ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

XÂY DỰNG HỆ THỐNG HỎI ĐÁP SỬ DỤNG RAG VỀ ĐẠI HỌC QUỐC GIA HÀ NỘI

BÁO CÁO BÀI TẬP LỚN
ỨNG DỤNG TRÍ TUỆ NHÂN TẠO CHO NGÔN NGỮ

Ngành : Trí Tuệ Nhân Tạo

Sinh viên: Dương Phương Hiểu - 22022659

. Phạm Long Nhật - 22022520

. Trần Kim Dũng - 22022633

. Hồ Hà Ngọc Nhất - 22022626

Giảng viên hướng dẫn: PGS.TS. Nguyễn Phương Thái

. : TS. Trần Hồng Việt

HÀ NỘI, 2025

Mục lục

Danh sách hình vẽ

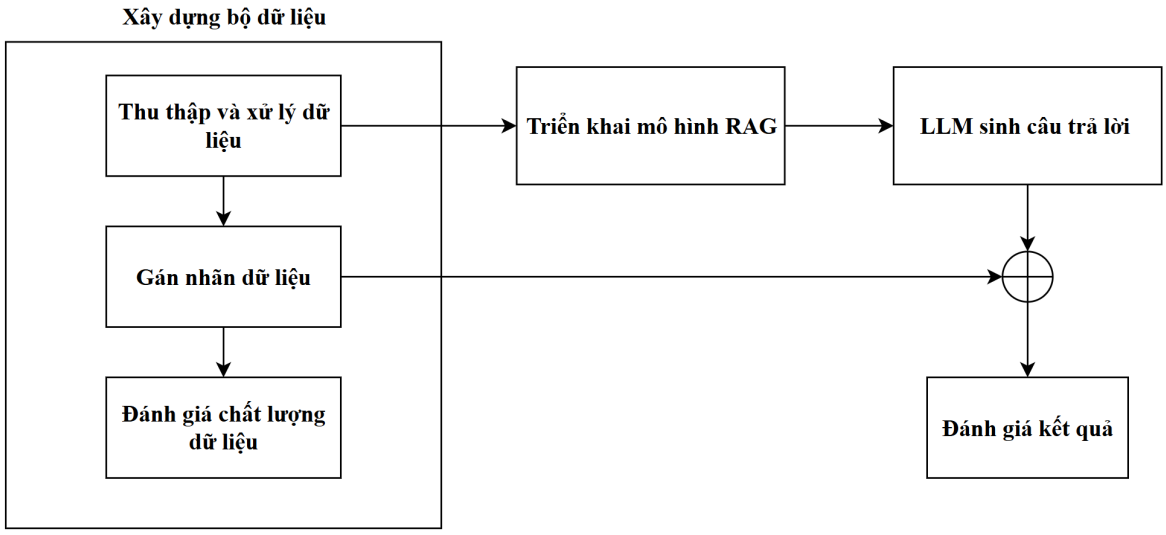
Danh sách bảng

Chương 1

Giới thiệu

Trong bối cảnh chuyển đổi số ngày nay, lưu lượng truy cập về Đại học Quốc gia Hà Nội nói chung và Trường đại học Công Nghệ nói riêng ngày càng tăng nhanh, đặc biệt là các sinh viên, giảng viên của trường muốn tìm kiếm các thông tin về các hoạt nghiên cứu, học bổng hay như các em học sinh, các phụ huynh đang muốn tìm hiểu các thông tin về quy chế tuyển sinh, môi trường học tập tại Đại học Quốc gia Hà Nội để lựa chọn nguyện vọng phù hợp cho kỳ thi trung học phổ thông quốc gia sắp tới. Việc tra cứu các thông tin này trên website của trường nhìn chung khá khó khăn vì lượng thông tin khá lớn và cần nhiều thao tác, điều này dẫn đến các nhu cầu về việc xây dựng các hệ thống hỏi đáp (chatbot) có thể trả lời các câu hỏi về trường chỉ bằng cách thực hiện đặt câu hỏi và nhận câu trả lời. Nhưng các hệ thống hỏi đáp truyền thống chỉ dựa vào các kiến thức đã được học trong tham số của mô hình tỏ ra không phù hợp bởi vì không thể đáp ứng được khi lượng thông tin mới xuất hiện liên tục.

