độ liên quan của tài liệu, phát hiện hallucination và tự điều chỉnh truy xuất thông tin.
- **Reflexion Agentic RAG:** Biến thể RAG kết hợp với học phản hồi ngôn ngữ (*verbal reinforcement learning*) giúp Agent lặp lại quá trình tạo bản nháp, truy xuất tài liệu bổ sung, chỉnh sửa và hoàn thiện câu trả lời.
- **BM25 Re-ranking:** Kỹ thuật sắp xếp lại tài liệu truy xuất bằng thuật toán BM25Plus, nâng cao độ liên quan và giảm nhiễu trong kết quả trả về.
- **PageRAG Ingestion Pipeline:** Xử lý dữ liệu theo từng trang, trích xuất metadata thông minh, lưu vector cùng metadata vào cơ sở dữ liệu, giúp quản lý tài liệu dài và phức tạp như báo cáo tài chính hay hợp đồng pháp lý.
- **Adaptive RAG Routing:** Hệ thống phân nhánh truy vấn thông minh, chọn con đường xử lý phù hợp dựa trên loại dữ liệu và mục tiêu câu hỏi, ví dụ: vectorstore tài chính, SQL nội bộ, hoặc web search.

Nghiên cứu không đi sâu vào quá trình huấn luyện LLM từ đầu (*pretraining*) mà tập trung vào việc tích hợp LLM có sẵn với RAG và Agent framework, xây dựng pipeline hoàn chỉnh để triển khai các Agent AI có khả năng suy luận, truy xuất và phản hồi thông tin.

**Đối tượng nghiên cứu:**
Đối tượng nghiên cứu của đề tài gồm các thành phần sau:

- **Các mô hình ngôn ngữ lớn (LLM):** Bao gồm GPT, Llama và các mô hình tương tự, phục vụ cho việc sinh ngôn ngữ, reasoning, truy xuất kiến thức và phản hồi người dùng.
- **Vector Databases và kỹ thuật indexing:** Sử dụng FAISS, ChromaDB hoặc các cơ sở dữ liệu vector khác, kết hợp embedding và cấu trúc chỉ mục hiệu quả để truy xuất tài liệu nhanh chóng và chính xác.
- **Pipeline RAG thực tế:** Áp dụng trong các lĩnh vực như:
  - Tài chính: phân tích báo cáo SEC filings, báo cáo 10-K, 10-Q
  - Pháp lý: xử lý hợp đồng, case study, văn bản luật
  - Doanh nghiệp: quản lý dữ liệu nội bộ, báo cáo, kiến thức tổ chức
  - Trợ lý ảo: tạo câu trả lời chính xác, hỗ trợ nghiệp vụ, chăm sóc khách hàng

Nghiên cứu tập trung vào việc tích hợp **LLM + RAG + Agent Framework** nhằm tạo ra Agent AI thông minh, tự phản hồi, tự truy xuất dữ liệu và đưa ra quyết định chính xác.

### 1.3 Phương pháp nghiên cứu

Đề tài áp dụng phương pháp kết hợp giữa **nghiên cứu lý thuyết**, **mô hình thực nghiệm** và **đánh giá định lượng** để đảm bảo tính khoa học và độ tin cậy trong kết quả. Cụ thể:

1. **Nghiên cứu lý thuyết (Literature Review):**
   - Thu thập, tổng hợp các bài báo, sách, tài liệu kỹ thuật về RAG, Self-RAG, Reflexion, Adaptive RAG và các Agent Framework.
   - Phân tích các kỹ thuật vector search, re-ranking, query rewriting, metadata extraction và cơ chế tự đánh giá câu trả lời (*self-evaluation*) trong các pipeline hiện đại.
2. **Phân tích mô hình thực nghiệm:**
   - Xây dựng pipeline RAG mẫu sử dụng ChromaDB vectorstore để lưu trữ embedding.
   - Áp dụng BM25Plus Re-ranking để nâng cao chất lượng truy xuất.

- Sử dụng cơ chế Query Rewriting và Metadata Extraction để cải thiện độ chính xác và giảm sai lệch thông tin.
- Tích hợp Self-Evaluation nhằm phát hiện hallucination và đánh giá mức độ đầy đủ của câu trả lời.

3. **Mô phỏng và đánh giá:**

- Xây dựng Agent RAG mẫu với FastAPI backend và Streamlit frontend.
- Sử dụng LLM reasoning kết hợp truy xuất dữ liệu để đánh giá khả năng suy luận, độ chính xác và chất lượng câu trả lời.
- Thử nghiệm với nhiều loại truy vấn khác nhau để kiểm tra khả năng linh hoạt của Agent.

4. **So sánh định lượng:**

- Đánh giá hiệu quả RAG so với mô hình không sử dụng RAG.
- So sánh accuracy trước và sau khi áp dụng Re-ranking, Query Rewriting.
- Đo tốc độ truy xuất, độ liên quan và mức độ đầy đủ của câu trả lời.

Phương pháp nghiên cứu kết hợp giữa **lý thuyết – mô hình – thực nghiệm** nhằm đảm bảo kết luận chính xác, khoa học và có thể áp dụng vào các hệ thống Agent AI thực tế.

# 2: Trí tuệ nhân tạo và Large Language Models (LLMs)

## 2.1 Khái niệm AI và các nhánh liên quan

Trí tuệ nhân tạo (Artificial Intelligence – AI) là lĩnh vực nghiên cứu và phát triển các hệ thống máy tính có khả năng thực hiện các nhiệm vụ thông minh tương tự con người, bao gồm nhận thức, học tập, suy luận, lập kế hoạch và xử lý ngôn ngữ tự nhiên. Mục tiêu của AI là tạo ra những hệ thống không chỉ thực hiện các lệnh cố định mà còn có khả năng tự học và thích ứng với dữ liệu mới.

AI được chia thành nhiều nhánh chính, bao gồm:

- **Machine Learning (ML):** Tập trung vào việc xây dựng các thuật toán cho phép máy tính học từ dữ liệu. Thay vì lập trình chi tiết các quy tắc, ML dựa vào dữ liệu để tạo ra mô hình dự đoán hoặc phân loại.

- **Deep Learning (DL):** Là nhánh con của ML, sử dụng mạng nơ-ron nhiều lớp để học các biểu diễn phức tạp từ dữ liệu. DL đặc biệt hiệu quả với dữ liệu phi cấu trúc như hình ảnh, âm thanh và văn bản. Kiến trúc Transformer là nền tảng cho các Large Language Models (LLMs) hiện đại.

- **Natural Language Processing (NLP):** Nghiên cứu cách máy tính hiểu, phân tích và tạo ra ngôn ngữ tự nhiên. NLP bao gồm nhiều tác vụ như dịch máy, tóm tắt văn bản, trả lời câu hỏi, phân loại và sinh văn bản.

- **Reinforcement Learning (RL):** Tập trung vào việc học hành vi tối ưu dựa trên phản hồi từ môi trường. RL thường được sử dụng để tinh chỉnh LLM thông qua các cơ chế như Reinforcement Learning from Human Feedback (RLHF).

Nhờ sự phát triển đồng thời của các nhánh trên, AI có khả năng xử lý các tác vụ ngôn ngữ phức tạp, tạo ra các hệ thống trợ lý ảo, chatbots thông minh và Agent AI có khả năng suy luận.

## 2.2 Kiến trúc và cơ chế hoạt động của LLM

Large Language Models (LLMs) là các mô hình ngôn ngữ lớn, được huấn luyện trên lượng dữ liệu văn bản khổng lồ để sinh văn bản tự nhiên, trả lời câu hỏi, dịch ngôn ngữ và thực hiện nhiều tác vụ NLP khác. Các LLM hiện đại như GPT, Llama hay Claude chủ yếu dựa trên kiến trúc **Transformer** với các thành phần sau:

- **Tokenization:** Văn bản được chia thành các token (từ, cụm từ hoặc ký tự) để mô hình xử lý. Tokenization giúp chuẩn hóa dữ liệu, loại bỏ ký tự đặc biệt và giữ lại thông tin ngữ nghĩa.

- **Embedding:** Mỗi token được ánh xạ thành vector số trong không gian nhiều chiều. Embedding cho phép mô hình nắm bắt mối quan hệ ngữ nghĩa giữa các từ.

- **Mạng Transformer:** Bao gồm các lớp Encoder và Decoder, sử dụng cơ chế Self-Attention để xác định mức độ liên quan giữa các token, từ đó học bối cảnh ngôn ngữ.

- **Huấn luyện trên dữ liệu lớn:** LLM học từ hàng tỷ token văn bản để nắm bắt quy tắc ngữ pháp, cấu trúc câu, mối quan hệ ngữ nghĩa và kiến thức tổng quát.

- **Sinh văn bản (Generation):** Mô hình dự đoán token tiếp theo dựa trên xác suất học được từ dữ liệu huấn luyện, tạo ra câu trả lời hoặc đoạn văn bản hoàn chỉnh.

- **Fine-tuning và RLHF:** Để nâng cao hiệu quả trong các tác vụ cụ thể, LLM có thể được tinh chỉnh (*fine-tuning*) hoặc áp dụng RLHF để cải thiện chất lượng câu trả lời và giảm sai lệch.

## 2.3 Hạn chế của LLM truyền thống

Mặc dù LLM có khả năng sinh ngôn ngữ mạnh mẽ, chúng vẫn tồn tại một số hạn chế khi áp dụng thực tế:

- **Hallucination:** LLM đôi khi tạo ra thông tin sai hoặc không tồn tại. Hiện tượng này xảy ra vì mô hình dự đoán token dựa trên xác suất học được, không dựa trên dữ liệu thực tế.

- **Giới hạn dữ liệu huấn luyện:** LLM chỉ học từ dữ liệu có sẵn trong quá trình huấn luyện. Các sự kiện gần đây hoặc dữ liệu riêng của doanh nghiệp sẽ không được mô hình nhận biết nếu không huấn luyện lại.

- **Cập nhật thông tin hạn chế:** LLM truyền thống không có cơ chế truy xuất dữ liệu bên ngoài theo thời gian thực. Kiến thức của mô hình chỉ giới hạn ở thời điểm huấn luyện.

- **Khó kiểm soát về độ chính xác và logic:** LLM có thể trả lời mơ hồ hoặc phi logic khi câu hỏi phức tạp hoặc vượt quá phạm vi dữ liệu đã học.

Những hạn chế này đã tạo ra nhu cầu phát triển các kỹ thuật như **Retrieval-Augmented Generation (RAG)**, giúp kết hợp khả năng sinh ngôn ngữ với truy xuất dữ liệu thực tế, tăng độ chính xác và giảm hallucination.

## 2.4 Retrieval-Augmented Generation (RAG)

### 2.4.1 Định nghĩa RAG

Retrieval-Augmented Generation (RAG) là một kỹ thuật kết hợp giữa khả năng sinh ngôn ngữ của các Large Language Models (LLMs) và khả năng truy xuất thông tin từ các nguồn dữ liệu bên ngoài. Thay vì dựa hoàn toàn vào kiến thức đã học trong quá trình huấn luyện, RAG cho phép mô hình tham khảo tài liệu thực tế hoặc cơ sở dữ liệu được chuẩn hóa trước khi tạo câu trả lời.

Điều này giúp mô hình vừa tận dụng sức mạnh ngôn ngữ của LLM, vừa có khả năng truy cập thông tin mới, chính xác và cập nhật, giảm thiểu hiện tượng hallucination. RAG hiện được áp dụng rộng rãi trong các ứng dụng đòi hỏi độ chính xác cao, chẳng hạn như phân tích tài chính, tư vấn pháp lý, trợ lý ảo doanh nghiệp và hệ thống hỏi đáp học thuật.

### 2.4.2 Ý tưởng chính: Kết hợp truy xuất và mô hình sinh

Nguyên lý cơ bản của RAG bao gồm hai bước chính:

- **Retrieval (Truy xuất):** Khi nhận đầu vào (query), hệ thống sẽ truy xuất các tài liệu hoặc đoạn văn bản liên quan từ cơ sở dữ liệu, vector store hoặc nguồn dữ liệu bên ngoài. Việc này đảm bảo rằng mô hình dựa trên thông tin thực tế, thay vì chỉ dựa vào kiến thức đã học.

- **Generation (Sinh ngôn ngữ):** Sau khi có các tài liệu liên quan, LLM sẽ kết hợp thông tin này để tạo ra câu trả lời. Mô hình sinh câu trả lời một cách tự nhiên, có logic và có thể trích dẫn nguồn dữ liệu khi cần.

Một số biến thể nâng cao của RAG như Self-RAG hay Reflexion Agentic RAG còn tích hợp thêm các bước như: - **Đánh giá độ liên quan của tài liệu** (document relevance grading) - **Phát hiện hallucination** - **Kiểm tra tính đầy đủ của câu trả lời** - **Viết lại truy vấn** (query rewriting) để tăng khả năng truy xuất

Điều này giúp hệ thống có thể tự cải thiện kết quả trả lời, tương tự như quá trình tự phản hồi của con người.

### 2.4.3 Lợi ích

Việc kết hợp truy xuất và sinh ngôn ngữ trong RAG mang lại nhiều lợi ích đáng kể:

- **Giảm hallucination:** Bằng cách tham khảo dữ liệu thực tế, mô hình giảm khả năng tạo ra thông tin sai hoặc không tồn tại. Đây là điểm đặc biệt quan trọng trong các ứng dụng đòi hỏi độ chính xác cao.

- **Cập nhật theo dữ liệu riêng:** RAG cho phép mô hình truy xuất các tài liệu mới, dữ liệu doanh nghiệp hoặc thông tin chuyên ngành mà LLM truyền thống không có. Điều này giúp câu trả lời luôn cập nhật và phù hợp với bối cảnh hiện tại.

- **Giảm chi phí huấn luyện:** Thay vì huấn luyện lại toàn bộ mô hình để cập nhật thông tin mới, RAG chỉ cần cập nhật cơ sở dữ liệu hoặc embeddings. Điều này tiết kiệm chi phí tính toán và thời gian, đồng thời duy trì khả năng sinh ngôn ngữ mạnh mẽ của LLM.

Nhờ những lợi ích trên, RAG đang trở thành phương pháp chuẩn trong nhiều ứng dụng thực tế, từ trợ lý ảo, hệ thống hỏi đáp đến phân tích dữ liệu doanh nghiệp.

## 2.5 Kiến trúc tổng quan của RAG

Retrieval-Augmented Generation (RAG) không chỉ là một kỹ thuật kết hợp giữa sinh ngôn ngữ và truy xuất thông tin mà còn là một hệ thống phức tạp với nhiều thành phần phối hợp chặt chẽ. Hiểu rõ kiến trúc tổng quan của RAG giúp chúng ta xây dựng các pipeline hiệu quả, đồng thời tối ưu hóa khả năng truy xuất dữ liệu và chất lượng câu trả lời.

### 2.5.1 Thành phần Retrieval (Bộ truy xuất)

Bộ truy xuất là thành phần cốt lõi trong RAG, chịu trách nhiệm tìm kiếm và cung cấp thông tin liên quan đến truy vấn từ kho dữ liệu. Mục tiêu của Retrieval là đảm bảo rằng LLM nhận được các tài liệu thực tế và chính xác trước khi sinh câu trả lời.

Các chức năng chính của bộ truy xuất bao gồm:

- **Trích xuất dữ liệu liên quan:** Khi nhận truy vấn, hệ thống truy xuất các tài liệu hoặc đoạn văn bản từ cơ sở dữ liệu, có thể là vector store, database quan hệ hoặc dữ liệu phi cấu trúc. Việc trích xuất dựa trên các tiêu chí như từ khóa, embedding similarity hoặc metadata.

- **Xử lý truy vấn:** Truy vấn thường được chuẩn hóa, tách token và đôi khi được viết lại (query rewriting) để tăng khả năng truy xuất tài liệu chính xác. Việc này đặc biệt quan trọng với các truy vấn phức tạp hoặc thiếu từ khóa rõ ràng.

