ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO ĐỒ ÁN CUỐI KỲ

Vietnamese RAG

Môn: **Natural Language Processing (NLP)**

Giảng viên hướng dẫn:	PGS.TS. ĐINH ĐIỀN	
	TS. NGUYỄN HỒNG BỬU LONG	
Nhóm sinh viên thực hiện:	NGUYỄN HỮU KHÁNH HƯNG	23120271
	PHẠM QUỐC KHÁNH	23120283
	PHẠM THÀNH NAM	23120301
	VŨ TRẦN PHÚC	23120333
Lớp :	CQ2023/22	

TP HCM, 01/2026

Mục lục

Lời cảm ơn

Nhóm xin bày tỏ lòng biết ơn sâu sắc đến **PGS.TS. Đinh Điền**, **TS. Nguyễn Hồng Bửu Long**, **TS. Lương An Vinh** và **ThS. Lê Đức Khoan** đã tận tình hướng dẫn, giảng dạy và đóng góp nhiều ý kiến quý báu trong suốt quá trình học tập và thực hiện đồ án.

Sự chỉ dẫn và hỗ trợ của các Thầy không chỉ giúp nhóm hoàn thành đề tài đúng hướng mà còn giúp nhóm nâng cao kiến thức chuyên môn, tư duy nghiên cứu và kỹ năng thực hành. Nhóm xin chân thành cảm ơn và kính chúc các Thầy thật nhiều sức khỏe, thành công trong công tác giảng dạy và nghiên cứu.

Tóm tắt (Abstract)

Các mô hình ngôn ngữ lớn (LLM) thường gặp vấn đề "ảo giác" (trả lời sai sự thật) khi thiếu kiến thức chuyên ngành. Để giải quyết vấn đề này trong lĩnh vực Y học cổ truyền, nhóm nghiên cứu đề xuất giải pháp **Agentic Corrective RAG (CRAG)**.

Điểm đột phá của hệ thống là khả năng **tự đánh giá và sửa lỗi** thay vì tin tưởng mù quáng vào dữ liệu tìm thấy. Hệ thống sử dụng cơ chế **Lọc lai (Hybrid Filter)** kết hợp giữa đánh giá ngữ nghĩa và suy luận logic để loại bỏ tài liệu nhiễu. Đặc biệt, khi tri thức nội bộ không đủ, hệ thống tự động kích hoạt **Tìm kiếm Web** để bổ sung thông tin thời gian thực.

Kết quả thực nghiệm cho thấy phương pháp này cải thiện đáng kể độ chính xác và trung thực so với các hệ thống RAG truyền thống, được kiểm chứng qua các bộ thang đo khắt khe (LLM-as-a-judge).

1. Giới thiệu

1.1. Bài toán

Sự phát triển của các Mô hình Ngôn ngữ Lớn (LLM) đã mở ra tiềm năng to lớn trong việc tự động hóa trả lời câu hỏi. Tuy nhiên, trong các lĩnh vực chuyên sâu như **Y học Cổ truyền (YHCT)**, việc áp dụng LLM gặp nhiều thách thức do tính chất đặc thù của thuật ngữ và sự khan hiếm dữ liệu số hóa chất lượng cao.

Bài toán đặt ra là xây dựng một hệ thống Hỏi-Đáp (QA) chuyên biệt cho miền dữ liệu YHCT tiếng Việt. Hệ thống yêu cầu khả năng:

1. Truy xuất chính xác thông tin từ kho tri thức nội bộ (corpus).
2. Đảm bảo câu trả lời có căn cứ (grounded), trích dẫn nguồn rõ ràng.
3. Giảm thiểu tối đa hiện tượng "ảo giác" (hallucination) khi hệ thống không tìm thấy tài liệu phù hợp.

1.2. Động lực

Phương pháp *Retrieval-Augmented Generation (RAG)* truyền thống (Naive RAG) giải quyết tốt vấn đề cập nhật kiến thức cho LLM. Tuy nhiên, nó tồn tại điểm yếu chí tử: **"Rác vào, rác ra" (Garbage In, Garbage Out)**. Khi bộ truy hồi (Retriever) trả về tài liệu sai hoặc không liên quan, LLM thường có xu hướng bịa đặt câu trả lời để làm hài lòng người dùng thay vì thừa nhận sự thiếu hụt thông tin.