Vì thế, trong báo cáo này, chúng em thực hiện xây dựng một hệ thống RAG cho nhiệm vụ hỏi đáp về Đại học Quốc gia Hà Nội và Trường đại học Công Nghệ. Hệ thống RAG này cho phép sử dụng các kiến thức trong kho dữ liệu về trường đã thu thập làm ngữ cảnh để trả lời câu hỏi, giúp đưa ra câu trả lời chính xác hơn và dễ dàng nắm bắt các thông tin mới. Chúng em xây dựng hệ thống trên theo một quy trình chuẩn từ thu thập và xây dựng bộ dữ liệu, triển khai mô hình RAG cơ bản kết hợp với các kỹ thuật tối ưu truy vấn như Rerank, Multi-query, few-shot learning, sau đó sinh ra câu trả lời với các mô hình ngôn ngữ lớn và cuối cùng là đánh giá kết quả, so sánh các phương pháp với nhau. Dự án giúp chúng em phát triển một hệ thống hỏi–đáp cơ bản về trường, bước đầu cho kết quả khả quan và có thể tiếp tục tối ưu



Hình 1.0.1: Tổng quan luồng xây dựng hệ thống

Chương 2

Xây dựng bộ dữ liệu

2.1 Lựa chọn nguồn dữ liệu

- **Nguồn tài liệu:** Tài liệu được xây dựng bằng cách thu thập thông tin từ các nguồn web công khai, bao gồm:
  - Trang chủ của Đại học Quốc gia Hà Nội (VNU), Domain chính là vnu.edu.vn trong đó bao gồm cả các subdomain của các trường thành viên ví dụ như ulis.vnu.edu.vn, education.vnu.edu.vn , …
  - Trang chủ của Trường Đại học Công nghệ (UET), thuộc VNU.
  - Sổ tay sinh viên …
- **Tiêu chí lựa chọn:** Việc lựa chọn các nguồn này nhằm mục đích xây dựng một cơ sở kiến thức tập trung vào thông tin liên quan đến VNU và UET

2.2 Cách thu thập và xử lý dữ liệu

- **Phương pháp trích xuất:** Dữ liệu thô (nội dung web) được trích xuất thông qua quy trình ”crawling”tự động.
- **Công cụ sử dụng:**
  1. Các script Python tùy chỉnh được phát triển trong module src/crawlers (ví dụ: $crawl\_vnu\_home()$, $crawl\_uet\_home()$, $crawl\_handbook()$) chịu trách nhiệm thực hiện việc thu thập dữ liệu.
  2. Các script này sử dụng thư viện Scrapy của Python để tương tác với các trang web và tải về nội dung.
  3. Toàn bộ quy trình được điều phối bởi script $run\_pipeline.py$, cho phép lựa chọn nguồn dữ liệu và các bước xử lý cụ thể.

- **Quy trình xử lý tiếp theo:** Sau khi thu thập, dữ liệu thô được lưu trữ trong thư mục Tiếp theo, dữ liệu trải qua các bước:

Hình 2.2.1: Dữ liệu thô

  - **Làm sạch dữ liệu** (thực hiện bởi clean_all_data() trong src/processors): Bao gồm các thao tác như xóa bỏ thẻ HTML, chuẩn hóa văn bản, và lọc bỏ nội dung không cần thiết. Dữ liệu sau làm sạch được lưu ở data/cleaned/.
  - **Tạo Chunks** (thực hiện bởi $process\_all()$ trong src/processors: Văn bản đã làm sạch được chia thành các đoạn (chunks) nhỏ hơn, kèm theo metadata. Các chunks này được lưu ở data/chunks/ dưới dạng tệp JSON

Hình 2.2.2: Chia chunk dữ liệu

2.3 Gán nhãn dữ liệu

Từ các chunks đã được chia như ở phần trên, chúng em lập khoảng 188 câu hỏi và câu trả lời cho phần test và khoảng 3 câu hỏi cho phần train (phần train ở đây chúng em

dùng làm ví dụ cho kỹ thuật few-shot learning). Một số ví dụ của bộ câu hỏi câu trả lời như sau:

1. Q: ”Đại học Quốc gia Hà Nội được thành lập năm nào?” A: ”1993”
2. Q: ”Trụ sở chính của Đại học Quốc gia Hà nội nằm ở đâu?” A: ”Hòa Lạc, huyện Thạch Thất, thành phố Hà Nội.”

Về tiêu chí để lựa chọn và tạo bộ test và train, thì bộ câu hỏi và câu trả lời phải đảm bảo sự đa dạng về các chủ đề liên quan và để đánh giá một cách khách quan khả năng trả lời câu hỏi của hệ thống RAG qua các dạng câu hỏi khác nhau.

