XÂY DỰNG HỆ THỐNG CHATBOT HỎI ĐÁP TRÊN TÀI LIỆU KỸ THUẬT SỬ DỤNG RETRIEVAL-AUGMENTED GENERATION

Đặng Khánh Hòa[1*], Phạm Thị Thu Trang[1], Phạm Minh Việt[1], Nguyễn Thanh Nghị[1]
[*]Tác giả liên hệ, email: ptttrang@hou.edu.vn ORCID: 0009-0006-3599-1387

Ngày tòa soạn nhận được bài báo: 15/01/2026
Ngày phản biện đánh giá: 17/03/2026
Ngày bài báo được duyệt đăng: 14/04/2026

DOI: 10.59266/houjs.2026.1173

Tóm tắt**:** *Bài báo trình bày quá trình thiết kế và hiện thực một hệ thống chatbot hỏi đáp trên tài liệu kỹ thuật dựa trên kiến trúc Retrieval-Augmented Generation (RAG) nhằm giảm thiểu hiện tượng ảo giác (hallucination) và tăng cường khả năng truy vết nguồn thông tin. Hệ thống được xây dựng theo kiến trúc hai pha tách biệt: pha ngoại tuyến bao gồm nạp dữ liệu, làm sạch, phân đoạn văn bản, sinh vector nhúng và lập chỉ mục; pha trực tuyến thực hiện truy xuất ngữ nghĩa, xếp hạng lại, xây dựng ngữ cảnh và sinh câu trả lời thông qua mô hình ngôn ngữ lớn. Nghiên cứu triển khai và so sánh hai chiến lược phân đoạn (theo ký tự và theo cấu trúc markdown), ba chế độ truy xuất (dense, keyword, hybrid), mô hình nhúng đa ngôn ngữ BAAI/bge-m3 và mô hình xếp hạng lại BAAI/bge-reranker-v2-m3, đồng thời hỗ trợ hai backend sinh văn bản (Qwen2.5-7B-Instruct cục bộ qua Ollama và Gemini 1.5 Flash đám mây). Trên bộ kiểm thử gồm 25 câu hỏi có gán nhãn về tài liệu viễn thông 5G, cấu hình dense retrieval đạt Hit@1 = 84% và MRR = 0,913; khi bổ sung bước reranking, Hit@1 tăng lên 92% và MRR tăng lên 0,960 trong khi Hit@3 duy trì ở mức 100%. Kết quả chứng minh rằng kết hợp truy xuất ngữ nghĩa dày đặc với xếp hạng lại Cross-Encoder là hướng tiếp cận hiệu quả cho bài toán hỏi đáp theo miền; đồng thời cơ chế file-based cache và fallback backend giúp hệ thống có tính khả thi cao trên hạ tầng phần cứng phổ thông.*

Từ khóa**:** *tạo sinh truy hồi tăng cường, truy hồi dày đặc, tái xếp thứ hạng, tài liệu kỹ thuật, mô hình ngôn ngữ lớn*

I. Đặt vấn đề

Trong những năm gần đây, sự phát triển vượt bậc của các mô hình ngôn ngữ lớn (Large Language Model - LLM) dựa trên kiến trúc Transformer đã tạo ra một bước ngoặt đáng kể trong lĩnh vực xử lý ngôn ngữ tự nhiên (NLP). Cơ chế self-attention cho phép Transformer mô hình

[1] Khoa Điện - Điện tử, Trường Đại học Mở Hà Nội, Hà Nội, Việt Nam

hóa hiệu quả các quan hệ phụ thuộc tầm xa trong chuỗi văn bản, từ đó trở thành nền tảng cho phần lớn các LLM hiện đại như GPT, BERT và các biến thể của chúng (Vaswani và cộng sự, 2017). Các mô hình này đã chứng minh khả năng vượt trội trong nhiều tác vụ ngôn ngữ bao gồm dịch máy, tóm tắt văn bản, trả lời câu hỏi và phân tích cảm xúc.