Để khắc phục, kiến trúc **Corrective RAG (CRAG)** được đề xuất nhằm bổ sung cơ chế "tự đánh giá và sửa lỗi". Bên cạnh đó, việc quản lý luồng xử lý phức tạp (có rẽ nhánh, vòng lặp) đòi hỏi một khung làm việc linh hoạt như **LangGraph**, cho phép mô hình hóa hệ thống dưới dạng đồ thị trạng thái (StateGraph) thay vì các chuỗi xử lý tuyến tính cứng nhắc.

1.3. Đóng góp

1. Tái hiện ý tưởng CRAG: thêm bước chấm tài liệu truy hồi và kích hoạt web search khi cần.

2. Tích hợp agentic RAG bằng LangGraph: triển khai pipeline retrieve → grade → (rewrite+web) → generate.

3. Áp dụng cho miền tiếng Việt chuyên ngành (YHCT) với embedding đa ngôn ngữ bge-m3.

4. Thiết kế đánh giá Faithfulness/Relevance/Correctness và xuất báo cáo tự động.

2. Nghiên cứu liên quan

2.1. Retrieval-Augmented Generation (RAG)

RAG là kỹ thuật kết hợp giữa mô hình truy hồi thông tin (Retriever) và mô hình sinh văn bản (Generator). Retriever tìm kiếm các đoạn văn bản (chunks) liên quan từ cơ sở dữ liệu vector dựa trên độ tương đồng ngữ nghĩa, sau đó Generator sử dụng các đoạn văn này làm ngữ cảnh (context) để sinh câu trả lời. Mặc dù RAG giúp giảm ảo giác so với việc chỉ dùng LLM thuần túy, hiệu quả của nó phụ thuộc hoàn toàn vào chất lượng của tài liệu truy hồi được.

2.2. Corrective Retrieval-Augmented Generation (CRAG)

Được giới thiệu bởi Yan và cộng sự (2024), CRAG nâng cấp RAG bằng cách thêm một bộ đánh giá (Retrieval Evaluator) để kiểm định chất lượng tài liệu trước khi đưa vào sinh câu trả lời. Dựa trên kết quả đánh giá (Chính xác/Mơ hồ/Sai), hệ thống sẽ kích hoạt các hành động sửa lỗi tương ứng như cắt tỉa kiến thức hoặc tìm kiếm thông tin bổ sung từ Web, giúp hệ thống bền vững hơn trước nhiễu.

2.3. Agentic RAG và LangGraph

Khác với các chuỗi RAG tuyến tính (Linear Chains), **Agentic RAG** trao quyền cho hệ thống tự quyết định bước tiếp theo dựa trên trạng thái hiện tại. **LangGraph** là thư viện hỗ trợ xây dựng các hệ thống này dưới dạng đồ thị có hướng, cho phép thiết lập các vòng lặp

(loops) và rẽ nhánh điều kiện (conditional edges). Điều này đặc biệt hữu ích cho CRAG, nơi hệ thống có thể cần quay lại bước tìm kiếm nhiều lần nếu chưa đủ thông tin.

2.4. Embedding và mô hình sinh

Để triển khai hệ thống cho tiếng Việt chuyên ngành, việc lựa chọn mô hình đóng vai trò then chốt:

Embedding Model: Sử dụng **BAAI/bge-m3**, một mô hình SOTA (State-of-the-art) về nhúng đa ngôn ngữ, hỗ trợ độ dài ngữ cảnh lớn và khả năng nắm bắt tốt các thuật ngữ Hán-Việt trong Y học cổ truyền.

Generative Model: Sử dụng **Qwen/Qwen2.5-7B-Instruct**, dòng mô hình mã nguồn mở có khả năng suy luận và tuân theo chỉ dẫn (instruction following) tốt nhất hiện nay trong phân khúc 7B. Mô hình này được tối ưu hóa để chạy cục bộ (Local Deployment) thông qua thư viện *Transformers* với cấu hình lượng tử hóa (`bfloat16`/4-bit), đóng vai trò là "bộ não" đa nhiệm: vừa là giám khảo (grader), vừa là biên tập viên (rewriter), vừa là tác giả (generator).