Dựa trên những tiêu chí đó, chúng em thực hiện gán nhãn bộ dữ liệu. Đầu tiên, từ mỗi đoạn chunks ở trên sẽ chọn ra những thông tiin mang tính đóng góp cho việc hiểu biết thêm về Đại học Quốc gia Hà Nội cũng như trường Đại học Công Nghệ, đồng thời loại bỏ những thông tin thừa thãi không liên quan. Sau đó, từ những thông tin đã lọc ra đó sẽ tạo ra một hoặc nhiều cặp câu hỏi và câu trả lời tùy vào lượng thông tin đó. Các câu trả lời được gán nhãn được đảm bảo tối giản nhất có thể, tránh các từ không liên quan để mô hình có thể đánh giá một cách hiệu quả hơn. Quá trình xây dựng được thực hiện hoàn toàn thủ công bởi các thành viên trong nhóm, đảm bảo độ chính xác và tính phù hợp của từng cặp câu hỏi.

2.4 Đánh giá chất lượng dữ liệu IAA

Ngoài ra, để đảm bảo bộ dữ liệu đã gán nhãn có chất lượng tốt và chính xác, chúng em đã thực hiện gán nhãn độc lập qua 2 thành viên, chọn ra 30 câu hỏi chung và mỗi bạn sẽ gán nhãn theo ý của bản thân để đưa ra các câu trả lời cho 30 câu hỏi này. Kết quả thu được là giá trị exact_match tức là số lượng câu trả lời giống nhau giữa 2 bạn là 19/30 câu. Các chỉ số có giá trị là Recall = 0.91, Precision = 0.96, F1-score = 0.92. Dựa trên kết quả trên, chúng em đánh giá rằng bộ dữ liệu được gán nhãn có độ tin cậy tốt và phù hợp để sử dụng cho các bước xử lý tiếp theo như huấn luyện mô hình hoặc phân tích thống kê.

Chương 3

Phương pháp

Trong phần này, chúng em triển khai và so sánh nhiều biến thể của hệ thống RAG với các kỹ thuật tối ưu hoá nhằm lựa chọn cấu hình mang lại hiệu quả cao nhất. Cụ thể, chúng em xây dựng hệ thống RAG cơ bản, sau đó lần lượt tích hợp các kỹ thuật như Rerank, Multi-query và Few-shot learning để cải thiện hiệu suất truy xuất và sinh câu trả lời.

3.1 Hệ thống RAG cơ bản

Một hệ thống RAG cơ bản bao gồm 3 thành phần chính là chunking (chia dữ liệu), Vector database để lưu trữ dữ liệu được embedding, và retriever (truy vấn dữ liệu). Dựa trên quy trình đó, chúng em thực hiện chia các tài liệu (documents) từ bộ dữ liệu thành các chunk, việc này giúp chia nhỏ chia nhỏ các phần thông tin giúp khả năng truy vấn hiệu quả hơn. Sau nhiều lần thử nghiệm và tham khảo các dự án đã có, chúng em quyết định lựa chọn độ dài cho mỗi đoạn (chunk) là 512 và độ dài overlap là 100 nhằm đảm bảo giữ lại được thông tin và không bị rời rạc giữa các đoạn. Quá trình chia chunk thực hiện đệ quy dựa trên các ký tự phân tách như , , khoảng trắng, giúp giữ được sự liên kết giữa các đoạn. Sau khi chia dữ liệu thành các đoạn nhỏ (chunk), chúng em sử dụng mô hình "BAAI/bge-m3" để chuyển các đoạn này thành vector embedding. Mô hình này được lựa chọn nhờ khả năng hỗ trợ đa ngôn ngữ, xử lý được đầu vào lên tới 8192 token và có chất lượng embedding tốt. Ngoài ra, với khoảng 560 triệu tham số, mô hình vẫn có thể chạy được trên phần cứng không quá mạnh. Và để lưu trữ các vector embedding này, chúng em lựa chọn ChromaDB, đây là một vector database dùng để lưu trữ dữ liệu vector một cách hiệu quả và khá phổ biến. Trong quá trình thực hiện, chúng em có so sánh và cân nhắc sử dụng FAISS, nhưng để dễ sử dụng và tích hợp với dự án có quy mô không quá lớn như hiện tại, ChromaDB vẫn là lựa chọn hàng đầu

Cuối cùng, sau khi đã xây dựng vector database, quá trình truy vấn được thực hiện bằng cách chuyển câu hỏi của người dùng thành dạng vector và so sánh giữa vector này với các vector embedding trong database để tìm ra các chunk có độ tương đồng cao nhất

với câu hỏi, ở đây, chúng em lựa chọn top 10 chunk liên quan nhất và mặc định sử dụng độ tương đồng so sánh là cosine similarity.