Tuy nhiên, khi được triển khai như các tác nhân hỏi đáp độc lập trong môi trường doanh nghiệp hoặc tổ chức, LLM bộc lộ hai hạn chế mang tính cấu trúc. Thứ nhất, tri thức của mô hình bị "đóng băng" tại thời điểm huấn luyện, dẫn đến tình trạng thông tin lỗi thời khi áp dụng vào các lĩnh vực có dữ liệu thay đổi nhanh. Thứ hai, mô hình có xu hướng sinh ra những câu trả lời có vẻ hợp lý về mặt ngôn ngữ nhưng thiếu căn cứ thực tế, hiện tượng được gọi là hallucination (ảo giác).

Trong bối cảnh triển khai theo miền chuyên biệt như hỏi đáp tài liệu kỹ thuật, quy chuẩn nghiệp vụ, hướng dẫn vận hành thiết bị hay văn bản nội bộ của tổ chức, yêu cầu về tính chính xác, khả năng truy vết nguồn và cập nhật linh hoạt còn quan trọng hơn sự trôi chảy của ngôn ngữ. Một hệ thống chỉ dựa hoàn toàn vào tham số của mô hình sẽ không thể đáp ứng được khi kho tri thức nghiệp vụ thay đổi thường xuyên hoặc khi người dùng có nhu cầu trích dẫn cụ thể về nguồn tài liệu. Retrieval-Augmented Generation (RAG) được Lewis và cộng sự (2020) đề xuất như một hướng giải quyết hiệu quả, kết hợp LLM với cơ chế truy xuất thông tin, cho phép câu trả lời được sinh ra dựa trên các đoạn văn bản liên quan đã được tìm thấy trong tập dữ liệu được cung cấp.

Kể từ khi được đề xuất, kiến trúc RAG đã được ứng dụng rộng rãi trên nhiều miền dữ liệu chuyên biệt khác nhau như y tế, giáo dục, pháp luật. Gao và cộng sự (2024) đã tổng hợp và phân loại các nghiên cứu RAG thành ba hệ hình chính: Naive RAG, Advanced RAG và Modular RAG, trong đó Naive RAG thực hiện quy trình tuyến tính indexing-retrieval-generation, Advanced RAG bổ sung các bước tiền xử lý truy vấn và hậu xử lý kết quả, còn Modular RAG cho phép tổ hợp linh hoạt các thành phần theo nhu cầu cụ thể của bài toán. Nghiên cứu này thuộc nhóm Advanced RAG với việc tích hợp bước reranking bằng Cross-Encoder sau giai đoạn truy xuất sơ bộ, đồng thời hướng tới mục tiêu triển khai trên phần cứng phổ thông mà phần lớn các nghiên cứu trước đó chưa đề cập. Các đóng góp chính của nghiên cứu bao gồm:

Đề xuất và hiện thực một pipeline RAG hoàn chỉnh cho bài toán hỏi đáp tài liệu kỹ thuật với hai pha ngoại tuyến và trực tuyến được tách biệt rõ ràng, đảm bảo tính mô-đun và khả năng mở rộng.

So sánh và phân tích các lựa chọn chunking và retrieval, đồng thời tích hợp bước reranking bằng Cross-Encoder nhằm cải thiện khả năng đưa tài liệu chính xác lên vị trí ưu tiên trong danh sách kết quả.

Đánh giá định lượng trên bộ câu hỏi có gán nhãn và chỉ ra tác động của bước xếp hạng lại đối với các chỉ số Hit@1 và MRR trong triển khai thực tế trên phần cứng phổ thông.

II. Cơ sở lý thuyết

2.1. Retrieval-Augmented Generation

RAG là mô hình kết hợp giữa truy xuất thông tin (Information Retrieval - IR) và sinh văn bản bằng LLM. Trong kiến trúc này, truy vấn của người dùng được sử dụng để tìm kiếm các đoạn văn bản liên quan từ kho tri thức bên ngoài, sau đó các đoạn được tìm thấy được đưa vào ngữ cảnh

(context) của prompt để LLM sinh câu trả lời. Điểm mạnh cốt lõi của RAG là chuyển dịch một phần tri thức từ "bộ nhớ tham số" (parametric memory) của mô hình sang "bộ nhớ không tham số" (non-parametric memory) dưới dạng kho tài liệu có thể cập nhật độc lập, nhờ đó câu trả lời bám sát nội dung tài liệu hiện hành và dễ được kiểm chứng hơn (Lewis và cộng sự, 2020).