- **Tối ưu hóa hiệu quả:** Bộ truy xuất cần đảm bảo độ chính xác và tốc độ cao. Một số kỹ thuật phổ biến bao gồm tìm kiếm gần đúng (Approximate Nearest Neighbor – ANN), sử dụng chỉ mục HNSW hoặc IVF, và kết hợp với các bộ lọc metadata để giảm thiểu dữ liệu không liên quan.

Trong các ứng dụng thực tế, hiệu suất của bộ truy xuất quyết định phần lớn chất lượng câu trả lời. Nếu Retrieval không chính xác, LLM có thể sinh câu trả lời sai hoặc hallucination, dù khả năng sinh ngôn ngữ của mô hình vẫn tốt.

### 2.5.2 Thành phần Generator (LLM)

Generator là Large Language Model chịu trách nhiệm tạo câu trả lời dựa trên các tài liệu đã được Retrieval cung cấp. Đây là phần sinh ngôn ngữ của RAG, đóng vai trò biến thông tin thô thành câu trả lời tự nhiên, logic và dễ hiểu.

Các đặc điểm chính của Generator:

- **Tích hợp thông tin từ tài liệu:** Generator không chỉ sinh câu trả lời dựa trên truy vấn mà còn dựa vào nội dung tài liệu đã được truy xuất. Điều này giúp giảm hallucination và tăng tính chính xác.

- **Khả năng sinh ngôn ngữ tự nhiên:** Nhờ kiến trúc Transformer, Generator có thể tạo ra câu trả lời mạch lạc, phù hợp ngữ cảnh, thậm chí có thể viết theo phong cách chuyên ngành.

- **Tùy chỉnh và tinh chỉnh:** Generator có thể được fine-tuning hoặc sử dụng Reinforcement Learning from Human Feedback (RLHF) để cải thiện độ chính xác và logic. Một số hệ thống còn cho phép Generator tự đánh giá câu trả lời và yêu cầu tài liệu bổ sung nếu cần.

Generator và Retrieval phải phối hợp chặt chẽ. Thông tin mà Generator nhận được càng chính xác và đầy đủ, câu trả lời cuối cùng càng đúng và ít sai lệch.

### 2.5.3 Vector Database (FAISS, Milvus, Chroma...)

Vector Database là thành phần lưu trữ các embeddings của tài liệu, giúp hệ thống truy xuất thông tin nhanh và hiệu quả. Khi các đoạn văn bản được chuyển đổi thành embeddings số, chúng được lưu trong Vector Database để phục vụ cho việc tìm kiếm gần đúng.

Một số điểm quan trọng:

- **FAISS:** Là thư viện mã nguồn mở của Facebook, hỗ trợ tìm kiếm ANN trên dữ liệu embedding lớn. FAISS nổi bật với khả năng tối ưu bộ nhớ và tốc độ truy xuất cao.

- **Milvus:** Là hệ quản trị vector dữ liệu mạnh mẽ, hỗ trợ phân tán, khả năng mở rộng linh hoạt và tích hợp dễ dàng với nhiều loại LLM. Milvus thường được dùng trong môi trường doanh nghiệp lớn với dữ liệu hàng triệu embeddings.

- **Chroma:** Một Vector Database nhẹ, dễ triển khai, phù hợp với các dự án nhỏ và thử nghiệm. Chroma hỗ trợ quản lý metadata đi kèm embeddings và tích hợp trực tiếp với pipeline RAG.

Vector Database không chỉ lưu embeddings mà còn lưu metadata quan trọng như tiêu đề, ngày tạo, loại tài liệu. Điều này giúp cải thiện khả năng lọc và re-ranking, tăng độ chính xác khi truy xuất tài liệu liên quan.

### 2.5.4 Pipeline RAG chuẩn

Pipeline RAG chuẩn bao gồm một chuỗi các bước phối hợp từ ingestion dữ liệu đến sinh câu trả lời. Quy trình cơ bản gồm:

- **Ingestion:** Dữ liệu được thu thập từ nhiều nguồn (PDF, trang web, cơ sở dữ liệu, API) và chuẩn hóa, tách chunk văn bản, trích xuất metadata.

- **Embedding:** Mỗi chunk văn bản được chuyển thành vector số (embedding) để lưu trong Vector Database. Embedding capture ngữ nghĩa và mối quan hệ giữa các từ, giúp truy xuất thông tin chính xác.

- **Indexing:** Tạo chỉ mục cho các embeddings, sử dụng ANN (HNSW, IVF) để tối ưu tốc độ truy xuất. Indexing cũng có thể bao gồm re-ranking theo BM25 hoặc các phương pháp hybrid để tăng độ liên quan.

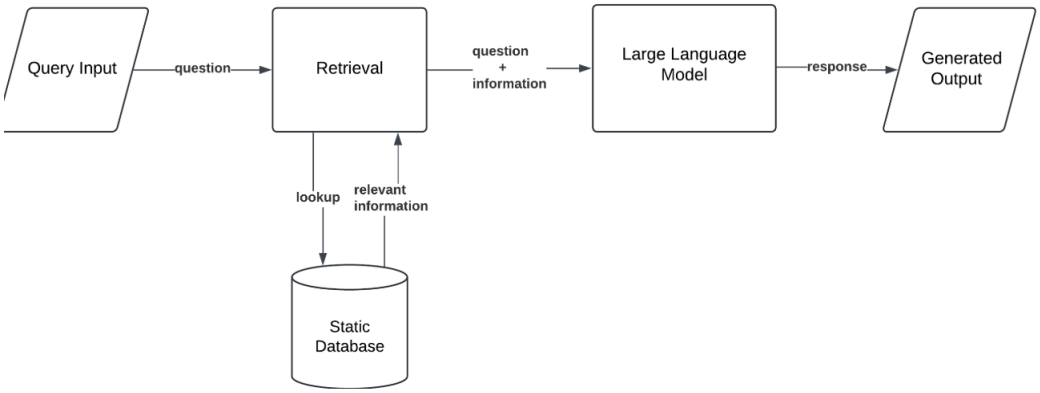
- **Query:** Truy vấn của người dùng được tiền xử lý, tokenized, và chuyển thành embedding nếu dùng vector search. Một số hệ thống còn thực hiện query rewriting để tăng khả năng tìm kiếm tài liệu liên quan.

- **Retrieval:** Dựa trên embedding query, hệ thống tìm kiếm các chunk tài liệu gần nhất trong Vector Database, lọc theo metadata và đánh giá độ liên quan.

- **Generation:** Generator nhận các tài liệu liên quan và truy vấn, kết hợp thông tin để sinh câu trả lời tự nhiên, mạch lạc, có logic và chính xác.

- **Optional Re-ranking / Feedback Loop:** Một số pipeline nâng cao áp dụng đánh giá câu trả lời, phát hiện hallucination, hoặc tự truy xuất thêm nếu câu trả lời chưa đầy đủ.

Chuỗi quy trình này giúp RAG trở thành hệ thống linh hoạt, có khả năng cập nhật thông tin mới mà không cần huấn luyện lại toàn bộ LLM. Đồng thời, pipeline cũng đảm bảo tốc độ, độ chính xác và khả năng mở rộng theo khối lượng dữ liệu lớn.




## 2.6 Embedding và Vector Search

Retrieval-Augmented Generation (RAG) dựa nhiều vào khả năng truy xuất thông tin từ kho dữ liệu lớn. Một thành phần then chốt giúp RAG thực hiện truy xuất hiệu quả là embedding, kết hợp với các phương pháp tìm kiếm vector (vector search). Phần này trình bày khái niệm, các mô hình embedding phổ biến, khoảng cách đánh giá similarity và kỹ thuật tối ưu truy xuất.

### 2.6.1 Khái niệm embedding

Embedding là một biểu diễn số học (vector) của dữ liệu, trong đó thông tin ngữ nghĩa được giữ lại. Trong NLP, embedding ánh xạ các token, câu hoặc đoạn văn bản vào không gian nhiều chiều sao cho các biểu diễn vector gần nhau có nghĩa ngữ nghĩa tương tự.

Một số đặc điểm quan trọng của embedding:

- Giữ mối quan hệ ngữ nghĩa: Các từ hoặc câu có nghĩa gần nhau trong ngôn ngữ được ánh xạ gần nhau trong không gian vector. Ví dụ, "king" và "queen" sẽ có embeddings gần nhau hơn so với "king" và "apple".

- Chuẩn hóa dữ liệu: Embedding giúp chuẩn hóa dữ liệu phi cấu trúc thành dạng số, phục vụ cho việc tính toán similarity, clustering hay truy xuất.

- Dễ dàng tích hợp với mô hình học máy: Các embeddings có thể được lưu trong Vector Database để truy xuất nhanh hoặc sử dụng làm input cho các mô hình downstream.

Embedding là bước nền tảng của pipeline RAG, vì chất lượng embeddings quyết định độ chính xác của retrieval và cuối cùng là chất lượng câu trả lời.

### 2.6.2 Các mô hình embedding (OpenAI, BGE, Instructor, . . . )

Hiện nay, có nhiều mô hình embedding phổ biến, cung cấp các vector chất lượng cao, phù hợp với nhiều ngôn ngữ và ứng dụng:

- **OpenAI Embeddings:** Các mô hình như `text-embedding-3-small` và `text-embedding-3-large` cho phép chuyển câu, đoạn văn hay tài liệu thành vector 1536 chiều. Chúng hỗ trợ tốt các tác vụ truy xuất thông tin, semantic search, clustering.

- **BGE (BigScience General Embeddings):** Mô hình open-source, hỗ trợ nhiều ngôn ngữ và tập trung vào chất lượng semantic embedding. BGE thích hợp cho các ứng dụng yêu cầu truy xuất đa ngôn ngữ và tổng hợp thông tin phức tạp.

- **Instructor:** Một framework cho phép fine-tuning embedding cho các tác vụ cụ thể. Ví dụ, bạn có thể huấn luyện embedding tối ưu cho truy xuất tài liệu tài chính hoặc pháp lý, từ đó tăng độ chính xác cho RAG.

- **Các mô hình khác:** Ngoài ra còn có các embedding từ HuggingFace, Sentence-BERT, LaBSE... mỗi mô hình có điểm mạnh riêng về ngữ cảnh, ngôn ngữ hoặc tốc độ truy xuất.

Việc lựa chọn embedding phù hợp phụ thuộc vào:

1. Loại dữ liệu (văn bản, PDF, đoạn văn) 2. Ngôn ngữ 3. Kích thước dữ liệu 4. Mức độ chính xác yêu cầu cho retrieval

### 2.6.3 Khoảng cách: cosine / dot product / L2

Sau khi dữ liệu được chuyển thành vector, việc đánh giá mức độ liên quan giữa truy vấn và tài liệu dựa trên các phép đo khoảng cách (distance metric) hoặc độ tương đồng (similarity):

- **Cosine similarity:** Đo góc giữa hai vector. Giá trị từ -1 đến 1, với 1 nghĩa là hai vector hoàn toàn giống nhau về hướng. Phổ biến trong NLP vì nó chỉ quan tâm đến hướng vector, bỏ qua độ lớn.

- **Dot product:** Tích vô hướng giữa hai vector. Thường dùng khi embeddings đã được chuẩn hóa và muốn kết hợp với trọng số attention trong một số kiến trúc retrieval.

- **L2 distance (Euclidean distance):** Khoảng cách Euclidean giữa hai vector. Phù hợp khi cần đo khoảng cách thực sự trong không gian vector, đặc biệt với dữ liệu có scale chuẩn.

Việc lựa chọn metric ảnh hưởng trực tiếp đến chất lượng retrieval. Trong RAG, cosine similarity và dot product là phổ biến hơn, đặc biệt với embeddings chuẩn hóa, vì chúng thể hiện rõ mối quan hệ ngữ nghĩa.

### 2.6.4 Tối ưu truy xuất: ANN (FAISS HNSW, IVF . . . )

Với khối lượng dữ liệu lớn, tìm kiếm exact nearest neighbor trở nên tốn kém về thời gian và bộ nhớ. Do đó, các kỹ thuật tìm kiếm gần đúng (Approximate Nearest Neighbor – ANN) được áp dụng để tối ưu hóa tốc độ và hiệu quả:

- **FAISS (Facebook AI Similarity Search):** Thư viện phổ biến, hỗ trợ nhiều chỉ mục ANN như HNSW, IVF, PQ, giúp tìm kiếm nhanh trên hàng triệu vector.

- **HNSW (Hierarchical Navigable Small World):** Cấu trúc đồ thị nhiều lớp, cho phép tìm kiếm ANN hiệu quả, với tốc độ cao và độ chính xác tốt. Thường dùng trong môi trường doanh nghiệp cần truy xuất thời gian thực.

- **IVF (Inverted File Index):** Chia không gian vector thành các cluster, tìm kiếm trong cluster gần query nhất. Giảm đáng kể số lượng vector cần so sánh, tiết kiệm tài nguyên.

- **Kết hợp hybrid:** Một số hệ thống kết hợp ANN với BM25 hoặc các bộ lọc metadata để vừa tối ưu tốc độ, vừa tăng độ chính xác.

Các kỹ thuật ANN này là nền tảng để triển khai RAG với dữ liệu lớn, cho phép Agent AI truy xuất nhanh, phản hồi kịp thời mà vẫn đảm bảo chất lượng câu trả lời.

### 2.6.5 Kết luận

Embedding và Vector Search là thành phần không thể thiếu trong kiến trúc RAG. Chất lượng embeddings, phương pháp đo khoảng cách và kỹ thuật tìm kiếm ANN quyết định hiệu quả truy xuất thông tin và giảm hallucination. Hiểu rõ các thành phần này giúp xây dựng pipeline RAG tối ưu, đảm bảo tốc độ, độ chính xác và khả năng mở rộng cho các ứng dụng thực tế như tài chính, pháp lý, trợ lý ảo hay phân tích dữ liệu doanh nghiệp.

# 3: Page-RAG: Data Ingestion Pipeline cho Retrieval-Augmented Generation

## 3.1 Giới thiệu

Trong lĩnh vực trí tuệ nhân tạo và xử lý ngôn ngữ tự nhiên, Retrieval-Augmented Generation (RAG) đã trở thành một phương pháp hiệu quả để kết hợp khả năng sinh ngôn ngữ của các Large Language Models (LLMs) với khả năng truy xuất dữ liệu thực tế từ các nguồn bên ngoài. Phương pháp RAG truyền thống, mặc dù đã cải thiện đáng kể độ chính xác và giảm hallucination, vẫn gặp một số hạn chế khi phải xử lý các tài liệu dài hoặc phức tạp. Các hạn chế này bao gồm việc LLM phải đọc toàn bộ document, dẫn đến tiêu tốn tài nguyên tính toán, khả năng bỏ sót thông tin quan trọng, và khó khăn trong việc lọc dữ liệu theo các thuộc tính như năm, loại tài liệu hay tác giả.

Page-RAG là một bước tiến trong việc khắc phục các vấn đề này. Thay vì lấy toàn bộ document làm đơn vị cơ bản, Page-RAG chia tài liệu thành các page-level chunks, mỗi trang được xử lý riêng biệt, embedding thành vector và gắn metadata chi tiết. Cách tiếp cận này giúp

LLM tập trung vào các thông tin liên quan, tăng độ chính xác của truy xuất và tối ưu hóa hiệu suất tính toán. Metadata extraction kết hợp với vector database giúp triển khai các Agent AI thông minh có khả năng truy xuất, re-ranking và sinh câu trả lời chính xác dựa trên dữ liệu thực tế.