3.2 Chiến lược Reranking và Multi-query

- **Reranking**: Việc chỉ thực hiện chọn ra 10 chunk có liên quan nhất bằng cách dùng độ tương đồng cosine similarity thực chất là tìm ra các đoạn chunk có embedding gần nhất với câu hỏi trong không gian vector, nhưng thực chất nó không thực sự chính xác để đánh giá sự phù hợp sâu sắc về mặt ngữ nghĩa giữa câu hỏi và đoạn văn đó. Vì thế kỹ thuật Reranking đã được chúng em áp dụng để giải quyết vấn đề trên, thay vì sử dụng trực tiếp top k câu trả lời được sinh ra từ quá trình truy vấn, nó sẽ thực hiện đánh giá lại điểm tương đồng giữa các chunk và câu hỏi về mặt ngữ nghĩa và sắp xếp lại thứ tự ưu tiên của chúng. Mô hình được sử dụng để thực hiện Reranking là BAAI/bge-reranker-base là một mô hình khá phổ biến với khoảng 278 triệu tham số. 10 chunk có điểm cao nhất sẽ được dùng làm ngữ cảnh để đưa vào cho mô hình ngôn ngữ lớn để sinh ra câu trả lời.
- **Multi-query**: Kết hợp với kỹ thuật Reranking, chúng em xây dựng thêm một hệ thống RAG áp dụng thêm một kỹ thuật tối ưu truy vấn là Multi-query, kỹ thuật này tận dụng việc sẽ có một mô hình để đánh giá lại điểm tương đồng giữa câu hỏi và câu trả lời sau quá trình retriever, vì thế dựa trên câu hỏi ban đầu sẽ đi qua một mô hình ngôn ngữ để tạo ra các phiên bản khác nhau của câu hỏi và từ đó lấy ra các ngữ cảnh một cách đa dạng. Cuối cùng thực hiện Rerank lại các ngữ cảnh này với câu hỏi ban đầu để được top ngữ cảnh liên quan nhất. Ở đây, chúng em thực hiện tạo ra 3 câu hỏi dựa trên ngữ cảnh, mỗi câu hỏi thực hiện lấy 10 truy vấn ban đầu sau đó kết hợp với 10 truy vấn của câu hỏi ban đầu để tạo thành một tập ngữ cảnh, cuối cùng thực hiện Rerank trên tập này để lấy ra 10 ngữ cảnh phù hợp nhất với câu hỏi gốc. Nhưng việc sử dụng một mô hình LLM để tạo ra các câu hỏi trong kỹ thuật này cũng khiến cho tốc độ sinh ra kết quả của mô hình tăng lên.

3.3 Áp dụng Few-shot learning

Sau khi đã áp dụng các kỹ thuật liên quan đến tối ưu hóa truy vấn của RAG. Một phần cũng rất quan trọng giúp tăng hiệu quả của các mô hình ngôn ngữ lớn đó là tinh chỉnh Prompt. Một Prompt hiệu quả giúp mô hình ngôn ngữ lớn hiểu được ngữ cảnh của RAG và đưa ra các câu trả lời theo mong muốn của lập trình viên. Cụ thể trong hệ thống trả lời câu hỏi này, chúng em mong muốn câu trả lời từ hệ thống phải ngắn gọn và trực tiếp không dài dòng, vì thế ngoài việc hướng dẫn cho mô hình biết phải đưa ra câu trả lời chính xác và ngắn gọn, em sử dụng thêm kỹ thuật Few-shot learning, tức là sẽ đưa thêm một vài hướng dẫn về các câu hỏi và câu trả lời có đặc điểm mong muốn. (Hình 2) Việc này giúp cho mô hình biết được rằng dựa trên context và câu hỏi đầu vào thì nó phải đưa

```
template = """Dựa trên ngữ cảnh, chỉ cung cấp câu trả lời 1 lần 1 cách trực tiếp chỉ nằm trên 1 dòng dù không hoàn chỉnh, ngắn gọn nhất cho câu hỏi.
            Không thêm từ ngữ thừa, giới thiệu hay giải thích. Trả lời bằng tiếng Việt.
            Ví dụ 1:
            Câu hỏi: Viện Trần Nhân Tông được thành lập năm nào?
            Câu trả lời: Năm 2016.

            Ví dụ 2:
            Câu hỏi: Trung tâm Hỗ trợ Sinh viên của Đại học Quốc gia Hà Nội trước đây có tên là gì?
            Câu trả lời: Trung tâm Nội trú Sinh viên.

            Đến lượt bạn:
            ngữ cảnh: {context}
            Câu hỏi: {query}
            Câu trả lời:
            """
```

Hình 3.3.1: Ví dụ về Few-shot learning

ra câu trả lời có cấu trúc như câu trả lời ví dụ.