Chất lượng đầu ra phụ thuộc vào ba mắt xích: (1) phân đoạn tài liệu (chunking); (2) chất lượng truy xuất (retrieval); và (3) đóng gói ngữ cảnh (context packaging). Các cải tiến như Self-RAG và CRAG bổ sung khả năng phản tư trên kết quả retrieval nhưng đòi hỏi tính toán lớn hơn (Asai và cộng sự, 2023; Shi và cộng sự, 2024).

2.2. Mô hình ngôn ngữ lớn

LLM là các mạng nơ-ron với hàng tỷ tham số, được huấn luyện trên khối lượng lớn dữ liệu văn bản. Kiến trúc Transformer (Vaswani và cộng sự, 2017) là nền tảng cho các LLM hiện đại như GPT-3 (Brown và cộng sự, 2020), LLaMA (Touvron và cộng sự, 2023) và Qwen (Bai và cộng sự, 2023). Các mô hình mã nguồn mở kết hợp kỹ thuật lượng tử hóa (GPTQ, AWQ) đã mở ra khả năng triển khai cục bộ trên phần cứng phổ thông.

Trong kiến trúc RAG, LLM đóng vai trò sinh văn bản ở cuối pipeline. Chất lượng đầu ra phụ thuộc vào khả năng bám sát ngữ cảnh được cung cấp. Các nghiên cứu cho thấy ngay cả mô hình nhỏ khi có ngữ cảnh chính xác vẫn sinh câu trả lời chất lượng cao, mở ra cơ hội triển khai RAG trên phần cứng hạn chế (Gao và cộng sự, 2024).

2.3. Dense retrieval và sentence embedding

Dense retrieval mã hóa câu hỏi và tài liệu vào cùng không gian vector ngữ nghĩa, cho phép tìm kiếm đoạn văn tương đồng về ý nghĩa ngay cả khi không trùng từ vựng. Dense Passage Retrieval (Karpukhin và cộng sự, 2020) đặt nền móng cho hướng tiếp cận này, vượt trội hơn BM25 trong các tác vụ hỏi đáp miền mở.

Sentence-BERT (Reimers & Gurevych, 2019) đề xuất kiến trúc Bi-Encoder cho phép sinh embedding độc lập và so sánh theo cosine similarity. Nghiên cứu này sử dụng BAAI/bge-m3 nhờ hỗ trợ đa ngôn ngữ (bao gồm tiếng Việt), cửa sổ 8.192 token và embedding 1.024 chiều (BAAI, n.d.-a).

2.4. Passage reranking bằng Cross-Encoder

Bi-Encoder có hạn chế: mỗi đoạn văn được mã hóa độc lập không có tương tác với câu hỏi. Cross-Encoder reranking (Nogueira & Cho, 2019) khắc phục bằng cách xử lý đồng thời cặp (câu hỏi, đoạn văn) như một chuỗi đầu vào duy nhất, cho phép nắm bắt tương tác sâu và gán điểm phù hợp chính xác hơn.

Chiến lược hai giai đoạn kết hợp Bi-Encoder (retrieve) và Cross-Encoder (rerank) đã được xác nhận là hiệu quả trong nhiều nghiên cứu và ứng dụng thực tế. Mô hình BAAI/bge-reranker-v2-m3 được sử dụng trong nghiên cứu này kế thừa kiến trúc này với khả năng hỗ trợ đa ngôn ngữ, phù hợp với tài liệu kỹ thuật tiếng Việt (BAAI, n.d.-b).

III. Phương pháp, vật liệu nghiên cứu

3.1. Kho tri thức và bộ câu hỏi đánh giá

Kho tri thức gồm ba tài liệu kỹ thuật tiếng Việt về công nghệ 5G (~9,2 MB): đặc tả kiến trúc 5G NR, hướng dẫn triển khai trạm gốc gNodeB, và công nghệ ảo hóa mạng lõi 5G Core. Bộ 25 câu hỏi có

gán nhãn expected_doc_id được xây dựng để đánh giá, bao phủ các dạng: định nghĩa khái niệm, mô tả chức năng, câu hỏi kỹ thuật định lượng và phân biệt khái niệm tương đồng.