### 3.1.1 Ưu điểm nổi bật của Page-RAG

Page-RAG mang lại nhiều ưu điểm vượt trội so với RAG truyền thống, đặc biệt trong môi trường doanh nghiệp hoặc tài liệu học thuật lớn:

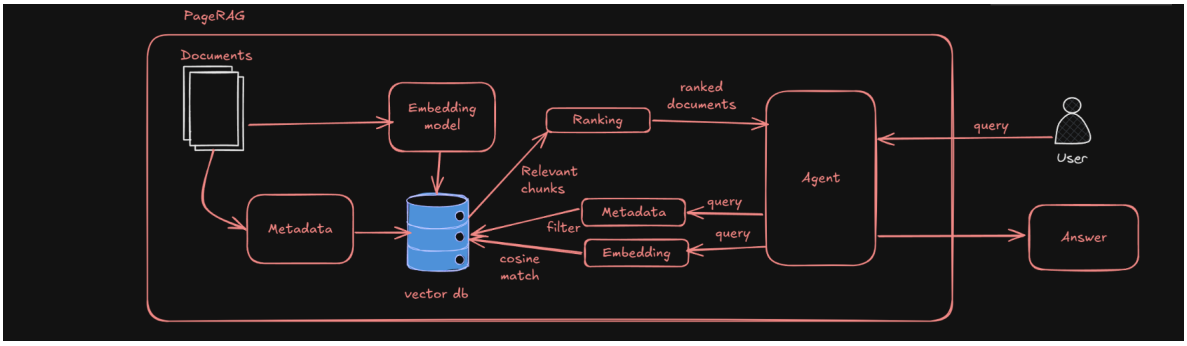
- **Truy xuất chính xác theo từng trang:** Thay vì xử lý toàn bộ document, LLM chỉ cần tập trung vào các page-level chunks liên quan đến truy vấn. Điều này vừa giảm tải bộ nhớ, vừa tăng tốc độ sinh câu trả lời.
- **Metadata filtering:** Các metadata chi tiết (năm, loại tài liệu, tác giả, section) cho phép lọc thông tin theo nhiều tiêu chí, giúp hệ thống truy xuất chính xác các chunk liên quan nhất.
- **Xử lý dữ liệu lớn:** Hàng nghìn tài liệu, mỗi document chia thành nhiều page-level chunks, vẫn có thể được lưu trữ, truy xuất và sinh câu trả lời hiệu quả nhờ vector database tối ưu.
- **Tăng khả năng tích hợp Agent AI:** Page-RAG hỗ trợ các agent nâng cao như Self-RAG, Reflexion RAG hay Adaptive RAG, từ đó giúp hệ thống AI có khả năng tự phản hồi, học hỏi và cải thiện chất lượng câu trả lời theo thời gian.

### 3.1.2 Ứng dụng thực tiễn

Page-RAG có thể áp dụng rộng rãi trong nhiều lĩnh vực nhờ khả năng xử lý dữ liệu lớn, truy xuất chính xác và metadata filtering:

- **Tài chính:** Phân tích báo cáo SEC filings (10-K, 10-Q), giúp các chuyên gia tài chính trích xuất thông tin theo công ty, năm, quý hoặc section cụ thể, từ đó đưa ra dự báo và quyết định chiến lược.
- **Pháp lý:** Quản lý hợp đồng, hồ sơ pháp lý, và case law. Metadata filtering cho phép lọc theo section, điều khoản, ngày ký, hoặc thẩm quyền, giúp luật sư tìm kiếm nhanh các phần thông tin cần thiết.
- **Học thuật:** Index các bài báo, luận văn, báo cáo nghiên cứu với metadata chi tiết như tác giả, năm xuất bản, chủ đề và section, giúp việc trích xuất thông tin nghiên cứu trở nên hiệu quả hơn.
- **Doanh nghiệp:** Xây dựng knowledge base nội bộ, trợ lý AI doanh nghiệp có khả năng truy xuất dữ liệu chính xác từ nhiều dự án và bộ phận khác nhau, hỗ trợ ra quyết định và tự động hóa quy trình công việc.
- **Y tế:** Quản lý patient records, lọc theo ngày, phòng ban, bác sĩ điều trị, giúp hệ thống y tế truy xuất thông tin nhanh và hỗ trợ chẩn đoán, báo cáo và nghiên cứu.

## 3.2 Các thành phần chính của Page-RAG



### 3.2.1 Document Ingestion

Document ingestion là bước đầu tiên và nền tảng trong Page-RAG, đảm bảo rằng dữ liệu đầu vào được chuẩn hóa trước khi đưa vào hệ thống. Các bước chính bao gồm:

- **Text extraction:** Chuyển đổi nội dung từ PDF, Word, HTML hoặc text sang dạng text chuẩn. Các công cụ phổ biến như doclink, PyPDF2, pdfminer hoặc Tika giúp trích xuất thông tin chính xác từ các định dạng khác nhau.
- **Page splitting:** Chia document thành các page-level chunks để tăng khả năng truy xuất. Mỗi page được coi là một đơn vị dữ liệu độc lập.
- **Data normalization:** Chuẩn hóa văn bản, loại bỏ ký tự đặc biệt, gộp dòng và chuẩn hóa định dạng, giúp embeddings chính xác hơn.

### 3.2.2 Metadata Extraction

Metadata là các thông tin mô tả giúp đánh giá relevance và hỗ trợ truy xuất:

- **Các loại metadata:** Tên công ty, loại tài liệu, năm/quý, section, tiêu đề, tác giả.
- **Phương pháp trích xuất:** Kết hợp LLM để phân tích văn bản hoặc rule-based parsing cho các thông tin có cấu trúc rõ ràng.
- **Vai trò của metadata:**
  - Lọc các page không liên quan trước khi gửi tới LLM.
  - Hỗ trợ re-ranking và query filtering, tăng độ chính xác của retrieval.
  - Chuẩn hóa dữ liệu, tạo điều kiện thuận lợi cho việc tích hợp với các hệ thống Self-RAG, Reflexion RAG.

### 3.2.3 Vectorization và Storage

Mỗi page được embedding thành vector số học, lưu trữ trong một vector database:

- **Mô hình embedding:** OpenAI, BGE, Instructor hoặc các mô hình đa ngôn ngữ khác.
- **Vector Database:** Chroma, FAISS, Milvus hỗ trợ lưu trữ và truy xuất hàng triệu vector.
- **Chức năng:**
  - Truy xuất nhanh theo vector similarity.
  - Metadata filtering để lọc page theo nhiều tiêu chí.
  - Hỗ trợ mở rộng quy mô với dữ liệu lớn.

### 3.2.4 Duplicate Detection và Hashing

Tránh duplicate ingestion bằng cách:

- Tạo **hash** từ nội dung của từng page.
- Kiểm tra hash trước khi ingest để đảm bảo page chưa được lưu trữ.

### 3.2.5 Query Processing và Retrieval

Khi người dùng gửi query, hệ thống thực hiện các bước:

1. Chuyển query thành vector embedding.
2. Tìm các page gần nhất trong vector database dựa trên similarity.
3. Lọc các page theo metadata.
4. Optional: re-ranking bằng BM25 hoặc MMR để tăng độ liên quan và chất lượng truy xuất.

### 3.2.6 Integration với Agent AI và LLM

Sau khi retrieval, Agent AI nhận page-level chunks và metadata để:

- **Document relevance grading:** Đánh giá độ liên quan của từng page đối với truy vấn.
- **Hallucination check:** Kiểm tra nội dung sinh ra có khớp dữ liệu thực tế hay không.
- Sinh câu trả lời logic, chính xác dựa trên các chunk và metadata.
- **Query rewriting:** Nếu thông tin chưa đủ, tự động tạo truy vấn mới và thực hiện retrieval.

## 3.3 Ưu điểm, ứng dụng và kết luận

### 3.3.1 Ưu điểm

Page-RAG đem lại nhiều lợi ích quan trọng, đặc biệt trong việc triển khai RAG với dữ liệu lớn và đa dạng:

- Truy xuất chính xác theo page, giảm hallucination.
- Metadata filtering cho phép lọc thông tin theo nhiều tiêu chí.
- Tối ưu storage, tránh duplicate và giảm kích thước database.
- Dễ dàng mở rộng với dữ liệu lớn, hàng triệu page.
- Hỗ trợ tích hợp với Self-RAG, Reflexion RAG, Adaptive RAG để nâng cao chất lượng câu trả lời.

### 3.3.2 Ứng dụng thực tế

1. **Tài chính:** SEC filings, báo cáo công ty theo section, năm, quý.
2. **Pháp lý:** Hợp đồng, case law theo section và điều khoản.
3. **Học thuật:** Index paper, thesis, trích xuất theo tác giả, năm, section.
4. **Doanh nghiệp:** Knowledge base nội bộ, trợ lý AI truy xuất nhanh.
5. **Y tế:** Patient records, truy xuất theo date, department, provider.

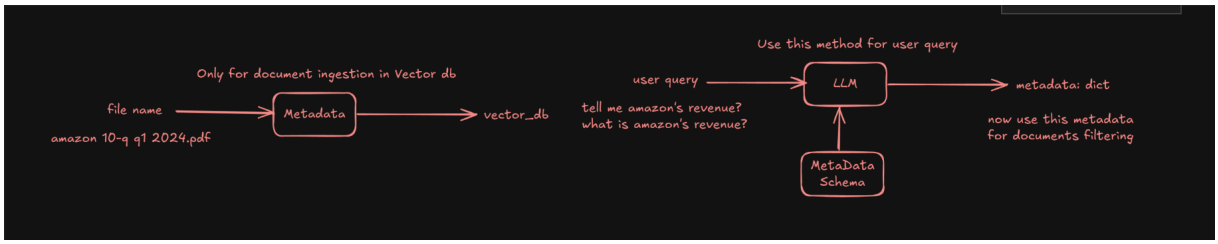
### 3.3.3 Kết luận

Page-RAG là công cụ nền tảng giúp triển khai RAG hiệu quả trên các tài liệu lớn và đa dạng. Việc chia nhỏ document, gắn metadata, embedding và lưu trong vector database mang lại các lợi ích:

- Tăng độ chính xác và tốc độ retrieval.
- Giảm hallucination và dữ liệu dư thừa.
- Hỗ trợ Agent AI sinh câu trả lời logic, chính xác, đầy đủ.
- Là nền tảng để triển khai Self-RAG, Reflexion RAG, Adaptive RAG, giúp hệ thống AI thông minh, tự học và cải thiện chất lượng câu trả lời dựa trên dữ liệu thực tế.

Page-RAG hiện đang được áp dụng trong nhiều lĩnh vực như tài chính, pháp lý, y tế, doanh nghiệp và học thuật. Nó không chỉ tối ưu hóa retrieval mà còn nâng cao khả năng tích hợp Agent AI, cung cấp cơ sở lý thuyết và thực hành cho các hệ thống RAG tiên tiến.

## 3.4 Tiền xử lý dữ liệu trong Vector Database




Hình trên minh họa quy trình tiền xử lý dữ liệu trước khi lưu trữ vào **Vector Database**. Quy trình này được thiết kế nhằm chuẩn hóa dữ liệu, loại bỏ các yếu tố nhiễu, đồng thời đảm bảo dữ liệu có chất lượng cao để phục vụ cho các bước truy xuất thông tin của các hệ thống RAG. Mỗi bước trong pipeline đều đóng vai trò quan trọng, từ việc trích xuất văn bản thô, chia nhỏ tài liệu, trích xuất metadata, đến việc embedding và lưu trữ trong cơ sở dữ liệu vector.

### 3.4.1 Text Extraction

Bước đầu tiên trong pipeline là trích xuất văn bản từ các tài liệu thô. Dữ liệu có thể ở nhiều định dạng khác nhau, bao gồm PDF, Word, HTML hoặc text thô. Quá trình này sử dụng các công cụ như `doclink`, `PyPDF2`, `pdfminer` hoặc `tika` để chuyển đổi nội dung thành dạng văn bản chuẩn hóa. Mục tiêu của việc trích xuất này là giữ lại toàn bộ thông tin quan trọng, đồng thời duy trì cấu trúc cơ bản của tài liệu, bao gồm các heading, paragraph và metadata gốc nếu có. Việc chuẩn hóa văn bản ở bước này đảm bảo rằng các bước xử lý tiếp theo hoạt động trên dữ liệu sạch, giảm nguy cơ mất mát thông tin và lỗi khi tạo embeddings.

### 3.4.2 Page Splitting

Sau khi văn bản đã được chuẩn hóa, tài liệu được chia thành các **page-level chunks**. Mỗi trang hoặc đoạn văn bản được xử lý như một đơn vị riêng biệt nhằm tăng độ chính xác của quá trình truy xuất sau này. Phân chia theo page giúp mô hình LLM tập trung vào thông tin liên quan cho từng câu hỏi cụ thể, giảm thiểu hiện tượng hallucination, đồng thời cho phép gắn metadata riêng biệt cho từng chunk. Kỹ thuật này đặc biệt hữu ích khi làm việc với tài liệu lớn, ví dụ như báo cáo SEC filings, hợp đồng pháp lý hay luận văn nghiên cứu, nơi mỗi page chứa thông tin quan trọng cần truy xuất chính xác.

### 3.4.3 Metadata Extraction

Metadata đóng vai trò quan trọng trong việc định hướng quá trình truy xuất thông tin. Metadata là các thông tin mô tả nội dung tài liệu, như tên công ty, loại tài liệu, năm/quý, section hoặc tác giả. Có hai cách chính để trích xuất metadata: sử dụng các phương pháp rule-based dựa trên template hoặc định dạng tài liệu, và phương pháp dựa trên LLM, trong đó mô hình ngôn ngữ nhận diện và gán giá trị metadata một cách tự động. Metadata không chỉ giúp lọc các page không liên quan, mà còn hỗ trợ các bước re-ranking, query filtering, và tích hợp dễ dàng với các hệ thống Self-RAG, Reflexion RAG, hoặc Adaptive RAG. Khi metadata được chuẩn hóa tốt, hệ thống có thể tăng độ chính xác của truy xuất, tiết kiệm tài nguyên và giảm thời gian xử lý.

### 3.4.4 Vectorization

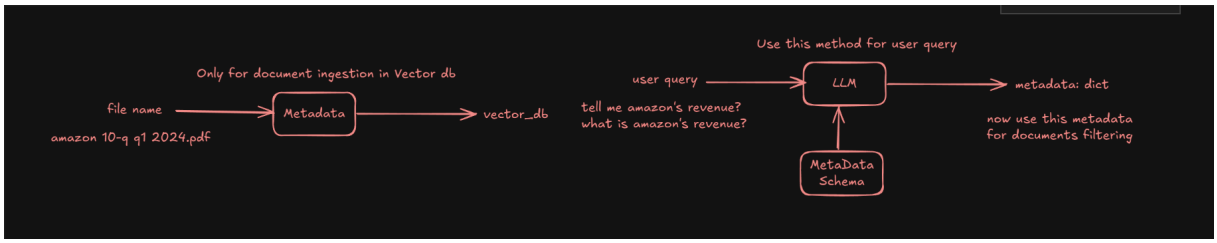
Sau khi dữ liệu đã được chia nhỏ và gắn metadata, từng chunk được chuyển đổi thành các **embedding vector** bằng các mô hình embedding hiện đại như OpenAI, BGE hoặc Instructor. Vectorization giữ lại mối quan hệ ngữ nghĩa của các từ, câu, hoặc đoạn văn bản, giúp hệ thống có thể tính toán similarity giữa query và tài liệu một cách chính xác. Các embedding này sẽ được lưu trữ trong Vector Database, cho phép tìm kiếm gần đúng (Approximate Nearest Neighbor - ANN) và hỗ trợ các bước retrieval, re-ranking, cũng như query filtering hiệu quả. Chất lượng của embedding quyết định trực tiếp độ chính xác của quá trình truy xuất thông tin và giảm thiểu hallucination của LLM khi sinh câu trả lời.