Reranker: Tích hợp **BAAI/bge-reranker-base** hoạt động theo cơ chế Cross-Encoder, giúp tính toán điểm tương đồng chi tiết giữa câu hỏi và tài liệu, bổ trợ cho việc đánh giá của LLM.

3. Dữ liệu

3.1. Corpus tri thức nội bộ

Nhóm xây dựng kho tri thức từ các tài liệu Y học cổ truyền tiếng Việt (định dạng .docx). Trong notebook, dữ liệu được nạp và chia thành nhiều tập chunk (từ chunks1 đến chunks4) tương ứng với các tài liệu nguồn khác nhau (ví dụ: bệnh ngũ quan, nội khoa y học cổ truyền, …). Mỗi chunk được lưu dưới dạng Document kèm metadata (tiêu đề/mục/bệnh) để thuận tiện truy vết khi đánh giá.

3.2. Tiền xử lý & chia đoạn (chunking)

Nhóm triển khai tiền xử lý và chunking theo cấu trúc tài liệu: tách theo chương/mục bằng regex marker, sau đó tách đệ quy theo heading để giữ ngữ cảnh theo cây mục lục. Với các đoạn gạch đầu dòng, pipeline ưu tiên “gom” các bullet liên quan vào cùng mục cha nhằm hạn chế mất ngữ cảnh khi truy hồi. Kết quả chunking được dùng để tạo vector index trên Chroma.

3.3. Tập đánh giá (Evaluation set)

Notebook sử dụng một tập đánh giá (eval_data) dạng JSON gồm các trường: question và ground_truth_answer (được nạp từ file *.json trong Google Drive). Trong log chạy mẫu, tập đánh giá có 53 câu hỏi. Tập này được dùng để chạy hệ thống, chấm điểm tự động và xuất báo cáo ra Excel.

4. Phương pháp

4.1. Tổng quan kiến trúc

Hệ thống gồm 5 khối chính:

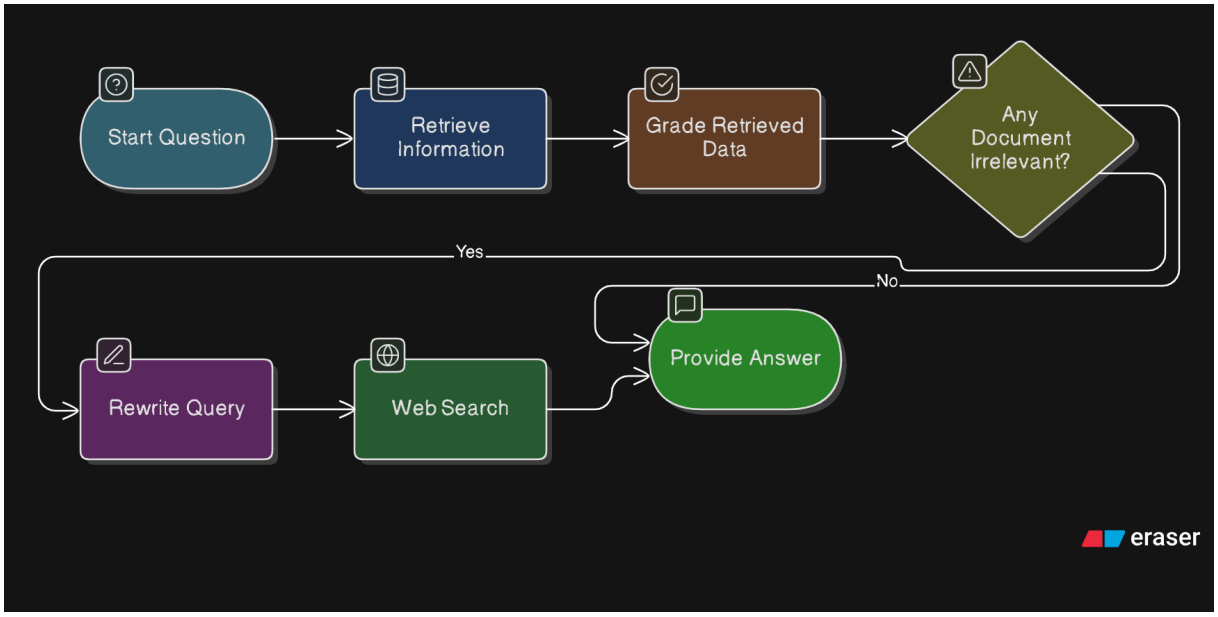
1. Vector Indexing: embedding BAAI/bge-m3 + Chroma (cosine, collection_name='y_hoc_co_truyen', persist_directory='./YHCT').