Hai cấu hình retrieval được so sánh trực tiếp: (i) dense retrieval đơn thuần không có bước xếp hạng lại và (ii) dense retrieval được bổ sung bước reranking bằng Cross-Encoder. Các tham số cấu hình được giữ cố định xuyên suốt thực nghiệm để đảm bảo tính so sánh: chunk_size = 800 ký tự, chunk_overlap = 160 ký tự, mô hình embedding BAAI/bge-m3, và mô hình reranking BAAI/bge-reranker-v2-m3.

3.2. Chỉ số đánh giá

Bốn chỉ số đánh giá được sử dụng: Hit@k đo tỷ lệ câu hỏi mà tài liệu đúng xuất hiện trong top-k kết quả truy xuất; Mean Reciprocal Rank (MRR) tính trung bình nghịch đảo thứ hạng của tài liệu đúng đầu tiên qua tập câu hỏi, phản ánh mức độ ổn định của hệ thống trong việc đẩy tài liệu đúng lên vị trí cao; và thứ hạng trung bình của tài liệu đúng cung cấp thêm góc nhìn trực quan về chất lượng xếp hạng. Bảng 1 tổng hợp thống kê xây dựng chỉ mục cho hai biến thể chunking:

Bảng 1. Thống kê kho tri thức và chỉ mục của hai biến thể chunking

Biến thể	Số chunk	Thời gian (s)	Số chiều vector	Batch size
Char-based	938	413,5	1024	8 (CUDA)
Markdown-aware	1.109	454,7	1024	8 (CUDA)

Biến thể markdown-aware tạo ra số chunk nhiều hơn 18,2% so với char-based (1.109 so với 938) do giữ lại các ranh giới cấu trúc của tài liệu như tiêu đề và mục con. Chênh lệch thời gian build khoảng 41,2 giây là hoàn toàn chấp nhận được trong kịch bản xử lý ngoại tuyến, đặc biệt khi mục tiêu là bảo toàn tốt hơn ngữ cảnh logic trong các tài liệu kỹ thuật có cấu trúc phân cấp dày đặc.

IV. Kiến trúc hệ thống và phương pháp đề xuất

4.1. Kiến trúc tổng thể

Hệ thống được tổ chức theo ba lớp chức năng độc lập có giao tiếp rõ ràng. Lớp giao diện bao gồm giao diện chat cho người dùng cuối và giao diện quản trị để nạp, xóa tài liệu và kích hoạt tái lập chỉ mục. Lớp quản lý chỉ mục và cache chịu trách nhiệm đọc tài liệu nguồn, làm sạch dữ liệu, phân đoạn, sinh vector nhúng và lưu trữ các artifact xử lý trung gian. Lớp pipeline RAG hoạt động theo thời gian thực để tiếp nhận câu hỏi, thực hiện truy xuất, xếp hạng lại, xây dựng ngữ cảnh và gọi mô hình sinh câu trả lời.

Việc phân tách rõ ràng giữa pha ngoại tuyến và pha trực tuyến là quyết định kiến trúc trung tâm của hệ thống. Các tác vụ tốn kém về tính toán như phân đoạn văn bản, sinh embedding và lập chỉ mục được đẩy hoàn toàn sang pha ngoại tuyến, trong khi pha trực tuyến chỉ cần tải embedding đã tính sẵn vào bộ nhớ và thực hiện các phép tính ma trận tối ưu hóa.

Thay vì phụ thuộc vào một vector database đầy đủ tính năng ngay từ đầu, hệ thống sử dụng thiết kế file-based cache: kho tri thức được lưu trong thư mục dữ liệu tệp, còn các kết quả trung gian như corpus đã phân đoạn, ma trận embedding và metadata chỉ mục được serialized và lưu dưới dạng file nhị phân. Cách tiếp cận này cho phép triển khai nhanh trên máy

tính cá nhân, thuận tiện sao chép và dễ tái lập thực nghiệm. Để đảm bảo tính toàn vẹn dữ liệu, hệ thống sử dụng cơ chế ghi nguyên từ thông qua file tạm trước khi thay thế file chính thức, ngăn ngừa rủi ro hỏng dữ liệu trong trường hợp gián đoạn đột ngột.