### 3.4.5 Duplicate Detection và Hashing

Để tránh việc ingest dữ liệu trùng lặp, mỗi chunk được tạo một giá trị hash dựa trên nội dung của nó. Trước khi lưu vào Vector Database, hệ thống kiểm tra hash của chunk để đảm bảo rằng không có dữ liệu nào đã tồn tại trong cơ sở dữ liệu. Kỹ thuật này không chỉ giúp tiết kiệm không gian lưu trữ mà còn đảm bảo dữ liệu trong Vector Database luôn duy trì tính duy nhất, tránh trùng lặp và dư thừa thông tin. Hashing là bước quan trọng, đặc biệt khi hệ thống ingest hàng nghìn hoặc hàng triệu page-level chunks từ nhiều nguồn khác nhau.

### 3.4.6 Lưu trữ vào Vector Database

Cuối cùng, các embedding cùng metadata được lưu trữ trong Vector Database, chẳng hạn như Chroma, FAISS hoặc Milvus. Việc lưu trữ này cho phép truy xuất nhanh theo vector similarity và đồng thời hỗ trợ filtering dựa trên metadata. Vector Database được thiết kế để mở rộng, có khả năng quản lý hàng triệu vector, phục vụ các hệ thống RAG quy mô lớn. Khi tích hợp với Agent AI, các chunk được retrieve sẽ được đánh giá relevance, kiểm tra hallucination và sử dụng để sinh câu trả lời logic, chính xác và đầy đủ. Nhờ pipeline này, hệ thống Page-RAG có thể triển khai retrieval hiệu quả trên dữ liệu lớn và đa dạng, đồng thời hỗ trợ các hệ thống tự phản hồi như Self-RAG, Reflexion RAG, và Adaptive RAG.




Hình 1: Sơ đồ minh họa quy trình tiền xử lí dữ liệu của Vector Database

# 4: RAG Data Retrieval

## 4.1 Tổng quan

Trong môi trường dữ liệu tài chính như SEC filings (báo cáo 10-K, 10-Q), việc truy xuất thông tin chính xác là một nhiệm vụ khó khăn do số lượng tài liệu lớn và cấu trúc phức tạp. Phương pháp Retrieval-Augmented Generation (RAG) giúp giải quyết vấn đề này bằng cách kết hợp giữa truy xuất thông tin dựa trên Vector Database và khả năng sinh ngôn ngữ của Large Language Models (LLM).

Workflow chính của Page-RAG trong ví dụ này bao gồm các bước:

- Setup: Khởi tạo vector store (ChromaDB), embeddings (Ollama Embeddings), và LLM (ChatOllama).
- Metadata Extraction: Từ truy vấn người dùng, trích xuất thông tin như công ty, loại tài liệu, năm và quý tài chính. **Keyword Generation: Sinh 5 từ khóa chuyên biệt từ thuật ngữ SEC filings để tăng độ chính xác truy xuất.**
- Filtered Search: Tìm tài liệu dựa trên metadata và từ khóa nội dung, sử dụng MMR search.
- BM25 Re-Ranking: Sắp xếp lại kết quả dựa trên điểm số BM25Plus trên heading và nội dung từng page để tăng độ liên quan

## 4.2 RAG Data Retrieval and Re-Ranking: Workflow và Giải thích Code

Trong phần này, chúng tôi trình bày quy trình truy xuất tài liệu thông minh từ các báo cáo tài chính SEC filings, bao gồm các bước thiết lập hệ thống, trích xuất metadata, sinh từ khóa, truy xuất dữ liệu từ vector database và sắp xếp lại kết quả bằng BM25Plus để tăng độ chính

xác. Mục tiêu của phương pháp **Retrieval-Augmented Generation (RAG)** là kết hợp giữa khả năng truy xuất dữ liệu chính xác từ cơ sở dữ liệu vector và khả năng sinh ngôn ngữ logic, đầy đủ của Large Language Models (LLM), từ đó giảm thiểu tình trạng LLM sinh thông tin không chính xác (hallucination) và tối ưu hóa hiệu quả truy xuất trong môi trường dữ liệu lớn và phức tạp.

### 4.2.1 Setup và Cấu hình Hệ thống

- **CHROMA_DIR:** Thư mục lưu trữ dữ liệu vector của ChromaDB, nơi các vector embedding được sinh từ từng page của tài liệu. Việc lưu trữ tách biệt và chuẩn hóa này cho phép truy xuất nhanh chóng, dễ dàng mở rộng quy mô, và duy trì tính nhất quán của dữ liệu khi số lượng tài liệu tăng lên hàng nghìn, thậm chí hàng triệu page.
- **COLLECTION_NAME:** Tên của collection trong ChromaDB, tương ứng với bảng dữ liệu trong cơ sở dữ liệu quan hệ. Collection này tổ chức các vector embedding theo từng chủ đề hoặc loại tài liệu, giúp việc lọc theo metadata trở nên trực quan và hiệu quả, đồng thời cho phép quản lý dữ liệu vector một cách linh hoạt và mở rộng.
- **EMBEDDING_MODEL:** Mô hình dùng để chuyển đổi văn bản thành vector số học, nhằm tính toán độ tương đồng giữa truy vấn của người dùng và nội dung tài liệu. Mỗi page được embedding thành một vector riêng, cho phép truy xuất theo similarity, hỗ trợ các kỹ thuật MMR hoặc BM25Plus trong các bước re-ranking sau này.
- **LLM_MODEL:** Mô hình ngôn ngữ lớn (LLM) chịu trách nhiệm sinh truy vấn, trích xuất metadata, tạo từ khóa và sinh câu trả lời logic dựa trên dữ liệu đã retrieve. Sự kết hợp giữa vector embedding và LLM giúp hệ thống vừa truy xuất chính xác vừa sinh ngôn ngữ tự nhiên có ý nghĩa.

Các đối tượng code tương ứng:

```
embeddings= OllamaEmbeddings(model=EMBEDDING_MODEL, base_url=BASE_URL)
vector_store = Chroma(collection_name=COLLECTION_NAME, embedding_function=embeddings, persis
llm = ChatOllama(model=LLM_MODEL, base_url=BASE_URL)
```

**OllamaEmbeddings** chịu trách nhiệm chuyển đổi văn bản thành vector số học. Mỗi page tài liệu khi ingest sẽ được embedding và lưu vào vector store, đảm bảo khả năng truy xuất nhanh và chính xác.

**Chroma** là vector database cho phép truy xuất các vector tương đồng với truy vấn, kết hợp metadata filtering để giảm số lượng page phải duyệt và tối ưu hiệu quả retrieval.

**ChatOllama** là LLM, được sử dụng để trích xuất metadata, sinh từ khóa đặc thù và sinh câu trả lời logic dựa trên page đã retrieve, đóng vai trò quan trọng trong việc giảm hallucination và đảm bảo thông tin đầy đủ.

### 4.2.2 Metadata Extraction

Metadata là thông tin mô tả tài liệu, bao gồm công ty, loại báo cáo, năm, quý và các thông tin định danh khác. Metadata được sử dụng để:

**Lọc các page không liên quan:** Chỉ những tài liệu phù hợp với truy vấn mới được retrieve, giảm số lượng dữ liệu cần xử lý và tăng độ chính xác.

**Hỗ trợ re-ranking:** Metadata cho phép hệ thống ưu tiên các tài liệu liên quan hơn khi thực hiện sắp xếp BM25Plus hoặc các kỹ thuật MMR.

**Chuẩn hóa dữ liệu và tích hợp với LLM:** Metadata chuẩn hóa giúp LLM xử lý thông tin dễ dàng, đồng thời hỗ trợ các hệ thống cao cấp như Self-RAG, Reflexion-RAG, và Adaptive-RAG.

Ví dụ, truy vấn "Amazon Q3 2024 revenue" sẽ được LLM trích xuất metadata như sau:

```
{company_name: "amazon", doc_type: "10-q", fiscal_year: 2024, fiscal_quarter: "q3"}
```

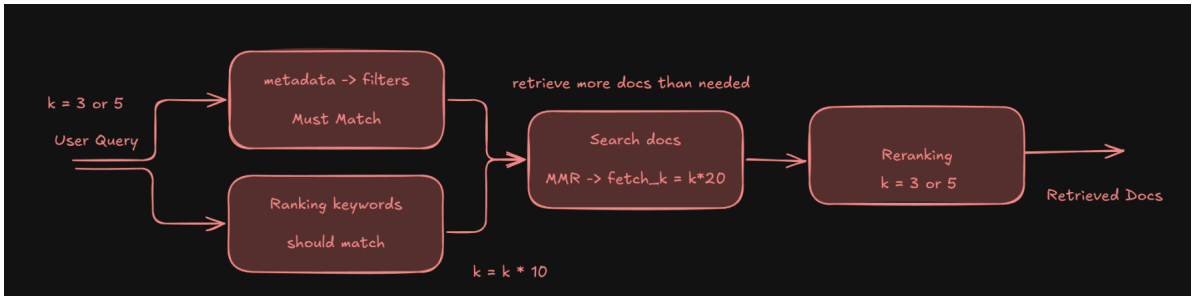
### 4.2.3 Keyword Generation

Mặc dù metadata giúp lọc tài liệu, nhưng để tăng độ chính xác, hệ thống cần sinh các từ khóa chuyên biệt từ thuật ngữ SEC filings. Mục tiêu của bước này là:

**Sinh 5 từ khóa liên quan đến truy vấn:** Các từ khóa được lựa chọn dựa trên heading, income statement, balance sheet và cash flows trong SEC filings, đảm bảo tính chính xác và liên quan.

**Hỗ trợ re-ranking:** Các từ khóa này giúp lọc page nội dung chi tiết, tăng khả năng LLM sinh câu trả lời chính xác và giảm hallucination.

### 4.2.4 Tìm kiếm tài liệu với Filters và Keywords




Sau khi có metadata và keywords, hệ thống xây dựng các tham số truy vấn để tìm kiếm tài liệu trong ChromaDB:

**Metadata Filters:** Lọc theo công ty, loại báo cáo, năm/quý tài chính để giới hạn phạm vi search.

**Content Filters:** Kiểm tra page có chứa ít nhất một trong các keywords đã sinh, đảm bảo page retrieve thực sự liên quan đến truy vấn.

**Số lượng kết quả top-k:** Tham số "k" xác định số lượng tài liệu trả về, giúp cân bằng giữa độ chính xác và hiệu năng.

Phương pháp retrieve sử dụng **MMR (Maximal Marginal Relevance)**, nhằm tránh lấy các page trùng lặp và tăng đa dạng trong kết quả.

- **Extract Headings và Content cho Re-Ranking:** Trong bước này, mỗi page tài liệu được tách thành các chunk dựa trên heading và nội dung liên quan. Hệ thống nhận diện các heading theo định dạng Markdown và lấy 1-2 đoạn văn ngay sau heading làm chunk. Việc này đảm bảo mỗi chunk vẫn giữ nguyên ngữ cảnh và thông tin quan trọng, chuẩn bị cho bước re-ranking chi tiết bằng BM25Plus.

- **Mục tiêu của Extract Headings:** Chia nhỏ page thành các chunk heading + content giúp BM25Plus đánh giá chi tiết mức độ liên quan của từng đoạn nội dung, thay vì đánh giá toàn bộ page. Điều này giảm thiểu rủi ro LLM sử dụng dữ liệu không liên quan, tăng tính chính xác và độ logic khi sinh câu trả lời, đồng thời chuẩn hóa dữ liệu để workflow retrieval hiệu quả hơn.

- **Ý nghĩa học thuật của Extract Headings:** Việc tách page thành heading + content chunks cho phép re-ranking đánh giá từng chủ đề nhỏ, từ đó xác định chính xác những thông tin liên quan đến query. Cách này giảm nguy cơ LLM bị hallucination và đảm bảo context truyền vào mô hình chứa dữ liệu hữu ích, nâng cao chất lượng câu trả lời và tính logic của hệ thống RAG.

- **BM25Plus Re-Ranking - Input:** Các Document objects đã retrieve từ vector store cùng danh sách ranking keywords chuyên biệt sinh ra từ query của người dùng. Tất cả chunks heading + content của các page này sẽ được chuyển thành token list, chuẩn hóa cho tính toán similarity và sắp xếp theo độ liên quan.

- **BM25Plus Re-Ranking - Quy trình:** Tạo BM25Plus model với corpus là các chunk từ page đã retrieve, tính score similarity giữa query keywords và chunk, sau đó sắp xếp top-k documents theo điểm số cao nhất. Quy trình này đảm bảo các page liên quan thực sự được lựa chọn để đưa vào LLM, giảm độ nhiễu thông tin và cải thiện độ chính xác.

- **Ý nghĩa học thuật của BM25Plus Re-Ranking:** Bước re-ranking giúp hệ thống chọn ra các page có độ liên quan cao nhất, giảm hallucination, tăng độ đầy đủ thông tin và tính logic trong câu trả lời. Đồng thời, phương pháp này cung cấp nền tảng vững chắc cho các mô hình nâng cao như Self-RAG, Reflexion-RAG và Adaptive-RAG, đảm bảo chất lượng context trước khi truyền vào LLM.
- **Tổng kết Workflow RAG với Re-Ranking:** Workflow tổng thể kết hợp metadata filtering, vector similarity và keyword-based re-ranking. Quy trình gồm: người dùng gửi query → trích xuất metadata → sinh 5 keyword → retrieve bằng MMR search → extract heading + content chunks → re-rank bằng BM25Plus → trả top-k page cho LLM. Hệ thống này tối ưu tốc độ, độ chính xác, giảm hallucination và tạo nền tảng vững chắc cho các ứng dụng nâng cao trong tài chính, pháp lý, y tế, học thuật và doanh nghiệp.

# 5: Enhanced Agentic PageRAG: Relevance Grading, Query Rewriting, and Cited Answers

## 5.1 Tổng quan

Enhanced Agentic PageRAG là một bước tiến quan trọng trong workflow của hệ thống Retrieval-Augmented Generation (RAG). Khác với Page-RAG tiêu chuẩn, hệ thống này không chỉ thực hiện retrieval và re-ranking mà còn tích hợp các bước agentic thông minh nhằm tối ưu độ chính xác, giảm hallucination và đảm bảo rằng các câu trả lời sinh ra được dựa trên dữ liệu thực tế và có trích dẫn rõ ràng.

Hình dưới minh họa quy trình hoạt động của Enhanced Agentic PageRAG. Toàn bộ workflow này diễn ra sau khi các page-level chunks đã được ingest, embedding và re-ranked bằng BM25Plus, đồng thời metadata đã được xác định rõ ràng.

## 5.2 Các thành phần chính của Enhanced Agentic PageRAG

- **Document Relevance Grading:** Trong bước này, LLM đánh giá mức độ liên quan của từng page được retrieve so với query của người dùng. Việc grading này giúp hệ thống loại bỏ các page không liên quan trước khi sinh câu trả lời, giảm tối đa rủi ro hallucination. Đồng thời, cơ chế này đảm bảo rằng chỉ những thông tin thực sự hữu ích mới được đưa vào context, từ đó nâng cao độ chính xác và tin cậy của câu trả lời cuối cùng.
- **Query Rewriting:** Nếu Document Relevance Grading nhận thấy dữ liệu retrieve chưa đủ để trả lời query, hệ thống tự động rewrite query ban đầu. Quá trình rewrite tập trung vào việc sử dụng các từ khóa chuyên biệt, ví dụ từ báo cáo SEC filings hoặc metadata tài liệu, để gửi lại retrieval. Cơ chế này giúp cải thiện khả năng retrieve các page thực sự liên quan, đồng thời tăng tính linh hoạt của hệ thống khi xử lý các truy vấn phức tạp hoặc không rõ ràng từ người dùng.
- **Single Retry Cycle:** Để tránh vòng lặp vô hạn trong quá trình query rewriting, hệ thống chỉ cho phép một lần retry. Điều này đảm bảo rằng workflow không bị treo hoặc quá tải, đồng thời duy trì hiệu quả tính toán. Mặc dù chỉ retry một lần, cơ chế này vẫn giúp tăng đáng kể chất lượng dữ liệu context được sử dụng cho LLM.
- **Cited Answers:** Sau khi query được đánh giá và retrieve dữ liệu liên quan, LLM sinh câu trả lời kèm trích dẫn. Các citation được format theo markdown và liên kết trực tiếp đến metadata của page, ví dụ: tên công ty, loại tài liệu, năm tài chính hoặc section. Điều này không chỉ giúp người dùng kiểm chứng thông tin mà còn nâng cao tính minh bạch và tin cậy của hệ thống.