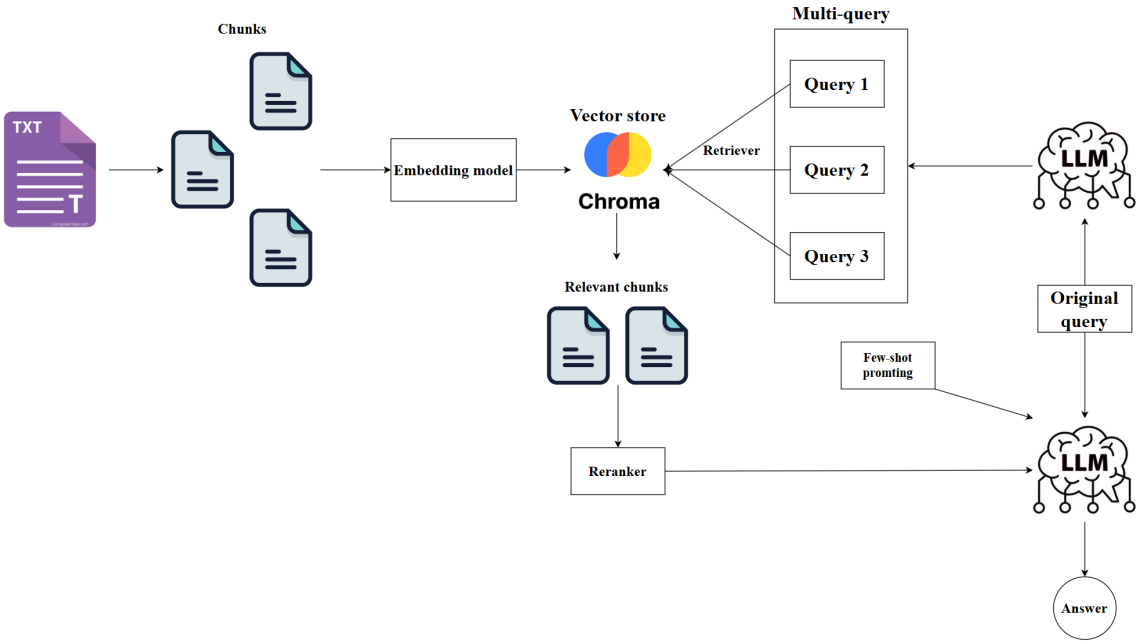
Tuy nhiên, với những LLM có quy mô tham số hạn chế, việc thêm quá nhiều ví dụ vào prompt đôi khi gây quá tải thông tin, khiến ngữ cảnh trọng yếu bị che khuất và làm giảm tính chính xác. Vì vậy, chúng em chỉ sử dụng một số lượng ví dụ vừa đủ để cân bằng giữa việc hướng dẫn định dạng đầu ra và việc giữ nguyên nội dung ngữ cảnh quan trọng.

3.4 Mô hình LLM cho RAG và thư viện triển khai

Để phục vụ cho việc sinh câu trả lời trong hệ thống, chúng tôi đã tiến hành thử nghiệm hai mô hình ngôn ngữ lớn (LLM) khác nhau: Llama 3.2 (với 3 tỷ tham số) và Gemma 2 (với 2 tỷ tham số). Cả hai mô hình này đều được lựa chọn dựa trên khả năng hỗ trợ đa ngôn ngữ và tiềm năng sinh ngôn ngữ tự nhiên chất lượng cao. Để sinh ra các câu trả lời ngắn gọn và trực tiếp, ngoài việc điều chỉnh prompt và few shot, các tham số của mô hình như Temperature được đặt là 0.1, và số lượng từ sinh ra tối đa là 50 để đảm bảo câu trả lời chính xác và không lan man.