So với kiến trúc RAG truyền thống (Naive RAG) vốn yêu cầu vector database chuyên dụng (như Pinecone, Weaviate) và hạ tầng GPU mạnh cho cả embedding lẫn generation, kiến trúc đề xuất trong nghiên cứu này có ba điểm khác biệt cốt lõi giúp triển khai trên phần cứng phổ thông. Thứ nhất, việc thay thế vector database bằng file-based cache với ma trận embedding serialized loại bỏ hoàn toàn phụ thuộc vào dịch vụ cơ sở dữ liệu vector, giảm đáng kể yêu cầu bộ nhớ và độ phức tạp vận hành. Thứ hai, chiến lược phân tách pha ngoại tuyến và trực tuyến cho phép các tác vụ nặng về tính toán (sinh embedding, lập chỉ mục) chỉ thực hiện một lần khi dữ liệu thay đổi, trong khi pha trực tuyến chỉ cần các phép nhân ma trận tối ưu bằng NumPy/ BLAS. Thứ ba, cơ chế fallback tự động từ GPU sang CPU và chiến lược giảm dần batch size khi phát hiện lỗi tràn bộ nhớ đảm bảo hệ thống hoạt động ổn định ngay cả trên GPU có VRAM hạn chế (4 GB). Hình 1 mô tả sơ đồ tổng thể kiến trúc hệ thống với ba lớp chức năng và luồng dữ liệu giữa pha ngoại tuyến và pha trực tuyến.

4.2. Pha ngoại tuyến: Ingestion, chunking và indexing

Trong pha ngoại tuyến, tài liệu PDF được trích xuất thành văn bản thô thông qua các thư viện xử lý PDF, sau đó trải qua bước làm sạch để loại bỏ ký tự thừa, chuẩn hóa khoảng trắng và xử lý các đặc thù định dạng. Văn bản đã làm sạch được đưa qua một trong hai chiến lược phân đoạn. Chiến lược thứ nhất (char-based) chia văn bản theo cửa sổ ký tự có kích thước cố định (chunk_size = 800) với độ chồng lấn kiểm soát được (chunk_overlap = 160) nhằm duy trì ngữ cảnh liên tiếp qua các ranh giới đoạn. Chiến lược thứ hai (markdown-aware) nhận diện tiêu đề phân cấp, mục con và danh sách theo heuristic gần với cú pháp markdown, ưu tiên cắt tại ranh giới cấu trúc logic của tài liệu thay vì tại ranh giới ký tự tùy ý.

Sau khi phân đoạn, từng chunk được đưa qua mô hình bge-m3 để sinh embedding 1.024 chiều. Kết quả embedding được chuẩn hóa về vector đơn vị (L2-normalization) để tính độ tương đồng bằng tích vô hướng (dot product), phép toán tương đương cosine similarity nhưng có chi phí tính toán thấp hơn. Bước sinh embedding được xử lý theo batch với cơ chế điều chỉnh kích thước batch khi phát hiện lỗi tràn bộ nhớ GPU và fallback sang CPU nếu cần, đảm bảo hoàn thành tác vụ trên nhiều loại phần cứng khác nhau.

Metadata quản lý chỉ mục bao gồm schema version, hash của tập tài liệu trong kho tri thức, và tham số cấu hình build như phương pháp chunking, kích thước đoạn và tên mô hình embedding. Cơ chế kiểm tra hash đảm bảo bất kỳ thay đổi nào trong dữ liệu đầu vào hoặc cấu hình build đều làm chỉ mục cũ mất hiệu lực và kích hoạt re-index tự động, ngăn chặn nguy cơ người dùng vô tình truy vấn trên chỉ mục lỗi thời.

4.3. Pha trực tuyến: Retrieval, reranking và generation

Khi nhận câu hỏi, hệ thống mã hóa câu hỏi thành vector embedding bằng cùng mô hình đã sử dụng trong pha ngoại tuyến, đảm bảo tính nhất quán của không