2. Retriever: similarity_score_threshold với k=3, score_threshold=0.3.

3. Hybrid Relevance Filter: LLM grader (yes/no) + reranker cross-encoder BAAI/bge-reranker-base. Điểm tổng hợp = $0.6 \cdot \text{sigmoid}(\text{rerank\_score}) + 0.4 \cdot \text{llm\_norm}$; giữ doc nếu score ≥ 0.5.

4. Query Rewrite + Web Search: nếu cần, rewrite câu hỏi và tìm kiếm web bằng DuckDuckGo (thư viện ddgs), gộp nội dung web thành một Document bổ sung ngữ cảnh.

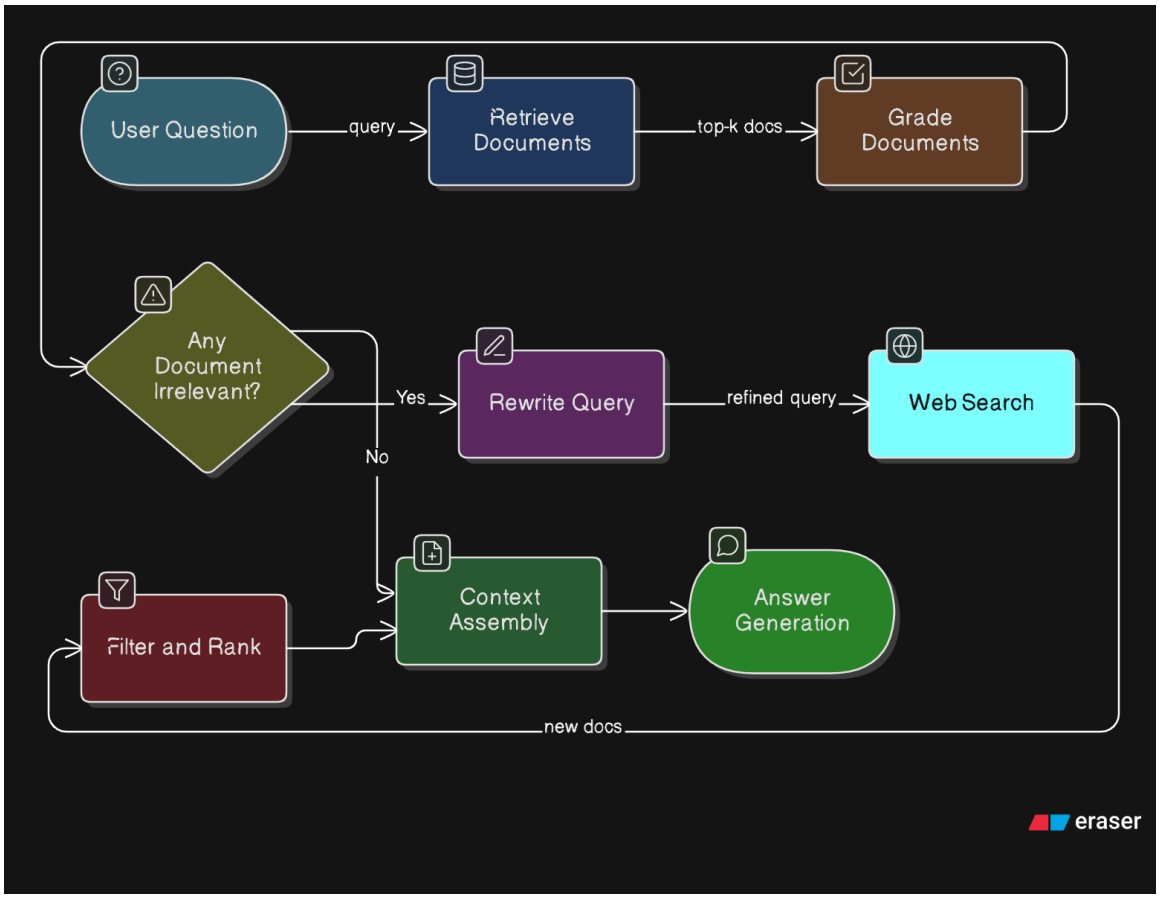
5. Answer Generation: qa_rag_chain sinh câu trả lời dựa trên context (docs đã lọc + web doc nếu có), dùng cùng Qwen2.5-7B-Instruct.