## 5.3 Ý nghĩa học thuật và ứng dụng thực tế

- **Tăng độ chính xác retrieval:** Kết hợp giữa Document Relevance Grading, query rewriting và metadata filtering giúp hệ thống chỉ lấy các page có khả năng trả lời query thực sự.

Điều này giảm thiểu việc LLM xử lý thông tin không liên quan, đồng thời tăng hiệu quả xử lý với dữ liệu lớn, có thể lên tới hàng triệu page.

- **Giảm hallucination:** Các cơ chế grading và re-ranking đảm bảo rằng chỉ các chunk nội dung thực sự phù hợp mới được đưa vào context, hạn chế tối đa việc LLM tạo thông tin giả hoặc thiếu căn cứ.
- **Cải thiện trải nghiệm người dùng:** Cited Answers cung cấp thông tin minh bạch, dễ tra cứu nguồn gốc dữ liệu, giúp người dùng nhanh chóng xác nhận và sử dụng thông tin trả về. Hệ thống cũng có khả năng rewrite query tự động, giúp người dùng không cần tinh chỉnh query quá nhiều.
- **Tính khả dụng với dữ liệu chuyên biệt:** Workflow này đặc biệt hiệu quả với các loại tài liệu có cấu trúc và metadata rõ ràng, như báo cáo SEC filings, hợp đồng pháp lý, hồ sơ y tế hay nghiên cứu học thuật. Metadata giúp lọc, keyword giúp re-ranking, và cited answer đảm bảo tính minh bạch.

## 5.4 Ứng dụng trong thực tế

- **Tài chính:** Trong phân tích SEC filings (10-K, 10-Q), hệ thống có thể tự động detect company, fiscal year, section, và chỉ retrieve các page liên quan. Query rewriting giúp người dùng tra cứu các thông tin tài chính phức tạp mà không cần phải biết trước cấu trúc báo cáo.
- **Pháp lý:** Với hợp đồng và hồ sơ case law, PageRAG agentic cho phép filter theo jurisdiction, clause hoặc thời gian, từ đó cung cấp trích dẫn chính xác khi người dùng tra cứu nội dung cụ thể.
- **Y tế:** Hệ thống có thể truy xuất patient records theo date, department, provider, đảm bảo tính riêng tư và tính chính xác khi đưa vào context cho LLM.
- **Doanh nghiệp:** Knowledge base nội bộ có thể được truy xuất nhanh chóng dựa trên metadata dự án, department, hoặc category, đồng thời trả về câu trả lời kèm citation giúp nhân viên dễ dàng kiểm chứng thông tin.
- **Học thuật:** Các nghiên cứu, thesis hay paper có thể được index theo author, year, topic, section, từ đó sinh câu trả lời logic với citation rõ ràng, hỗ trợ nghiên cứu và tổng hợp dữ liệu học thuật.

## 5.5 Kết luận

Enhanced Agentic PageRAG là bước tiến quan trọng trong phát triển hệ thống RAG nâng cao, tích hợp giữa retrieval, relevance grading, query rewriting và cited answer generation.

- Hệ thống đảm bảo rằng chỉ những page thực sự liên quan mới được đưa vào context, tăng độ chính xác và giảm hallucination.
- Cited Answers cung cấp minh bạch dữ liệu, giúp người dùng kiểm chứng, đồng thời tạo nền tảng cho các hệ thống nâng cao như Self-RAG, Reflexion-RAG và Adaptive-RAG.
- Workflow này đặc biệt phù hợp với các tập dữ liệu lớn, có cấu trúc rõ ràng và metadata phong phú, từ tài chính, pháp lý, y tế, doanh nghiệp đến nghiên cứu học thuật.
- Kết hợp giữa vector similarity, metadata filtering, BM25Plus re-ranking và agentic logic tạo ra một hệ thống AI thông minh, tự phản hồi và học hỏi, sẵn sàng triển khai trong các ứng dụng thực tế quy mô lớn.

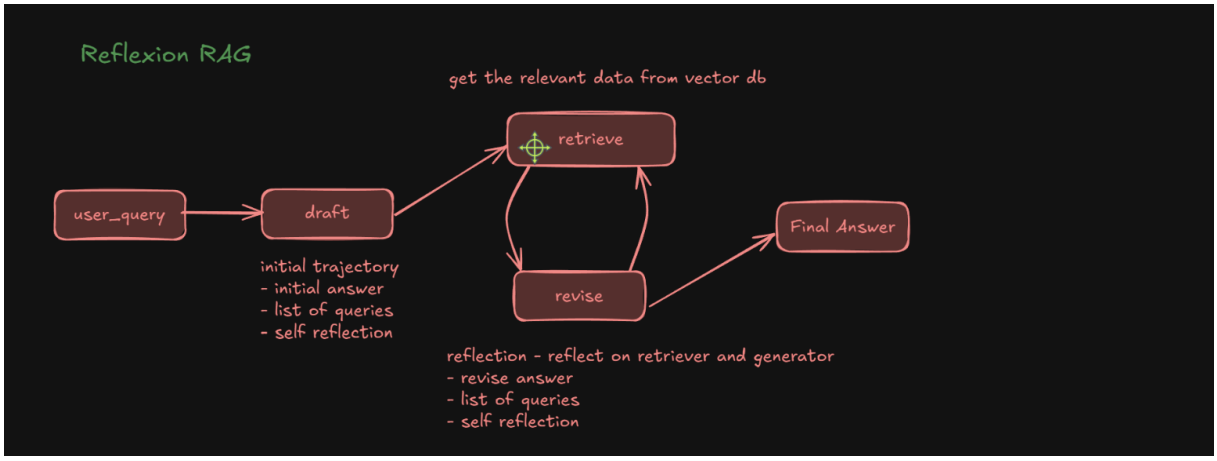
# 6: Reflexion Agentic RAG: Self-Improving Retrieval-Augmented Generation

## 6.1 Giới thiệu

Reflexion Agentic RAG là một hệ thống RAG tiên tiến, được thiết kế dựa trên nguyên tắc *iterative refinement* (tinh chỉnh lặp đi lặp lại). Khác với Page-RAG cơ bản, hệ thống này không chỉ truy xuất dữ liệu từ vector database mà còn tự đánh giá, sửa đổi và hoàn thiện câu trả lời dựa trên thông tin đã thu thập được. Mục tiêu là giảm tình trạng hallucination, tăng độ chính xác và đảm bảo câu trả lời logic, đầy đủ, đồng thời tạo nền tảng cho các hệ thống nâng cao như Adaptive RAG.

## 6.2 Workflow Reflexion Agentic RAG

Hình dưới minh họa workflow của Reflexion Agentic RAG, từ khi người dùng gửi query đến khi hệ thống trả về câu trả lời đã refine kèm trích dẫn và metadata đầy đủ.




Hình 2: Sơ đồ workflow của Reflexion Agentic RAG

### 6.2.1 Draft Initial Answer

**Mục tiêu:** Hệ thống tạo một bản nháp câu trả lời dựa trên query người dùng và dữ liệu hiện có trong vector store. Bản nháp này giúp xác định các phần thông tin còn thiếu hoặc chưa chính xác, từ đó chuẩn bị cho các bước retrieve và refine tiếp theo.

**Chi tiết:** LLM nhận query đầu vào và sinh nội dung dựa trên embedding hiện có, đồng thời đánh giá sơ bộ mức độ đầy đủ của thông tin. Các đoạn nháp này có thể chứa nhiều khoảng trống dữ liệu hoặc thiếu thông tin chi tiết từ tài liệu gốc.

**Ý nghĩa học thuật:** Bước này cung cấp cơ sở để hệ thống Reflexion có thể xác định những khoảng trống thông tin, tạo ra các query phụ trợ nhằm retrieve tài liệu bổ sung, từ đó nâng cao chất lượng output.

### 6.2.2 Document Retrieval

**Mục tiêu:** Từ bản nháp câu trả lời, hệ thống xác định các page có khả năng chứa thông tin cần thiết và retrieve từ vector database. Các kỹ thuật metadata filtering và BM25Plus/ MMR search được áp dụng nhằm đảm bảo chỉ các page liên quan thực sự được đưa vào context.

**Chi tiết:** Metadata như tên công ty, loại tài liệu, năm tài chính, quý được sử dụng để lọc dữ liệu. Sau đó, các từ khóa liên quan đến SEC filings hoặc chủ đề truy vấn được áp dụng để re-rank và chọn lọc các page phù hợp.

**Ý nghĩa học thuật:** Việc kết hợp metadata filtering và content similarity giúp giảm khối lượng page phải xử lý, tăng tốc độ truy xuất, đồng thời giảm nguy cơ LLM nhận dữ liệu không liên quan dẫn tới hallucination.

### 6.2.3 Answer Revision and Reflection

**Mục tiêu:** Sử dụng các page retrieved, LLM sửa đổi bản nháp, bổ sung thông tin và đánh giá tính đầy đủ của câu trả lời. Nếu các phần thông tin quan trọng còn thiếu, hệ thống sẽ chuẩn bị query mới để retrieve thêm dữ liệu.

**Chi tiết:** LLM đánh giá các đoạn nội dung đã retrieve, phát hiện chỗ chưa đầy đủ hoặc thông tin mâu thuẫn. Hệ thống tự sinh các truy vấn phụ trợ (query rewriting) để bổ sung tài liệu cần thiết, đảm bảo câu trả lời cuối cùng logic và chính xác.

**Ý nghĩa học thuật:** Reflexion Agentic RAG học hỏi trong quá trình sinh câu trả lời, thực hiện *self-reflection* để cải thiện chất lượng output. Điều này giúp nâng cao độ chính xác, giảm hallucination và tăng tính toàn diện cho câu trả lời cuối cùng.

### 6.2.4 Iterative Refinement

**Mục tiêu:** Cho phép hệ thống lặp lại quá trình retrieve và refine cho đến khi câu trả lời đầy đủ hoặc đạt số vòng lặp tối đa. Điều này khác biệt với Page-RAG truyền thống, giúp hệ thống tự động học hỏi và điều chỉnh kết quả.

**Chi tiết:** Nếu câu trả lời chưa hoàn thiện, hệ thống rewrite query, retrieve thêm page bổ sung và update lại câu trả lời. Mỗi vòng lặp đều sử dụng metadata, từ khóa và BM25Plus để đảm bảo chất lượng thông tin.

**Ý nghĩa học thuật:** Iterative refinement giúp hệ thống tự nâng cao chất lượng câu trả lời một cách tự động. Điều này tạo ra một vòng phản hồi liên tục, cải thiện khả năng xử lý các truy vấn phức tạp và giảm thiểu rủi ro thiếu thông tin.

### 6.2.5 Cited Answers và Keyword Ranking

**Mục tiêu:** Cung cấp câu trả lời đã refine kèm trích dẫn inline, các từ khóa chuyên biệt từ domain (financial keywords) và metadata đầy đủ. Điều này giúp người dùng kiểm chứng thông tin và hiểu rõ nguồn gốc dữ liệu.

**Chi tiết:** Hệ thống đánh dấu inline citations với company, section, year, và sử dụng các từ khóa SEC filings để đảm bảo dữ liệu retrieved thực sự liên quan. Câu trả lời được trình bày theo định dạng markdown hoặc HTML để thuận tiện cho việc hiển thị và đọc.

**Ý nghĩa học thuật:** Việc kết hợp cited answers và keyword ranking không chỉ nâng cao độ chính xác mà còn tạo ra tính minh bạch trong việc sử dụng dữ liệu. Người dùng có thể truy xuất lại nguồn thông tin, từ đó tăng tính đáng tin cậy của hệ thống.

### 6.2.6 Tổng kết Reflexion Agentic RAG Workflow

**Ý nghĩa tổng quan học thuật:** Workflow Reflexion Agentic RAG kết hợp metadata filtering, vector similarity, keyword-based re-ranking và iterative refinement. Điều này đảm bảo rằng LLM chỉ nhận thông tin thực sự liên quan, giảm hallucination, tối ưu tốc độ và độ chính xác retrieval, đồng thời tạo nền tảng vững chắc cho các hệ thống nâng cao như Self-RAG, Adaptive RAG và các agent AI tự phản hồi khác.

# 7: Self-Reflective RAG (Self-RAG): Hệ thống Tự Phản Chiếu Retrieval-Augmented Generation

## 7.1 Giới thiệu

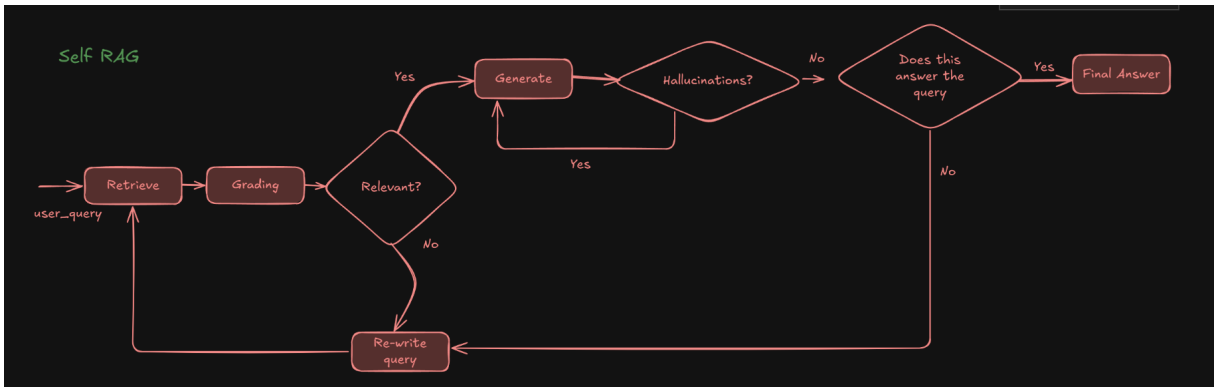
Self-RAG là một biến thể nâng cao của Retrieval-Augmented Generation, được thiết kế nhằm tăng khả năng tự đánh giá, tự học hỏi và tự cải thiện chất lượng câu trả lời của hệ thống AI. Khác với Page-RAG và Reflexion Agentic RAG, Self-RAG không chỉ dựa vào retrieval thông thường và iterative refinement mà còn thực hiện các bước tự kiểm tra tính đầy đủ của câu trả lời, phát hiện hallucination, và tự rewrite query để đảm bảo retrieval luôn chính xác. Hệ thống này tối ưu cho các tập dữ liệu lớn, đa dạng và phức tạp, đặc biệt phù hợp với các trường hợp tài chính, pháp lý, y tế, và nghiên cứu học thuật.

Self-RAG kết hợp nhiều cơ chế tiên tiến:

1. Document relevance grading: Xác định page hoặc chunk có liên quan nhất trước khi đưa vào context.
2. Hallucination detection: Phát hiện các thông tin LLM sinh ra không dựa trên dữ liệu thực tế.
3. Answer completeness checking: Đánh giá độ đầy đủ của câu trả lời dựa trên yêu cầu truy vấn.
4. Query rewriting: Tự động tạo truy vấn mới nếu dữ liệu retrieve chưa đầy đủ.

## 7.2 Workflow tổng quan

Hình dưới minh họa toàn bộ workflow Self-RAG, từ khi nhận query của người dùng đến khi trả về câu trả lời đã được kiểm tra và refine hoàn chỉnh.