gian biểu diễn. Hệ thống hỗ trợ ba chế độ truy xuất có thể cấu hình. Dense retrieval tính toán tích vô hướng giữa vector câu hỏi và toàn bộ ma trận embedding của corpus thông qua phép tính song song tối ưu bởi NumPy, sau đó lấy top-k ứng viên có điểm cao nhất. Keyword retrieval thực hiện một baseline đếm tần suất token để duy trì lợi thế với các truy vấn chứa thuật ngữ hiếm hoặc mã kỹ thuật đặc thù mà không có từ đồng nghĩa rõ ràng trong không gian ngữ nghĩa. Hybrid retrieval hợp nhất kết quả của hai chế độ trên, chuẩn hóa điểm về cùng thang và tính điểm kết hợp theo công thức:

$$S_{hybrid} = \alpha \cdot S_{dense} + (1 - \alpha) \cdot S_{keyword}$$

với α mặc định bằng 0,6, ưu tiên nhẹ cho kết quả ngữ nghĩa. Sau khi có danh sách ứng viên từ bước retrieval, hệ thống thực hiện xếp hạng lại bằng mô hình Cross-Encoder bge-reranker-v2-m3. Mô hình này xử lý đồng thời cặp (câu hỏi, chunk) như một chuỗi đầu vào nối tiếp và gán điểm phù hợp dựa trên sự tương tác ngữ nghĩa sâu giữa hai văn bản. Các đoạn văn có điểm rerank dưới ngưỡng quy định bị loại bỏ; phần còn lại được sắp xếp lại và chọn top-k cuối cùng để đưa vào prompt. Cách phân cấp hai giai đoạn này cân bằng hiệu quả giữa tốc độ lọc sơ bộ của Bi-Encoder và độ chính xác tinh chỉnh của Cross-Encoder.

4.4. Sinh câu trả lời và kiểm soát vận hành

Ngữ cảnh được đóng gói thành các khối có cấu trúc [Source i] kèm metadata gồm tên tài liệu nguồn, số trang và định danh đoạn. Prompt hệ thống đặt ra hai ràng buộc rõ ràng: câu trả lời phải ưu tiên bám sát các đoạn ngữ cảnh đã truy xuất, và phải từ chối trả lời khi ngữ cảnh không cung cấp đủ bằng chứng thay vì tự tổng hợp tri thức từ tham số mô hình. Chiến lược này trực tiếp hạn chế hiện tượng hallucination bằng cách thu hẹp không gian sinh văn bản của mô hình về phạm vi nội dung tài liệu được cung cấp.

Hệ thống hỗ trợ hai backend sinh văn bản có thể chuyển đổi linh hoạt. Backend cục bộ sử dụng mô hình Qwen2.5-7B-Instruct phiên bản lượng tử hóa 4-bit thông qua Ollama, cho phép vận hành trên GPU có VRAM 4 GB mà vẫn duy trì chất lượng sinh văn bản tiếng Việt chấp nhận được, đồng thời đảm bảo bảo mật dữ liệu nhạy cảm do toàn bộ xử lý diễn ra cục bộ. Backend đám mây sử dụng Gemini 1.5 Flash thông qua Google Gemini API để xử lý các truy vấn đòi hỏi suy luận phức tạp hoặc ngữ cảnh dài, tận dụng cửa sổ ngữ cảnh lên đến một triệu token của mô hình này. Trong chế độ tự động, một router lựa chọn backend phù hợp dựa trên cấu hình; nếu backend được ưu tiên gặp lỗi hoặc không khả dụng, yêu cầu được tự động chuyển sang backend còn lại (fallback). Cả bước embedding lẫn reranking đều được trang bị cơ chế giảm dần batch size khi phát hiện lỗi CUDA out-of-memory và fallback sang CPU, đảm bảo hệ thống hoàn thành tác vụ ngay cả trong điều kiện tài nguyên GPU hạn chế.

V. Kết quả và thảo luận

5.1. Kết quả

Bảng 2 trình bày kết quả đánh giá retrieval, là tập hợp các chỉ số quan trọng nhất đối với hệ thống RAG vì chất lượng truy xuất quyết định trực tiếp chất lượng ngữ cảnh đưa vào LLM. Nếu tài liệu đúng không xuất hiện trong top đầu, mô hình sinh văn bản khó có thể tạo ra câu trả lời chính xác và có căn cứ dù bản thân nó sở hữu năng lực ngôn ngữ tốt.