Về phương thức triển khai, mô hình Llama 3.2 được chạy cục bộ sử dụng nền tảng Ollama, trong khi Gemma 2 được tích hợp và gọi trực tiếp thông qua thư viện Hugging Face Transformers. Một điểm thuận lợi đáng chú ý là cả hai mô hình này đều có thể vận hành hiệu quả trên cấu hình máy tính cá nhân thông thường. Chúng em cũng đã xem xét các nền tảng đám mây (cloud) như Together AI, tuy nhiên, các giới hạn về số lượng token và request không đáp ứng được quy mô thử nghiệm của dự án.

Trong quá trình phát triển hệ thống, đặc biệt là cho thành phần RAG (Retrieval-Augmented Generation), chúng tôi đã sử dụng thư viện Langchain. Langchain là một framework mạnh mẽ, cung cấp sẵn các công cụ và hàm giúp đơn giản hóa đáng kể việc xây dựng các ứng dụng phức tạp liên quan đến LLM. Việc sử dụng Langchain đã góp phần đẩy nhanh tốc độ phát triển và nâng cao khả năng quản lý của hệ thống.



Hình 3.4.1: Cấu trúc hệ thống RAG chứa đầy đủ các kỹ thuật sử dụng

3.5 Hệ thống dựa trên LLM-only

Và cuối cùng, để so sánh sự hiệu quả của việc xây dựng hệ thống RAG so với các phương pháp truyền thông, hệ thống hỏi đáp dựa trên LLM-only đã được thử nghiệm , tức là không sử dụng ngữ cảnh được truy xuất từ hệ thống RAG mà chỉ sử dụng những kiến thức đã được học, cùng với đó là việc tinh chỉnh Prompt dựa kết hợp với Few-shot learning để mô hình LLM có thể biết được sẽ phải dùng những kiến thức tiềm ẩn đã được học nào để trả lời câu hỏi. Mô hình được sử dụng vẫn là LLama3.2 ở trên để có một so sánh trung thực nhất giữa các phương pháp.

Chương 4

Kết quả

4.1 Phương pháp đánh giá

Để đánh giá độ hiệu quả của các phương pháp và kỹ thuật đã sử dụng, chúng em sử dụng 4 chỉ số, trong đó có 3 chỉ số phổ biến là Recall, Precision và F1-Score.

- Precision: Là tỷ lệ giữa số lượng từ giống nhau giữa câu trả lời dự đoán và câu trả lời thực tế, với số lượng từ của câu trả lời dự đoán.
  - Công thức tính Precision như sau:

$$Precision = \frac{\text{Số lượng từ trùng nhau}}{\text{Số lượng từ của câu trả lời dự đoán}} \tag{4.1}$$

- Recall: Là tỷ lệ giữa số lượng từ giống nhau giữa câu trả lời dự đoán và câu trả lời thực tế, với số lượng từ của câu trả lời thực tế.
  - Công thức tính Recall như sau:

$$Recall = \frac{\text{Số lượng từ trùng nhau}}{\text{Số lượng từ của câu trả lời thực tế}} \tag{4.2}$$

- F1-score: Là giá trị trung bình điều hòa, giúp cân bằng giữa Precision và Recall
  - Công thức tính F1-score như sau:

$$F1 - score = \frac{2 * \text{ Precision * Recall}}{\text{Precision + Recall}} \tag{4.3}$$

- Exact-match: Có giá trị là 1 nếu hai câu giống nhau hoàn toàn, và 0 nếu ngược lại.

4.2 Kết quả

Sau khi thực hiện dự đoán trên bộ dữ liệu test bao gồm 188 câu hỏi và câu trả lời, kết quả thu được như sau:

Phương pháp/(Đều sử dụng few-shot)	Mô hình sử dụng	Recall	Precision	F1-Score	Exact-match
LLM-only	LLama 3.2	0.43	0.21	0.25	3 / 188
Retriever only	Gemma 2	0.34	0.42	0.33	13 / 188
	LLama 3.2	0.43	0.41	0.37	23 / 188
Retriever + Reranker	Gemma 2	0.37	0.42	0.34	15 / 188
	LLama 3.2	0.45	0.43	0.39	27 / 188
Retriever + Reranker + Multi-question	Gemma 2	0.38	0.43	0.35	16 / 188
	LLama 3.2	0.48	0.44	0.40	21 / 188

Hình 4.2.1: Kết quả của các phương pháp

Còn trong hình 4.2.1 là câu trả lời của các phương pháp đối với 3 câu hỏi tiêu biểu.