Hình 3: Sơ đồ tổng quan workflow của Self-RAG

Workflow này có thể được tóm tắt như sau:

1. Người dùng gửi query ban đầu.
2. Hệ thống retrieve các document phù hợp từ vector store dựa trên metadata filtering và keyword-based search.
3. Thực hiện document relevance grading để xác định các page quan trọng.
4. Sinh câu trả lời nháp dựa trên thông tin retrieved.
5. Kiểm tra hallucination và đánh giá độ đầy đủ của câu trả lời.
6. Nếu câu trả lời chưa hoàn thiện hoặc chứa hallucination, hệ thống rewrite query và retrieve thêm page bổ sung.
7. Lặp lại quá trình refine cho đến khi câu trả lời đạt yêu cầu hoặc đạt số vòng lặp tối đa.
8. Trả về câu trả lời cuối cùng kèm trích dẫn và metadata.

## 7.3 Document Relevance Grading

**Mục tiêu:** Xác định mức độ liên quan của từng page hoặc chunk trước khi đưa vào context, nhằm giảm nguy cơ LLM nhận dữ liệu không phù hợp và sinh hallucination. Việc này đảm bảo các tài liệu được đưa vào context thực sự có khả năng trả lời câu hỏi.

**Chi tiết:** Hệ thống sử dụng vector similarity kết hợp với các thuật toán re-ranking như BM25Plus hoặc MMR. Metadata filtering cũng được áp dụng để giới hạn phạm vi tài liệu, ví dụ theo công ty, loại báo cáo, năm tài chính, hoặc section cụ thể.

**Ý nghĩa học thuật:** Document relevance grading là bước nền tảng, giúp Self-RAG phân loại dữ liệu retrieved theo mức độ quan trọng, từ đó tăng hiệu quả và độ chính xác của các bước generate và refine tiếp theo.

### 7.4 Hallucination Detection

**Mục tiêu:** Hallucination detection trong Self-RAG hướng đến việc phát hiện những thông tin mà LLM sinh ra *không dựa trên dữ liệu thực tế đã được retrieve*. Trong các domain nhạy cảm như tài chính, pháp lý, y tế hay nghiên cứu khoa học, tính chính xác là yếu tố sống còn. Việc phát hiện kịp thời các hallucination không chỉ giúp nâng cao độ tin cậy của hệ thống mà còn giúp người dùng có thể dựa vào kết quả để ra quyết định một cách an toàn và minh bạch.

**Chi tiết:** Hệ thống Hallucination Detection sử dụng nhiều cơ chế phối hợp để kiểm tra độ tin cậy của câu trả lời:

- So sánh các thông tin sinh ra với các *metadata* và *page-level content chunks* đã được retrieve. Việc này đảm bảo rằng mọi dữ liệu xuất hiện trong câu trả lời đều có thể trace được về nguồn gốc thực tế.
- Kiểm tra trích dẫn inline: mỗi thông tin quan trọng phải có link hoặc citation trỏ về page hoặc section tương ứng, giúp người dùng dễ dàng xác minh.
- Phân tích keyword và semantic matching: hệ thống so sánh các keyword quan trọng trong câu trả lời với keyword và nội dung trong chunks. Nếu một phần thông tin không khớp với bất kỳ chunk nào, nó được đánh dấu là khả năng hallucination.
- Cross-check logic: ngoài việc kiểm tra nội dung, Self-RAG còn đánh giá tính logic và consistency giữa các phần thông tin trong câu trả lời để phát hiện những mâu thuẫn hoặc dữ liệu phi thực tế.

**Ý nghĩa học thuật:** Cơ chế này nâng cao tính đáng tin cậy của Self-RAG, giảm thiểu rủi ro sai lệch thông tin trong các domain quan trọng. Nó cho phép LLM sinh câu trả lời không chỉ dựa trên sự sáng tạo ngôn ngữ mà còn dựa trên dữ liệu thực tế có thể kiểm chứng. Đồng thời, việc Hallucination Detection còn là cơ sở để thực hiện các bước tiếp theo như query rewriting hoặc re-retrieval, tạo thành một vòng lặp học tự động giúp hệ thống cải thiện độ chính xác liên tục.

### 7.5 Answer Completeness Checking

**Mục tiêu:** Answer Completeness Checking trong Self-RAG nhằm đảm bảo câu trả lời được sinh ra không chỉ chính xác mà còn toàn diện, bao quát tất cả khía cạnh quan trọng mà truy vấn của người dùng yêu cầu. Trong nhiều tình huống, một câu trả lời thiếu thông tin có thể dẫn đến quyết định sai lầm, đặc biệt trong các domain như tài chính, y tế, pháp lý. Vì vậy, cơ chế này đảm bảo Self-RAG sinh ra nội dung đầy đủ, không bỏ sót phần quan trọng.

**Chi tiết:** Self-RAG thực hiện kiểm tra tính đầy đủ của câu trả lời thông qua nhiều bước:

- Kiểm tra sự xuất hiện của các *keyword quan trọng* liên quan trực tiếp đến truy vấn. Hệ thống đánh dấu nếu một số keyword không xuất hiện trong câu trả lời.
- Đánh giá coverage của các *page-level chunks* hoặc sections đã retrieve. Nếu câu trả lời bỏ qua các chunk quan trọng, hệ thống xác định rằng thông tin còn thiếu.
- Đánh dấu các phần thiếu: Khi phát hiện bất kỳ thiếu sót nào, Self-RAG sẽ gắn cờ để thực hiện bước *query rewriting*, nghĩa là hệ thống tự động sinh lại truy vấn dựa trên các phần bị bỏ sót để retrieve thêm thông tin cần thiết.
- Loop kiểm tra: Sau khi thực hiện query rewriting và retrieve bổ sung, hệ thống sẽ tổng hợp lại câu trả lời, tiếp tục kiểm tra tính đầy đủ cho đến khi tất cả khía cạnh của truy vấn được bao quát hoặc đạt giới hạn vòng lặp.

**Ý nghĩa học thuật:** Answer Completeness Checking là một bước kiểm soát chất lượng quan trọng trong Self-RAG, đảm bảo rằng câu trả lời không chỉ dựa trên một phần dữ liệu retrieved mà là tổng hợp đầy đủ thông tin từ nhiều page-level chunks. Việc thực hiện query rewriting dựa trên các phần thiếu là một cơ chế học chủ động: hệ thống tự động nhận diện lỗ hổng thông tin

và thực hiện truy vấn bổ sung để hoàn thiện câu trả lời. Điều này giúp Self-RAG khác biệt so với RAG truyền thống, nơi LLM có thể chỉ sinh nội dung dựa trên một số page retrieved mà không đánh giá tính toàn diện. Nhờ cơ chế này, Self-RAG vừa nâng cao độ chính xác, vừa đảm bảo độ đầy đủ, cung cấp câu trả lời logic, đáng tin cậy và có khả năng kiểm chứng.

### 7.6 Query Rewriting for Better Retrieval

**Mục tiêu:** Query Rewriting là cơ chế cho phép Self-RAG tự động tạo ra các truy vấn bổ sung nếu dữ liệu đã retrieve trước đó chưa đủ để trả lời toàn diện câu hỏi của người dùng. Việc này nhằm đảm bảo rằng tất cả các page hoặc chunk quan trọng liên quan đến truy vấn được tìm thấy và đưa vào context trước khi LLM sinh câu trả lời cuối cùng. Đây là bước chủ động, giúp hệ thống không chỉ dựa vào dữ liệu hiện có mà còn học hỏi và cải thiện retrieval theo thời gian thực.

**Chi tiết:** Khi Self-RAG phát hiện các phần thông tin còn thiếu hoặc câu trả lời nháp không bao quát tất cả khía cạnh cần thiết, hệ thống sẽ tự sinh ra một truy vấn mới (query phụ trợ) dựa trên những khoảng trống này. Truy vấn mới này được gửi lại đến vector store hoặc cơ sở dữ liệu thích hợp để retrieve thêm các tài liệu bổ sung. Cơ chế này thường kết hợp với metadata filtering và ranking keywords để tăng khả năng tìm thấy chính xác các page cần thiết. Kỹ thuật query rewriting giúp hệ thống thích nghi với từng câu hỏi cụ thể, tối ưu hoá retrieval và đảm bảo rằng LLM sẽ có context đầy đủ trước khi sinh nội dung.

**Ý nghĩa học thuật:** Query Rewriting làm tăng khả năng self-correction (tự điều chỉnh) của Self-RAG. Hệ thống trở nên chủ động, không chỉ dựa vào dữ liệu hiện có mà còn nhận diện các khoảng trống và cải thiện retrieval liên tục. Điều này đặc biệt quan trọng trong các domain phức tạp như tài chính, y tế hay pháp lý, nơi một thông tin bị bỏ sót có thể dẫn đến hậu quả nghiêm trọng. Ngoài ra, query rewriting còn giúp Self-RAG học cách tối ưu retrieval cho nhiều loại truy vấn khác nhau mà không cần can thiệp thủ công, nâng cao khả năng tự động hóa.

### 7.7 Iterative Refinement Loop

**Mục tiêu:** Iterative Refinement Loop là cơ chế lặp lại toàn bộ quá trình retrieve, grade, generate, kiểm tra hallucination và query rewriting nhiều lần cho đến khi câu trả lời cuối cùng đạt yêu cầu về tính chính xác và đầy đủ, hoặc đạt số vòng lặp tối đa được định nghĩa trước. Mục tiêu là đảm bảo rằng câu trả lời cuối cùng không bỏ sót thông tin quan trọng, đồng thời tối ưu chất lượng dữ liệu context mà LLM sử dụng.

**Chi tiết:** Mỗi vòng lặp bao gồm các bước chính:

1. Retrieve document dựa trên metadata filtering và ranking keywords để lấy các page hoặc chunk liên quan đến truy vấn. Đây là bước đảm bảo dữ liệu cơ sở là chính xác và phù hợp với mục đích câu hỏi.

2. Xếp hạng page/chunk bằng BM25Plus hoặc MMR nhằm sắp xếp các tài liệu theo độ liên quan, loại bỏ các thông tin trùng lặp và tăng độ đa dạng trong kết quả.

3. Sinh câu trả lời nháp bằng LLM dựa trên các page/chunk đã retrieve và ranked.

4. Kiểm tra hallucination để phát hiện các thông tin không có trong dữ liệu retrieved và kiểm tra answer completeness để xác định các phần thông tin còn thiếu.

5. Rewrite query nếu cần, tạo truy vấn phụ trợ dựa trên các khoảng trống đã phát hiện, và gửi lại để retrieve thêm tài liệu bổ sung.

**Ý nghĩa học thuật:** Iterative Refinement Loop là cơ chế học chủ động, giúp Self-RAG liên tục cải thiện câu trả lời bằng cách kết hợp retrieval, re-ranking và query rewriting. Bằng cách lặp lại quy trình, hệ thống giảm thiểu rủi ro thông tin thiếu hoặc sai lệch, tối ưu hóa độ chính xác và đảm bảo tính toàn diện. Đây là một bước quan trọng, giúp Self-RAG vượt trội so với RAG truyền thống, nơi LLM thường chỉ dựa vào một lượt retrieval duy nhất và không đánh giá tính đầy đủ hoặc độ tin cậy của dữ liệu.

### 7.8 Cited Answers và Metadata

**Mục tiêu:** Cited Answers và Metadata cung cấp khả năng minh bạch và traceability cho Self-RAG. Mục tiêu là trả về câu trả lời kèm thông tin trích dẫn chi tiết và metadata đầy đủ, cho phép người dùng kiểm chứng nguồn dữ liệu, hiểu rõ căn cứ thông tin, và sử dụng thông tin này trong các phân tích hoặc quyết định quan trọng.

**Chi tiết:** Các câu trả lời được sinh ra bởi LLM đi kèm với các trích dẫn inline, bao gồm các thông tin quan trọng như tên công ty, section trong tài liệu, năm tài chính, và các keyword chủ chốt liên quan đến domain (ví dụ trong SEC filings). Metadata đi kèm chứa thông tin bổ sung về document, page, section, và nguồn dữ liệu. Cấu trúc này cho phép người dùng dễ dàng xác minh dữ liệu retrieved và hỗ trợ các agent khác hoặc các bước xử lý tiếp theo nếu cần truy vấn bổ sung.

**Ý nghĩa học thuật:** Việc tích hợp cited answers và metadata giúp nâng cao độ tin cậy và minh bạch của Self-RAG. Người dùng và các hệ thống downstream có thể kiểm chứng nguồn dữ liệu, xác nhận tính chính xác và đầy đủ của câu trả lời. Trong các domain nhạy cảm, điều này cực kỳ quan trọng để giảm rủi ro quyết định sai lầm dựa trên thông tin không được xác minh. Đồng thời, cơ chế này tạo điều kiện thuận lợi để tích hợp Self-RAG vào các hệ thống lớn hơn, như Adaptive RAG hoặc Reflexion RAG, nơi dữ liệu phải liên tục được truy xuất, đánh giá và cập nhật.

### 7.9 Tổng kết Self-RAG Workflow

Self-RAG kết hợp:

- Metadata filtering và vector similarity để truy xuất chính xác các page/chunk.
- Keyword-based re-ranking (BM25Plus, MMR) để nâng cao chất lượng retrieval.
- Self-reflection mechanism để kiểm tra hallucination, đánh giá completeness và rewrite query nếu cần.
- Iterative refinement giúp câu trả lời cuối cùng logic, đầy đủ và có trích dẫn.

**Ý nghĩa tổng quan học thuật:** Self-RAG là bước tiến lớn so với RAG truyền thống và Page-RAG cơ bản. Hệ thống không chỉ retrieve dữ liệu mà còn tự học hỏi, tự kiểm tra và tự điều chỉnh câu trả lời. Điều này đặc biệt quan trọng trong các domain phức tạp, yêu cầu độ chính xác cao và khả năng tổng hợp dữ liệu từ nhiều tài liệu. Self-RAG đặt nền tảng cho các hệ thống agent AI tự phản hồi, thích nghi và nâng cao chất lượng trả lời theo thời gian thực.

## 8: Adaptive RAG: Learning to Navigate Through Knowledge Base

Adaptive RAG là một phương pháp tiên tiến trong Retrieval-Augmented Generation, được thiết kế nhằm tối ưu hóa việc truy xuất dữ liệu dựa trên phân tích thông minh ý định truy vấn (query intent). Không giống như các mô hình RAG truyền thống hoặc Page-RAG cơ bản, Adaptive RAG không thực hiện một luồng truy xuất duy nhất. Thay vào đó, hệ thống quyết định đường đi tối ưu của truy vấn dựa trên loại thông tin yêu cầu, từ đó lựa chọn phương thức retrieval phù hợp nhất.

Adaptive RAG kết hợp nhiều workflow retrieval khác nhau, bao gồm Self-RAG, Reflexion-RAG hoặc các retrieval agent đặc thù, giúp hệ thống linh hoạt, chủ động và tự phản hồi theo từng loại truy vấn cụ thể. Việc này đặc biệt quan trọng trong các domain phức tạp, nơi dữ liệu đa nguồn, nhiều định dạng và yêu cầu độ chính xác cao.

### 8.1 Mục tiêu và Ý tưởng cơ bản

**Mục tiêu:** Adaptive RAG nhằm tăng hiệu quả, tốc độ và độ chính xác của hệ thống RAG bằng cách xác định loại dữ liệu phù hợp với truy vấn trước khi thực hiện retrieval. Thay vì gửi

query đồng loạt tới tất cả các nguồn dữ liệu, hệ thống sẽ phân tích nội dung, intent, và ngữ cảnh của query để chọn đường đi tối ưu, giúp tiết kiệm tài nguyên tính toán và tăng khả năng trả lời chính xác ngay từ lần truy xuất đầu tiên.