Bảng 2. So sánh chất lượng retrieval giữa hai cấu hình

Phương án	Hit@1	Hit@3	MRR	Thứ hạng TB
Dense retrieval	84,0%	100,0%	0,913	1,20
Dense + Reranking (đề xuất)	92,0%	100,0%	0,960	1,08

Kết quả cho thấy cả hai cấu hình đều đạt Hit@3 bằng 100%, nghĩa là với bộ câu hỏi thực nghiệm hiện tại, dense retrieval đã đủ khả năng đưa tài liệu đúng vào trong ba kết quả đầu tiên một cách nhất quán. Tuy nhiên, sự khác biệt có ý nghĩa thực tiễn quan trọng nằm ở Hit@1 và MRR. Khi bổ sung bước reranking, Hit@1 tăng từ 84% lên 92%, tương ứng mức cải thiện tuyệt đối 8 điểm phần trăm. Đồng thời, MRR tăng từ 0,913 lên 0,960 và thứ hạng trung bình giảm từ 1,20 xuống 1,08. Điều này chứng minh rằng Cross-Encoder reranker không nhất thiết mở rộng khả năng tìm thấy tài liệu đúng trong top-k rộng, mà cải thiện rõ rệt khả năng xếp tài liệu đúng lên đúng vị trí đầu tiên.

5.2. Thảo luận

Ý nghĩa thực tiễn của cải thiện Hit@1 rất rõ ràng trong bối cảnh hệ thống RAG. Trong nhiều cấu hình triển khai thực tế, prompt chỉ lấy một hoặc vài nguồn đầu bảng để tiết kiệm token và giảm nhiễu ngữ cảnh. Vì vậy, việc kéo tài liệu đúng từ vị trí thứ hai hoặc thứ ba lên vị trí thứ nhất thường tạo ra tác động đến chất lượng câu trả lời cuối cùng lớn hơn so với việc đơn giản mở rộng top-k. Cross-Encoder phát huy vai trò ở đây vì nó xem xét trực tiếp sự tương tác ngữ nghĩa giữa câu hỏi và từng đoạn văn riêng biệt, từ đó loại bỏ được các ứng viên có chủ đề chung gần nhưng không thực sự trả lời đúng trọng tâm của truy vấn cụ thể.

Thực nghiệm cũng gợi ý rằng chi phí bổ sung của bước reranking là hợp lý trong bài toán này. Reranking chỉ được áp dụng trên một tập nhỏ ứng viên đã được lọc sơ bộ bởi dense retrieval (thường từ 10 đến 20 chunk), nên độ trễ thêm vào là chấp nhận được trong môi trường hỏi đáp tương tác không yêu cầu phản hồi trong micro-giây. Tỷ lệ chi phí - lợi ích này là một trong những lý do khiến chiến lược retrieve-then-rerank trở nên phổ biến trong các hệ thống RAG sản xuất.

VI. Kết luận

Bài báo đã trình bày thiết kế và hiện thực một hệ thống chatbot hỏi đáp tài liệu kỹ thuật theo kiến trúc Retrieval-Augmented Generation, trong đó toàn bộ pipeline từ ingestion, chunking, embedding, retrieval, reranking đến generation được hiện thực đầy đủ và đánh giá trên dữ liệu thực nghiệm. Kiến trúc hai pha, cơ chế file-based cache và chiến lược fallback backend hướng tới cân bằng giữa độ chính xác, khả năng truy vết nguồn và tính khả thi triển khai trên phần cứng phổ thông.

Kết quả thực nghiệm xác nhận rằng dense retrieval với mô hình BAAI/bge-m3 đã cung cấp nền tảng truy xuất tốt (Hit@3 = 100%), trong khi bước reranking bằng Cross-Encoder BAAI/bge-reranker-v2-m3 cải thiện đáng kể chất lượng xếp hạng vị trí đầu (Hit@1 từ 84% lên 92%, MRR từ 0,913 lên 0,960).

Định hướng phát triển trong tương lai bao gồm ba hướng chính: (1) tích hợp vector database chuyên dụng (Faiss, Qdrant) và thuật toán BM25 để nâng cao khả năng mở rộng và chất lượng keyword retrieval; (2) cải thiện quy trình tiền xử lý tài liệu với OCR chất lượng cao và

phân tích cấu trúc bảng biểu để xử lý tốt hơn các tài liệu phức tạp; và (3) phát triển cơ chế quản lý ngữ cảnh hội thoại nhiều lượt với memory buffer để hỗ trợ các phiên làm việc mang tính thảo luận sâu. Những cải tiến này sẽ đưa hệ thống tiến gần hơn đến một nền tảng trợ lý hỏi đáp tài liệu có thể triển khai ở quy mô tổ chức.