Chúng em có thực hiện đánh giá ý nghĩa thống kê giữa các phương pháp thông qua tính chỉ số P-value giữa 2 phương pháp. Kết quả so sánh dựa trên Phương pháp đánh giá được thiết lập với tỉ lệ lấy mẫu là 1 , replace = True để có thể lấy mẫu lại phần tử đã có. Số lượng lần lấy mẫu là 10000 đảm bảo đánh giá chính xác. Ở đây, chúng em sử dụng mức ý nghĩa (significance value = 0.05). Kết quả thu được như trong hình 4.2.2.

4.3 Phân tích kết quả

Đựa trên bảng kết quả hình 4.2.1 thu được, ta nhận thấy rằng các phương pháp và kỹ thuật với RAG sử dụng trong hệ thống đạt kết quả tốt hơn so với việc chỉ sử dụng mô hình LLM-only, Kỹ thuật kết hợp Rerank và Multi-question đạt kết quả tốt nhất ở cả 3 chỉ số Recall, Precision và F1-Score, nhưng còn kém hơn ở chỉ số Exact-match, điều này có thể là do việc sử dụng Multi-question tạo ra ngữ cảnh đa dạng hơn khiến câu trả lời mặc dù đúng hơn về mặt ngữ nghĩa nhưng lại có cách diễn đạt dài hơn hoặc khác đi một ít, khiến cho nó không trùng với câu trả lời gốc. So sánh giữa 2 mô hình dùng để sinh câu hỏi là LLama3.2 và Gemma2 ta thấy rằng LLama3.2 đạt kết quả tốt hơn nhiều so với Gemma2 ở các chỉ số, điều này là dễ hiệu vì có thể do sự chênh lệch về số lượng tham số mô hình.

Câu hỏi	Nhãn	LLM-only	Retriever	Retriever + Re-ranker	Retriever + Re-ranker + Multi-query
Đại học Quốc gia Hà Nội được thành lập năm nào?	1993	Đại học Quốc gia Hà Nội được thành lập năm 1940.	1993	1993	1993
Trụ sở chính của Đại học Quốc gia Hà Nội nằm ở đâu?	Hòa Lạc, huyện Thạch Thất, thành phố Hà Nội.	Trụ sở chính của Đại học Quốc gia Hà Nội nằm tại phố Dương Văn Minh, phường Cầu Diễn, quận Nam Từ Liêm.	Khu vực Hòa Lạc, thành phố Hà Nội.	Khu vực Hòa Lạc, huyện Thạch Thất, thành phố Hà Nội.	Khu vực Hòa Lạc (xã Thạch Hòa, huyện Thạch Thất, thành phố Hà Nội)
Tên tiếng Anh đầy đủ của Đại học Quốc gia Hà Nội là gì?	Vietnam National University, Hanoi	Đại học Quốc gia Hà Nội	Vietnam National University, Hanoi (VNU)	Vietnam National University, Hanoi (VNU)	Vietnam National University, Hanoi

Hình 4.2.2: So sánh câu trả lời của các phương pháp

2 hệ thống để so sánh	Số lượng lần thắng của hệ thống 1	Số lượng lần thắng của hệ thống 2	P-value	Có ý nghĩa thống kê không
Retrieve (1) và Re-ranker (2)	1945	8055	0.195	Không có ý nghĩa thống kê
Retriever (1) và Retriever + Re-ranker + Multi-query (2)	406	9594	0.0406	Có ý nghĩa thống kê
Re-rank (1) và Retriever + Re-rank + Multi-query (2)	2615	7385	0.2615	Không có ý nghĩa thống kê

Hình 4.2.3: Kết quả xác định ý nghĩa thống kê

Qua hình 4.4.2 ta thấy được mô hình LLM độc lập thường đưa ra câu trả lời sai lệch hoặc không đầy đủ, trong khi việc áp dụng RAG đã cải thiện đáng kể độ chính xác cơ bản. Các kỹ thuật bổ sung như Reranker và đặc biệt là Multi-query tiếp tục nâng cao độ chính xác và tính chi tiết của thông tin được truy xuất, dẫn đến câu trả lời đầy đủ và chính xác hơn hẳn, chứng tỏ các phương pháp RAG nâng cao là rất cần thiết để đảm bảo độ tin cậy và chất lượng cho các hệ thống hỏi đáp dựa trên tri thức.

Kết quả phân tích ý nghĩa thống kê ở hình 4.4.2 cho thấy rằng việc bổ sung cả Rerank và Multi-query vào hệ thống Retriever ban đầu (tạo thành ”Retriever + Rerank + Multi-query”) mang lại cải thiện hiệu suất có ý nghĩa thống kê rõ rệt so với chỉ dùng Retriever. Trong khi đó, sự khác biệt giữa Retriever và Reranker đơn lẻ, cũng như giữa Rerank và cấu hình kết hợp đầy đủ, không đủ mạnh để được coi là có ý nghĩa thống kê. Điều này nhấn mạnh tầm quan trọng của sự kết hợp các thành phần trong việc tối ưu hóa hiệu suất hệ thống tổng thể.