Hình 1. Minh hoạ workflow/pipeline trong notebook.

4.2. Mô hình hoá bằng LangGraph

Nhóm định nghĩa GraphState gồm: question, documents, web_search_needed và generation. Đồ thị LangGraph (StateGraph) có các node chính: retrieve, grade_documents_hybrid, rewrite_query, web_search, generate_answer. Luồng xử lý: retrieve → grade_documents_hybrid → (nếu web_search_needed='Yes' thì rewrite_query → web_search → generate_answer; ngược lại đi thẳng generate_answer) → END. Thiết kế dạng đồ thị giúp pipeline rẽ nhánh rõ ràng, dễ debug và mở rộng.



Hình 2. Minh hoạ đồ thị LangGraph (StateGraph).

4.3. Logic chấm và sửa (Corrective loop)

Trong node grade_documents_hybrid, hệ thống chấm từng tài liệu truy hồi bằng LLM grader (yes/no), đồng thời tính điểm reranker (CrossEncoder) cho cặp (question, document). Điểm tổng hợp được tính theo trọng số 0.6 (reranker) và 0.4 (LLM). Nếu sau khi lọc không còn tài liệu nào (hoặc một phần lớn bị đánh giá kém liên quan), biến web_search_needed được đặt 'Yes' để kích hoạt nhánh rewrite_query → web_search; ngược lại hệ thống chuyển thẳng đến generate_answer.

4.4. Hybrid scoring (LLM Grader + Reranker)

Để giảm trường hợp retriever trả về chunk nhiễu, nhóm kết hợp hai nguồn tín hiệu: (i) LLM grader trả về nhãn yes/no và (ii) reranker CrossEncoder (BAAI/bge-reranker-base) cho điểm liên quan theo cặp (question, doc). Trong code, rerank_score được chuẩn hoá về [0,1] bằng sigmoid; llm_norm = 1 nếu 'yes' và 0 nếu 'no'. Điểm cuối cùng: score = 0.6·rerank_norm + 0.4·llm_norm. Tài liệu được giữ lại nếu score ≥ 0.5. Cách làm này

vẫn giữ được ưu điểm "giải thích được" của grader (lọc thô) và thêm độ chính xác của reranker (lọc tinh).

5. Thiết lập thí nghiệm

5.1. Môi trường và thư viện

• LangChain & LangGraph để xây dựng chuỗi và đồ thị tác tử.

• ChromaDB làm vector database.

• HuggingFace Transformers để chạy Qwen/Qwen2.5-7B-Instruct local (dùng chung cho grader/rewrite/generate).

• sentence-transformers CrossEncoder: BAAI/bge-reranker-base dùng cho reranking và hybrid grading.

• ddgs (DuckDuckGo) làm công cụ tìm kiếm web.

5.2. Baseline và mô hình đề xuất

• Baseline (Naive RAG): similarity retrieval (k=3) → generate (không có grader/rewrite/web).

• Proposed (Agentic CRAG): retrieve → hybrid grade (LLM + reranker) → nếu cần rewrite + web search → generate.