Tài liệu tham khảo

Asai, A., Wu, Z., Wang, Y., Sil, A., & Hajishirzi, H. (2023). *Self-RAG: Learning to retrieve, generate, and critique through self-reflection.*

arXiv. https://arxiv.org/abs/2310.11511

BAAI. (n.d.-a). bge-m3. Hugging Face. https://huggingface.co/BAAI/bge-m3

BAAI. (n.d.-b). bge-reranker-v2-m3. Hugging Face. https://huggingface.co/BAAI/bge-reranker-v2-m3

Bai, J., Bai, S., Chu, Y., Cui, Z., Dang, K., Deng, X., Fan, Y., Ge, W., Han, Y., Huang, F., Hui, B., Ji, L., Li, M., Lin, J., Lin, R., Liu, D., Liu, G., Lu, C., Lu, K., ... Zhou, J. (2023). Qwen technical report. *arXiv*. https://arxiv.org/abs/2309.16609

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., ... Amodei, D. (2020). Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, 33, 1877-1901. https://papers.nips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html

Karpukhin, V., Oguz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., & Yih, W.-t. (2020). Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 6769-6781). Association for Computational Linguistics. https://aclanthology.org/2020.emnlp-main.550/

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Riedel, S., & Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, 33, 9459-9474. https://arxiv.org/abs/2005.11401

Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., Dai, Y., Sun, J., Wang, M., & Wang, H. (2024). Retrieval-augmented generation for large language models: A survey. *arXiv*. https://arxiv.org/abs/2312.10997

Nogueira, R., & Cho, K. (2019). Passage re-ranking with BERT. *arXiv*. http://arxiv.org/abs/1901.04085

Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)* (pp. 3982-3992). Association for Computational Linguistics. https://arxiv.org/abs/1908.10084

Shi, W., Gururangan, S., & Zettlemoyer, L. (2024). REPLUG: Retrieval-augmented language model pre-training. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL 2024)* (pp. 6564-6575). Association for Computational Linguistics. https://aclanthology.org/2024.naacl-long.361

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., & Lample, G. (2023). LLaMA: Open and efficient foundation language models. *arXiv*. https://arxiv.org/abs/2302.13971

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems*, 30. https://papers.nips.cc/paper/7181-attention-is-all-you-need.

DEVELOPING A TECHNICAL DOCUMENT QUESTION-ANSWERING CHATBOT USING RETRIEVAL-AUGMENTED GENERATION

Dang Khanh Hoa[1], Pham Thi Thu Trang[1], Pham Minh Viet[1], Nguyen Thanh Nghi[1]

Abstract: *This paper presents the design and implementation of a technical document question-answering chatbot based on Retrieval-Augmented Generation (RAG) to reduce hallucinations and improve source grounding. The proposed system follows a two-phase pipeline: offline ingestion, cleaning, chunking, embedding, and indexing; and online retrieval, reranking, context construction, and answer generation using a large language model. We implement and compare character-based and markdown-aware chunking strategies, three retrieval modes (dense, keyword, hybrid), the BAAI/bge-m3 multilingual embedding model, the BAAI/bge-reranker-v2-m3 reranker, and dual LLM backends (local Qwen2.5-7B-Instruct via Ollama and cloud-based Gemini 1.5 Flash). On a 25-question labeled benchmark over 5G telecommunications documents, dense retrieval achieves Hit@1 = 84% and MRR = 0.913; adding reranking improves Hit@1 to 92% and MRR to 0.960 while preserving Hit@3 at 100%. Results confirm that semantic retrieval combined with Cross-Encoder reranking is an effective strategy for domain-specific question answering, and that a lightweight file-based cache architecture supports robust deployment on commodity hardware.*

Keywords: *retrieval-augmented generation, dense retrieval, reranking, technical document, large language model*

[1] Faculty of Electric and Electronic Engineering, Hanoi Open University, Hanoi, Vietnam