4.4 Hạn chế và hướng phát triển trong tương lai

- Đa dạng hóa nguồn dữ liệu: Bổ sung thêm các nguồn dữ liệu phi cấu trúc như video bài giảng, hội thảo chuyên đề của ĐHQGHN và UET để hệ thống có thể trả lời cả những câu hỏi liên quan tới nội dung đa phương tiện. Tích hợp dữ liệu từ mạng xã hội (fanpage, group sinh viên) để cập nhật nhanh các thông tin hoạt động, sự kiện, học bổng mới.
- Cải tiến module truy vấn và sinh câu trả lời: Thử nghiệm và so sánh thêm các mô hình embedding thế hệ mới (ví dụ: OpenAI-Embeddings, Cohere) để nâng cao hiệu quả retriever và cải thiện độ bao phủ thông tin. Áp dụng kỹ thuật adaptive chunking: điều chỉnh độ dài và overlap của đoạn chia tài liệu dựa trên đặc thù nội dung (tài liệu kỹ thuật, văn bản chính sách, mô tả hoạt động).
- Tăng cường khả năng hiểu ngữ cảnh: Triển khai mô hình RAG hai chiều (bidirectional RAG) cho phép hệ thống vừa “đọc” ngược và “xuôi” giữa câu hỏi và docs, cải thiện khả năng hiểu các câu hỏi phức tạp.

Chương 5

Kết luận

Qua quá trình xây dựng và đánh giá, hệ thống RAG cho nhiệm vụ hỏi–đáp về Đại học Quốc gia Hà Nội nói chung và Trường Đại học Công nghệ nói riêng đã cho thấy hiệu quả vượt trội so với phương pháp LLM-only truyền thống. Việc áp dụng các kỹ thuật như Reranking, Multi-query và Few-shot learning đã cải thiện rõ rệt các chỉ số Recall, Precision và F1-Score, đồng thời giữ được độ chính xác thông tin cao. Mô hình Llama 3.2 thường đạt kết quả tốt hơn so với Gemma 2, cho thấy mô hình có quy mô tham số lớn hơn hỗ trợ tốt hơn cho việc trả lời câu hỏi chuyên sâu.

Việc kết hợp nhiều thành phần trong pipeline (retriever, reranker, LLM) cho phép hệ thống xử lý linh hoạt các yêu cầu phức tạp và cập nhật kiến thức mới nhanh chóng. Kết quả thử nghiệm bước đầu khẳng định tính khả thi và hiệu quả của RAG trong môi trường tri thức phong phú như ĐHQGHN.

Đóng góp của các thành viên

Link code: `https://github.com/DUNGTK2004/VNU-RAG-QA.git`

1. Dương Phương Hiếu
    - Tìm hiểu và thử nghiệm các mô hình LLM như Gemma3 để sinh ra câu trả lời.
    - Tìm hiểu và xây dựng mô hình RAG với kỹ thuật Rerank.
    - Đánh giá các chỉ số như F1-score, Recall, Precision.
    - Viết báo cáo phần đánh giá kết quả.
2. Phạm Long Nhật
    - Chọn nguồn và thu thập dữ liệu từ các trang web chính thống của VNU, UET.
    - Làm sạch và xử lý dữ liệu
    - Chia dữ liệu theo từng chunk nhỏ hơn theo từng page
    - Viết báo cáo phần thu thập, xử lý dữ liệu
3. Trần Kim Dũng
    - Xây dựng phần RAG gồm các bước như chia dữ liệu thành các chunk, embedding, lưu trữ vào database.
    - Thử nghiệm giữa các mô hình embeding, Reranking để chọn mô hình tốt.
    - Thử nghiệm thêm mô hình LLama3.2 để so sánh.
    - Xây dựng hệ thống RAG với Multi-query.
    - Viết báo cáo phần phương pháp.
4. Hồ Hà Ngọc Nhất
    - Thực hiện thu thập dữ liệu và xử lý dữ liệu.
    - Gán nhãn dữ liệu training, testing theo dạng câu hỏi, câu trả lời.
    - Đánh giá chất lượng dữ liệu.
    - Viết báo cáo phần gán nhãn và đánh giá dữ liệu.