5.3. Thước đo đánh giá

Sử dụng một mô hình LLM khác nhằm đánh giá chính xác theo 3 metrics gồm: Faithfulness, Relevance và Correctness theo thang điểm từ 1 đến 5 tăng dần tương ứng mức độ hoàn thiện của câu trả lời.

6. Kết quả và đánh giá

6.1. Kết quả định lượng

Chạy vòng lặp đánh giá trên tập eval_data (53 câu hỏi trong log chạy mẫu), notebook tự động:

• Sinh câu trả lời từ Agentic CRAG và chấm Faithfulness/Relevance/Correctness.

• Lưu bảng chi tiết ra results.xlsx (các cột chính: question, generated_answer, ground_truth, context_retrieved, score_*).

• (Tuỳ chọn) chạy Baseline Naive RAG và lưu kết quả ra naive_rag_results.xlsx

Trong phạm vi báo cáo này, nhóm trình bày quy trình và cách tái lập kết quả; số liệu cụ thể có thể đọc trực tiếp từ các file Excel đầu ra sau khi chạy notebook.

Bảng 1. Khung tổng hợp điểm (điền số liệu từ file Excel):

	Naive RAG	Corrective RAG	Agentic Corrective RAG
Faithfullness	3.98	_	4.21
Relevance	3.68	1. 48	3.96
Correctness	4.02	_	3.96

6.2. Phân tích định tính

• Trường hợp retriever trả tài liệu lệch: grader phát hiện kém liên quan → kích hoạt rewrite + web search → bổ sung ngữ cảnh → giảm trả lời sai.

• Trường hợp câu hỏi ngoài miền: web search giúp có ngữ cảnh nhưng có thể nhiễu; cần lọc chặt hơn hoặc cơ chế từ chối trả lời có kiểm chứng.

7. Thảo luận và hạn chế

1. Hybrid grading (LLM + reranker) cải thiện lọc nhiễu nhưng làm tăng chi phí suy luận (LLM) và suy luận reranker (CrossEncoder); cần tối ưu batching/caching để chạy nhanh hơn.

2. Web search không hoàn toàn tái lập (kết quả thay đổi theo thời điểm), ảnh hưởng reproducibility.

3. Phiên bản hiện tại tập trung vào lọc relevance và bổ sung web context; chưa triển khai đầy đủ các bước chọn lọc/tái cấu trúc tài liệu nâng cao.

4. Đánh giá LLM-as-a-judge có thể thiên lệch; cần thêm đánh giá thủ công hoặc metric khách quan hơn.

8. Kết luận và hướng phát triển

Nhóm đã xây dựng hệ thống Agentic Corrective RAG cho miền Y học cổ truyền tiếng Việt bằng LangGraph, với cơ chế đánh giá truy hồi và sửa sai (rewrite + web search)

trước khi sinh đáp án. Cách triển khai theo StateGraph giúp pipeline rõ ràng và dễ mở rộng.

Hướng phát triển:

- Tinh chỉnh trọng số/threshold của hybrid scoring, hoặc thay CrossEncoder bằng reranker nhẹ hơn để giảm độ trễ.
- Khai thác hybrid retrieval (dense + sparse) để tăng recall cho tiếng Việt.
- Caching kết quả web search và log version dữ liệu để tăng tính tái lập.
- Bổ sung cơ chế trích dẫn đoạn nguồn (source attribution) trong câu trả lời.

Tài liệu tham khảo

1. Yan, S.-Q., Gu, J.-C., Zhu, Y., & Ling, Z.-H. (2024). Corrective Retrieval Augmented Generation. arXiv:2401.15884.

2. LangChain-AI. LangGraph Documentation (StateGraph, conditional edges).

3. BAAI. bge-m3 multilingual embedding model (HuggingFace model card).

4. Qwen Team. Qwen2.5-7B-Instruct (HuggingFace model card).

5. BAAI. bge-reranker-base (Sentence-Transformers / HuggingFace model card).

6. ddgs. DuckDuckGo Search (Python library) documentation.

7. RAGAS. Retrieval-Augmented Generation Assessment metrics documentation.