**Ý tưởng cơ bản:**

- Mỗi query được phân tích để xác định loại thông tin cần truy xuất, ví dụ tài chính, dữ liệu nội bộ, hoặc kiến thức chung.
- Hệ thống sử dụng bộ quy tắc logic hoặc mô hình học máy để phân loại query và lựa chọn routing path phù hợp.
- Mỗi đường đi có phương pháp truy xuất đặc thù: vector store cho tài liệu page-level, SQL agent cho cơ sở dữ liệu có cấu trúc, hoặc web search cho kiến thức chung.
- Adaptive RAG tận dụng khả năng kết hợp giữa retrieval agent, metadata filtering, keyword ranking và iterative refinement để tối ưu hóa kết quả.
- Hệ thống không chỉ retrieve dữ liệu mà còn đánh giá chất lượng, độ đầy đủ và mức độ liên quan của các page/chunk trước khi gửi vào LLM, đảm bảo câu trả lời logic và toàn diện.

## 8.2 Routing Paths trong Adaptive RAG

Adaptive RAG định nghĩa ba đường đi chính, mỗi đường đi ứng với một loại dữ liệu và phương pháp retrieval khác nhau:

1. **Financial Documents (Vectorstore):** Truy vấn liên quan đến báo cáo tài chính, SEC filings, hoặc dữ liệu công ty được gửi vào vector store. Hệ thống áp dụng Self-RAG flow, bao gồm các bước: metadata filtering, ranking bằng BM25 hoặc MMR, document relevance grading, hallucination detection, và iterative refinement. Mục tiêu là đảm bảo LLM nhận được các page có độ liên quan cao nhất, với keyword chuyên biệt và trích dẫn đầy đủ, đặc biệt quan trọng trong phân tích báo cáo tài chính phức tạp, giúp giảm rủi ro hallucination và sai sót thông tin. Việc lựa chọn vectorstore làm đường đi chính cho dữ liệu tài chính cũng tối ưu hóa tốc độ truy xuất nhờ vào khả năng tìm kiếm similarity vector, đồng thời tích hợp dễ dàng với các cơ chế Self-RAG và Reflexion-RAG.
2. **Employee Database (SQL Database):** Truy vấn liên quan đến dữ liệu nội bộ của doanh nghiệp như thông tin nhân viên, bảng lương, dự án, v.v. được xử lý bởi SQL agent. Agent này trực tiếp thực hiện câu lệnh SQL, trích xuất dữ liệu chính xác từ cơ sở dữ liệu, sau đó cung cấp kết quả cho LLM để sinh câu trả lời logic. Phương pháp này đảm bảo dữ liệu nhạy cảm hoặc có cấu trúc quan hệ được truy xuất một cách chính xác, tránh việc LLM dựa trên vector similarity mà có thể dẫn tới thông tin sai lệch. SQL agent cũng hỗ trợ các tính năng như kiểm tra quyền truy cập, đảm bảo compliance, và kết hợp với metadata của từng bảng để lọc dữ liệu trước khi sinh câu trả lời.
3. **General Knowledge (Web Search):** Nếu truy vấn yêu cầu kiến thức chung hoặc thông tin cập nhật từ internet, Adaptive RAG sẽ kích hoạt web search agent. Agent này thực hiện tìm kiếm trên web hoặc cơ sở dữ liệu kiến thức công khai, tập hợp các tài liệu có liên quan, và cung cấp dữ liệu này cho LLM để sinh câu trả lời. Cơ chế này đảm bảo hệ thống không giới hạn ở dữ liệu nội bộ hoặc vectorstore, mà vẫn có khả năng cung cấp thông tin cập nhật, tổng hợp từ nhiều nguồn công khai, và giảm rủi ro thiếu thông tin khi giải quyết các query mở. Web search agent cũng có khả năng lọc dữ liệu theo domain, kiểm tra độ tin cậy của nguồn và ưu tiên trích dẫn từ các nguồn đáng tin cậy.

## 8.3 Adaptive RAG: Learning to Navigate Through Knowledge Base

Adaptive RAG là một phương pháp tiên tiến trong Retrieval-Augmented Generation, được thiết kế nhằm tối ưu hóa việc truy xuất dữ liệu dựa trên phân tích thông minh ý định truy vấn (query intent). Không giống như các mô hình RAG truyền thống hoặc Page-RAG cơ bản, Adaptive RAG không thực hiện một luồng truy xuất duy nhất. Thay vào đó, hệ thống quyết

định đường đi tối ưu của truy vấn dựa trên loại thông tin yêu cầu, từ đó lựa chọn phương thức retrieval phù hợp nhất.

Adaptive RAG kết hợp nhiều workflow retrieval khác nhau, bao gồm Self-RAG, Reflexion-RAG hoặc các retrieval agent đặc thù, giúp hệ thống linh hoạt, chủ động và tự phản hồi theo từng loại truy vấn cụ thể. Việc này đặc biệt quan trọng trong các domain phức tạp, nơi dữ liệu đa nguồn, nhiều định dạng và yêu cầu độ chính xác cao.

### 8.3.1 Mục tiêu và Ý tưởng cơ bản

**Mục tiêu:** Adaptive RAG nhằm tăng hiệu quả, tốc độ và độ chính xác của hệ thống RAG bằng cách xác định loại dữ liệu phù hợp với truy vấn trước khi thực hiện retrieval. Thay vì gửi query đồng loạt tới tất cả các nguồn dữ liệu, hệ thống sẽ phân tích nội dung, intent, và ngữ cảnh của query để chọn đường đi tối ưu, giúp tiết kiệm tài nguyên tính toán và tăng khả năng trả lời chính xác ngay từ lần truy xuất đầu tiên.

**Ý tưởng cơ bản:**

- Mỗi query được phân tích để xác định loại thông tin cần truy xuất, ví dụ tài chính, dữ liệu nội bộ, hoặc kiến thức chung.
- Hệ thống sử dụng bộ quy tắc logic hoặc mô hình học máy để phân loại query và lựa chọn routing path phù hợp.
- Mỗi đường đi có phương pháp truy xuất đặc thù: vector store cho tài liệu page-level, SQL agent cho cơ sở dữ liệu có cấu trúc, hoặc web search cho kiến thức chung.
- Adaptive RAG tận dụng khả năng kết hợp giữa retrieval agent, metadata filtering, keyword ranking và iterative refinement để tối ưu hóa kết quả.
- Hệ thống không chỉ retrieve dữ liệu mà còn đánh giá chất lượng, độ đầy đủ và mức độ liên quan của các page/chunk trước khi gửi vào LLM, đảm bảo câu trả lời logic và toàn diện.

### 8.3.2 Routing Paths trong Adaptive RAG

Adaptive RAG định nghĩa ba đường đi chính, mỗi đường đi ứng với một loại dữ liệu và phương pháp retrieval khác nhau:

1. **Financial Documents (Vectorstore):** Truy vấn liên quan đến báo cáo tài chính, SEC filings, hoặc dữ liệu công ty được gửi vào vector store. Hệ thống áp dụng Self-RAG flow, bao gồm các bước: metadata filtering, ranking bằng BM25 hoặc MMR, document relevance grading, hallucination detection, và iterative refinement. Mục tiêu là đảm bảo LLM nhận được các page có độ liên quan cao nhất, với keyword chuyên biệt và trích dẫn đầy đủ, đặc biệt quan trọng trong phân tích báo cáo tài chính phức tạp, giúp giảm rủi ro hallucination và sai sót thông tin. Việc lựa chọn vectorstore làm đường đi chính cho dữ liệu tài chính cũng tối ưu hóa tốc độ truy xuất nhờ vào khả năng tìm kiếm similarity vector, đồng thời tích hợp dễ dàng với các cơ chế Self-RAG và Reflexion-RAG.

2. **Employee Database (SQL Database):** Truy vấn liên quan đến dữ liệu nội bộ của doanh nghiệp như thông tin nhân viên, bảng lương, dự án, v.v. được xử lý bởi SQL agent. Agent này trực tiếp thực hiện câu lệnh SQL, trích xuất dữ liệu chính xác từ cơ sở dữ liệu, sau đó cung cấp kết quả cho LLM để sinh câu trả lời logic. Phương pháp này đảm bảo dữ liệu nhạy cảm hoặc có cấu trúc quan hệ được truy xuất một cách chính xác, tránh việc LLM dựa trên vector similarity mà có thể dẫn tới thông tin sai lệch. SQL agent cũng hỗ trợ các tính năng như kiểm tra quyền truy cập, đảm bảo compliance, và kết hợp với metadata của từng bảng để lọc dữ liệu trước khi sinh câu trả lời.

3. **General Knowledge (Web Search):** Nếu truy vấn yêu cầu kiến thức chung hoặc thông tin cập nhật từ internet, Adaptive RAG sẽ kích hoạt web search agent. Agent này thực hiện tìm kiếm trên web hoặc cơ sở dữ liệu kiến thức công khai, tập hợp các tài liệu có liên quan, và cung cấp dữ liệu này cho LLM để sinh câu trả lời. Cơ chế này đảm bảo hệ thống

không giới hạn ở dữ liệu nội bộ hoặc vectorstore, mà vẫn có khả năng cung cấp thông tin cập nhật, tổng hợp từ nhiều nguồn công khai, và giảm rủi ro thiếu thông tin khi giải quyết các query mở. Web search agent cũng có khả năng lọc dữ liệu theo domain, kiểm tra độ tin cậy của nguồn và ưu tiên trích dẫn từ các nguồn đáng tin cậy.

### 8.3.3 Intelligent Query Routing

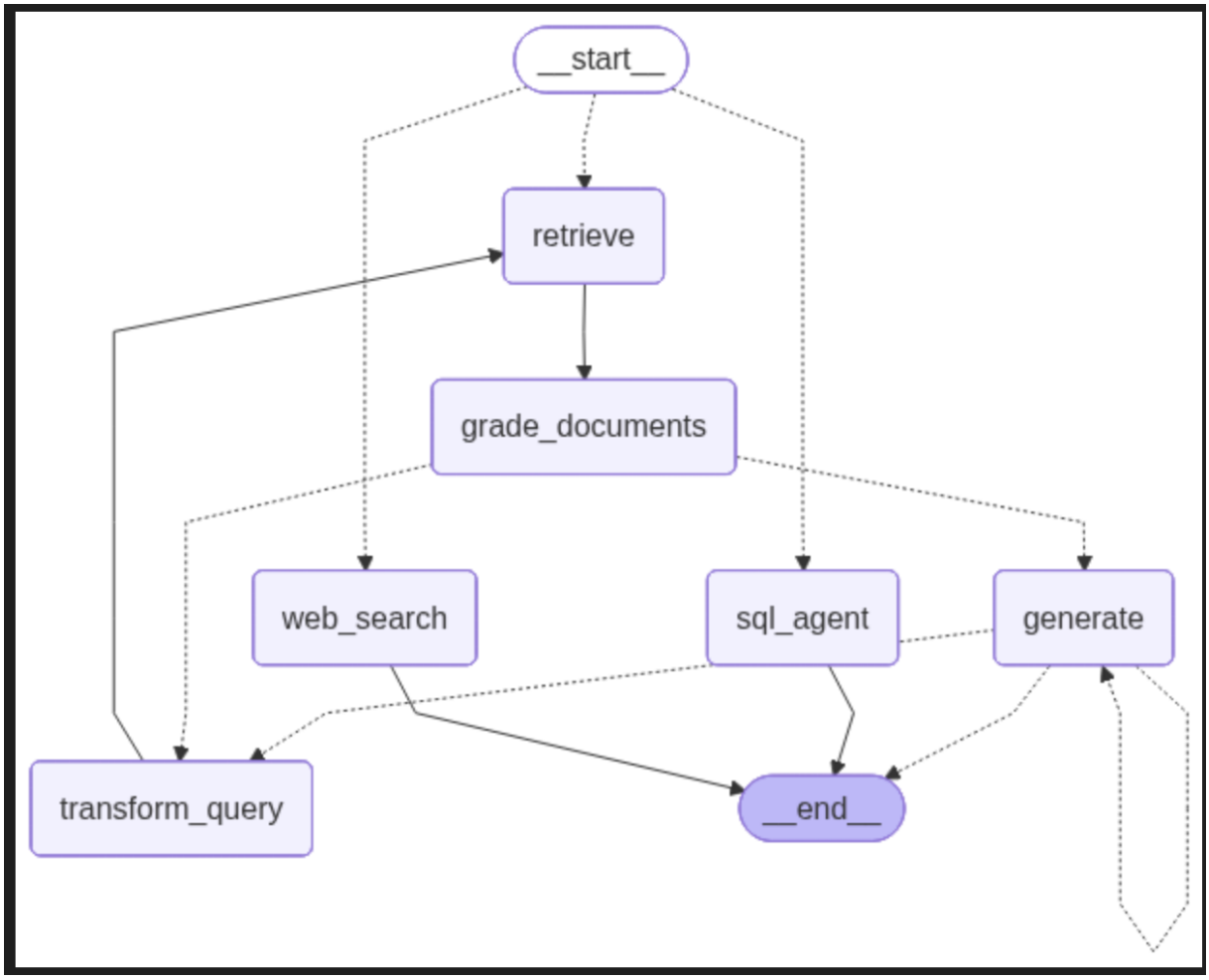
**Khái niệm:** Intelligent query routing là cơ chế phân tích query và quyết định đường đi tối ưu dựa trên nội dung, intent và ngữ cảnh. Không giống các hệ thống gửi query đồng thời tới tất cả node, Adaptive RAG chọn một hoặc nhiều đường đi phù hợp, tối ưu hóa tốc độ và độ chính xác retrieval.

**Cách hoạt động:**

- Bước 1: Query được phân tích để xác định intent, loại dữ liệu cần truy xuất, và mức độ ưu tiên cho từng loại thông tin.
- Bước 2: Hệ thống xác định routing path phù hợp (vectorstore, SQL database hoặc web search) dựa trên bộ quy tắc hoặc mô hình phân loại.
- Bước 3: Query được gửi tới agent tương ứng, có thể thực hiện tuần tự hoặc song song nếu nhiều đường đi khả thi, với trọng số ưu tiên khác nhau để tối ưu tài nguyên tính toán.
- Bước 4: Kết quả retrieval từ các đường đi được tổng hợp và chuẩn hóa, cung cấp cho LLM để sinh câu trả lời cuối cùng.
- Bước 5: Nếu câu trả lời chưa đầy đủ, hệ thống kích hoạt cơ chế query rewriting và iterative refinement cho các path hỗ trợ self-improvement, đảm bảo câu trả lời logic, toàn diện và có trích dẫn.

**Ý nghĩa học thuật:** Intelligent query routing giúp Adaptive RAG trở nên chủ động, giảm số lượng truy vấn không cần thiết, tăng tốc độ và độ chính xác retrieval, đồng thời tối ưu hóa chi phí tính toán khi xử lý nhiều nguồn dữ liệu lớn. Đây là cơ chế nền tảng để kết hợp nhiều agent AI và workflow retrieval trong cùng một hệ thống.

### 8.3.4 Workflow tổng thể của Adaptive RAG




Adaptive RAG kết hợp các cơ chế retrieval khác nhau thành một workflow liên kết chặt chẽ:

1. Phân tích query người dùng để xác định intent và loại dữ liệu cần truy xuất.
2. Lựa chọn routing path phù hợp: vectorstore, SQL database hoặc web search.
3. Thực hiện retrieval theo workflow tương ứng (Self-RAG, SQL agent, web search agent).
4. Áp dụng document relevance grading, hallucination detection, completeness checking nếu path là vectorstore.
5. Tổng hợp dữ liệu retrieved từ tất cả các nguồn cần thiết, chuẩn hóa metadata và trích dẫn.
6. Sinh câu trả lời cuối cùng, kèm cited answers và metadata đầy đủ.
7. Nếu câu trả lời chưa đầy đủ, kích hoạt query rewriting và iterative refinement cho các path hỗ trợ self-improvement, đảm bảo chất lượng câu trả lời cuối cùng.

### 8.3.5 Ứng dụng thực tiễn

Adaptive RAG đặc biệt phù hợp cho các hệ thống đa nguồn dữ liệu hoặc domain phức tạp:

- **Doanh nghiệp đa quốc gia:** Kết hợp dữ liệu nội bộ, báo cáo tài chính, và kiến thức công khai để trả lời câu hỏi về nhân sự, dự án, hoặc phân tích công ty.
- **Tài chính:** Tích hợp SEC filings, dữ liệu thị trường và báo cáo nội bộ để phân tích công ty, dự đoán doanh thu, hoặc đánh giá rủi ro tài chính.
- **Pháp lý và compliance:** Truy xuất hợp đồng, case law, quy định pháp luật kết hợp dữ liệu nội bộ để hỗ trợ luật sư hoặc bộ phận compliance.

- **Y tế và nghiên cứu khoa học:** Kết hợp patient records, tài liệu nghiên cứu và kiến thức y học cập nhật để hỗ trợ chẩn đoán, phân tích dữ liệu hoặc viết báo cáo nghiên cứu.

### 8.3.6 Ý nghĩa học thuật

Adaptive RAG đại diện cho bước tiến vượt bậc trong phát triển các hệ thống Retrieval-Augmented Generation. Hệ thống không chỉ retrieve dữ liệu dựa trên vector similarity mà còn phân tích thông minh, lựa chọn agent và path phù hợp với loại truy vấn.

Điều này giúp tăng độ chính xác, giảm rủi ro hallucination, đảm bảo tính đầy đủ và logic của câu trả lời. Đồng thời, cơ chế này mở ra khả năng xây dựng các hệ thống AI thích nghi theo thời gian, tự học hỏi và tối ưu workflow retrieval dựa trên đặc thù từng domain hoặc từng loại query.

Adaptive RAG đặt nền tảng cho các hệ thống đa-agent tự phản hồi, thích nghi với dữ liệu lớn và phức tạp, đồng thời kết hợp các cơ chế self-improvement, query rewriting, và iterative refinement, giúp AI ngày càng thông minh và đáng tin cậy hơn trong các ứng dụng thực tiễn.

## 8.4 Ứng dụng thực tiễn

Adaptive RAG được thiết kế đặc biệt cho các hệ thống cần quản lý và truy xuất dữ liệu từ nhiều nguồn khác nhau, hoặc trong các domain có độ phức tạp cao, nơi việc lựa chọn thông tin phù hợp và đáng tin cậy là tối quan trọng. Khả năng phân tích query thông minh và routing đến agent phù hợp giúp hệ thống này vượt trội so với các giải pháp RAG truyền thống, đặc biệt trong các môi trường dữ liệu đa dạng và liên tục thay đổi.

- **Doanh nghiệp đa quốc gia:** Trong các tập đoàn lớn, thông tin được lưu trữ rải rác ở nhiều hệ thống khác nhau: tài liệu nội bộ, báo cáo dự án, email, và các cơ sở dữ liệu chuyên biệt. Adaptive RAG cho phép truy xuất và kết hợp dữ liệu từ nhiều nguồn này để trả lời các câu hỏi phức tạp về nhân sự, tiến độ dự án, phân tích hiệu quả hoạt động hoặc báo cáo tổng hợp cho ban lãnh đạo. Hệ thống có thể tự động phân loại query và routing đến agent phù hợp, giúp giảm thiểu rủi ro lỗi do truy vấn nhầm nguồn hoặc bỏ sót dữ liệu quan trọng.

- **Tài chính và phân tích công ty:** Adaptive RAG hỗ trợ tích hợp dữ liệu SEC filings, báo cáo nội bộ, dữ liệu thị trường và phân tích thống kê để cung cấp các câu trả lời chính xác về hiệu quả tài chính, dự đoán doanh thu hoặc đánh giá rủi ro. Hệ thống có khả năng đánh giá tính đầy đủ và chính xác của câu trả lời, phát hiện các thông tin thiếu sót hoặc không liên quan, và tự động rewrite query nếu cần. Điều này đặc biệt quan trọng trong các phân tích phức tạp, nơi mọi quyết định tài chính đều dựa trên dữ liệu thực tế và đáng tin cậy.

- **Pháp lý và compliance:** Trong các ứng dụng pháp lý, Adaptive RAG có thể kết hợp truy xuất hợp đồng, case law, quy định pháp luật với dữ liệu nội bộ để hỗ trợ luật sư hoặc bộ phận compliance trong việc so sánh điều khoản, kiểm tra tính hợp lệ và đánh giá rủi ro pháp lý. Nhờ khả năng query routing và self-reflection, hệ thống tự động kiểm tra tính đầy đủ, phát hiện các trường hợp thiếu thông tin hoặc mâu thuẫn, đồng thời trích dẫn chính xác nguồn dữ liệu, giúp tăng độ tin cậy và minh bạch.

- **Y tế và nghiên cứu khoa học:** Adaptive RAG có thể kết hợp dữ liệu patient records, kết quả xét nghiệm, tài liệu nghiên cứu, và kiến thức y học cập nhật để hỗ trợ chẩn đoán, phân tích dữ liệu và viết báo cáo nghiên cứu. Hệ thống có khả năng đánh giá độ tin cậy của nguồn thông tin, kiểm tra hallucination, và rewrite query khi phát hiện khoảng trống dữ liệu, giúp các nhà nghiên cứu và bác sĩ nhận được thông tin toàn diện và chính xác, đặc biệt quan trọng trong các tình huống nhạy cảm liên quan đến sức khỏe.

## 8.5 Ý nghĩa học thuật

Adaptive RAG đại diện cho một bước tiến vượt bậc trong lĩnh vực Retrieval-Augmented Generation nhờ khả năng kết hợp thông minh nhiều nguồn dữ liệu, self-reflection và query

rewriting. Hệ thống không chỉ dựa vào vector similarity để retrieve dữ liệu mà còn có khả năng phân tích ngữ nghĩa của truy vấn, xác định intent, và lựa chọn routing path phù hợp để gửi query đến agent tối ưu, từ đó tăng độ chính xác và hiệu quả tổng thể.

Điều này mở ra nhiều ý nghĩa học thuật quan trọng:

- **Tăng độ chính xác và giảm rủi ro hallucination:** Nhờ self-reflection, hallucination detection và iterative refinement, Adaptive RAG có thể nhận diện các thông tin không liên quan hoặc thiếu cơ sở dữ liệu, tự động rewrite query để retrieve bổ sung dữ liệu đáng tin cậy. Điều này làm giảm thiểu khả năng LLM sinh câu trả lời sai hoặc thiếu thông tin, một vấn đề phổ biến trong các hệ thống RAG truyền thống.

- **Đảm bảo tính đầy đủ và logic của câu trả lời:** Bằng cách kết hợp metadata filtering, keyword-based re-ranking, và cited answers, hệ thống đảm bảo rằng mọi câu trả lời đều toàn diện, logic, và có nguồn dẫn rõ ràng. Các câu trả lời không chỉ chính xác mà còn có thể kiểm chứng, phù hợp cho các domain nhạy cảm như tài chính, pháp lý, y tế và nghiên cứu học thuật.

- **Khả năng thích nghi và self-improvement:** Adaptive RAG mở ra khả năng xây dựng hệ thống AI tự học hỏi và cải thiện theo thời gian thực. Khi query mới xuất hiện hoặc nguồn dữ liệu thay đổi, cơ chế routing thông minh kết hợp với iterative refinement và query rewriting giúp hệ thống thích nghi với dữ liệu lớn, phức tạp, và liên tục cập nhật, tạo ra một AI linh hoạt và tự phản hồi.

- **Mở rộng ứng dụng đa domain:** Nhờ khả năng routing và tích hợp nhiều agent (Self-RAG, Reflexion-RAG, SQL agent, Web search), Adaptive RAG cho phép triển khai trên nhiều domain khác nhau mà vẫn giữ được hiệu quả retrieval cao. Đây là nền tảng cho các hệ thống agent AI phức tạp, nơi thông tin từ nhiều nguồn cần được kết hợp một cách logic, toàn diện, và có thể giải thích được.

Tóm lại, Adaptive RAG không chỉ cải thiện hiệu quả retrieval mà còn nâng cao chất lượng thông tin, tính minh bạch và khả năng tự học của hệ thống AI. Nó đặt nền tảng cho các hệ thống đa-agent tự phản hồi, thích nghi với dữ liệu lớn, phức tạp, và là bước tiến quan trọng trong nghiên cứu và ứng dụng RAG hiện đại.

## 9: Kết luận và Tài liệu tham khảo

Trong bối cảnh ngày càng nhiều ứng dụng yêu cầu độ chính xác cao, tính cập nhật, và khả năng truy xuất từ nhiều nguồn dữ liệu — từ tài chính, pháp lý, y tế tới kiến thức học thuật — các kỹ thuật truyền thống dựa hoàn toàn vào mô hình ngôn ngữ lớn (LLMs) dần bộc lộ rõ rệt các hạn chế: thông tin lỗi thời, hallucination, thiếu khả năng truy xuất dữ liệu mới, và khó đảm bảo tính chính xác khi yêu cầu thông tin chuyên sâu.

Phương pháp Retrieval-Augmented Generation (RAG) giúp giảm bớt nhiều hạn chế đó bằng cách kết hợp truy xuất thông tin bên ngoài với khả năng sinh ngôn ngữ của LLM. Tuy nhiên, việc áp dụng RAG đơn giản, với retrieval cố định và không kiểm tra chặt chẽ chất lượng thông tin, dễ dẫn đến việc LLM sử dụng dữ liệu không phù hợp, dẫn đến output không chính xác, thiếu logic, hoặc thiếu trích dẫn rõ ràng.

Những nghiên cứu gần đây — đặc biệt Self-Reflective Retrieval-Augmented Generation (Self-RAG) — đã chỉ ra hướng đi mạnh mẽ để khắc phục các hạn chế đó. Self-RAG cho phép LLM *tự nhiên truy xuất theo nhu cầu*, tự đánh giá output, và quyết định xem có cần bổ sung thông tin hay không — thông qua các "reflection tokens". Kết quả thực nghiệm từ Self-RAG cho thấy cải thiện rõ rệt về độ chính xác, factuality và độ tin cậy trên nhiều nhiệm vụ: từ câu hỏi mở, xác minh tri thức, đến sinh văn bản dài có trích dẫn. :contentReference[oaicite:3]index=3

Bên cạnh đó, nghiên cứu Corrective Retrieval Augmented Generation (CRAG) bổ sung một lớp kiểm tra và sửa lỗi retrieval — tức là đánh giá độ "tốt / đủ" của tài liệu được retrieve trước khi LLM sử dụng chúng để sinh nội dung. Khi retrieval trả về tài liệu không phù hợp, CRAG có thể kích hoạt fallback — ví dụ mở rộng tìm kiếm trên web hoặc sử dụng chiến lược lọc/ghép

lại nội dung để tối ưu thông tin. Điều này làm tăng tính robust của pipeline RAG trước những truy vấn khó hoặc dữ liệu phức tạp. :contentReference[oaicite:5]index=5

Kết hợp các cơ chế này — retrieval theo nhu cầu, self-reflection, kiểm tra/hệ thống sửa lỗi retrieval, re-ranking, metadata filtering — tạo nên framework **Adaptive RAG**. Đây là hệ thống mạnh mẽ, linh hoạt, có khả năng:

- phục vụ nhiều domain khác nhau (tài chính, pháp lý, y tế, học thuật) với yêu cầu cao về độ chính xác và trích dẫn; - tự động điều chỉnh truy xuất và sinh nội dung phù hợp với từng truy vấn cụ thể; - giảm hallucination, tăng factuality và khả năng kiểm chứng; - mở rộng dễ dàng khi nguồn dữ liệu tăng — từ vector store nội bộ, database quan hệ, đến web, knowledge-base tổng quát.

Vì vậy, Adaptive RAG — với Self-RAG và các kỹ thuật hỗ trợ như CRAG — là hướng nghiên cứu và ứng dụng rất hứa hẹn cho các hệ thống AI thế hệ mới: thông minh, đáng tin cậy, minh bạch và sẵn sàng tích hợp trong công nghiệp hoặc nghiên cứu chuyên sâu.

Tuy nhiên, không nên bỏ qua các hạn chế và thách thức còn tồn tại, ví dụ:

- Việc thiết lập và duy trì vector store, metadata, indexing, re-ranking, fallback mechanisms đòi hỏi công sức và hạ tầng phù hợp; - Trong một số domain nhạy cảm, việc đảm bảo dữ liệu nguồn đáng tin cậy và kiểm chứng nhiều tầng là rất quan trọng; - Hiệu suất (tốc độ, chi phí tính toán) khi thực hiện retrieval + generation + self-reflection + re-retrieval đôi khi cao hơn RAG truyền thống; - Đòi hỏi thiết kế và tùy chỉnh phù hợp: embedding, re-ranking, retrieval evaluator, logic routing...

Nhìn chung, với sự phát triển của các thư viện vector store, embedding models, cơ chế re-ranking và các nghiên cứu như Self-RAG, CRAG, Adaptive RAG đang đặt nền móng cho các hệ thống AI thế hệ mới — đáng tin, minh bạch và mạnh mẽ — phù hợp với nhiều bài toán thực tiễn và nghiên cứu. Việc tiếp tục phát triển, tinh chỉnh và đánh giá các hệ thống này sẽ là một hướng rất giá trị trong tương lai.

**Tóm lại:** Adaptive RAG không chỉ là một kỹ thuật, mà là một *framework tổng thể* — kết hợp retrieval, generation, self-reflection, correction và routing thông minh — mở ra khả năng thiết kế các AI Agent thực sự thông minh, đáng tin cậy, và có thể áp dụng trong môi trường dữ liệu đa dạng, phức tạp và thay đổi liên tục.

## 9.1 Lời cảm ơn

Trước hết, tôi xin gửi lời biết ơn sâu sắc tới Thầy/Cô Lê Hoàng Sơn người hướng dẫn khoa học trực tiếp của tôi trong suốt quá trình nghiên cứu và thực hiện dự án này. Với kiến thức chuyên môn uyên thâm, sự kiên nhẫn và tinh thần tận tâm, Thầy/Cô đã luôn chỉ dẫn, giải đáp các thắc mắc và góp ý từng chi tiết trong việc thiết kế phương pháp, triển khai mô hình, cũng như phân tích kết quả. Sự định hướng và động viên của Thầy/Cô là nguồn cảm hứng lớn giúp tôi vượt qua những thử thách kỹ thuật và nghiên cứu trong lĩnh vực AI nâng cao.

## 9.2 Tài liệu tham khảo

- Asai, Akari; Wu, Zeqiu; Wang, Yizhong; Sil, Avirup; Hajishirzi, Hannaneh (2023). *Self-Reflective Retrieval-Augmented Generation (Self-RAG).* arXiv preprint arXiv:2310.11511. :contentReference[oaicite:6]index=6
- Yan, Shi-Qi; Gu, Jia-Chen; Zhu, Yun; Ling, Zhen-Hua (2024). *Corrective Retrieval Augmented Generation (CRAG).* arXiv preprint arXiv:2401.15884. :contentReference[oaicite:7]index=7
- Huang, Yizheng; Huang, Jimmy (2024). *A Survey on Retrieval-Augmented Text Generation for Large Language Models.* arXiv preprint arXiv:2404.10981. :contentReference[oaicite:8]index=8
- Shao, Zhihong; Gong, Yeyun; Shen, Yelong; Huang, Minlie; Duan, Nan; Chen, Weizhu (2023). *Enhancing Retrieval-Augmented Large Language Models with Iterative Retrieval-Generation Synergy.* arXiv preprint arXiv:2305.15294. :contentReference[oaicite:9]index=9
- Wikipedia contributors (2025). *Retrieval-augmented generation.* Wikipedia. :contentReference[oaicite:10]index=10