ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

Khóa Luận Tốt Nghiệp Đại Học

HỆ THỐNG TƯ VẤN MÔN HỌC CHO SINH VIÊN

KHOA TOÁN-TIN HỌC
NGÀNH KHOA HỌC DỮ LIỆU
KHÓA 2021

Sinh viên thực hiện	MSSV
Trịnh Minh Anh	21280005
Lê Hồ Hoàng Anh	21280085

Giảng viên hướng dẫn: TS. Tô Đức Khánh

Tp. Hồ Chí Minh, tháng 7, 2025

Lời Cảm Ơn

Chúng tôi xin chân thành gửi lời cảm ơn sâu sắc đến TS. Tô Đức Khánh. Thầy là người giảng viên hướng dẫn và hỗ trợ tận tụy xuyên suốt giai đoạn làm luận văn này. Bên cạnh đó, chúng tôi cũng xin gửi lời cảm ơn đến sự hỗ trợ tiếp sức từ phía gia đình và bạn bè.

Trong quá trình thực hiện luận văn này, chúng tôi sẽ không thể tránh được những sai sót. Chúng tôi kính mong nhận được sự góp ý từ quý thầy cô để chúng tôi có thể cải thiện tốt hơn luận văn nghiên cứu của mình.

Xin chân thành cảm ơn.

Tóm tắt nội dung

Luận văn tập trung vào việc thiết kế và xây dựng hệ thống hỏi-đáp thông tin học phần dành cho sinh viên đại học. Hệ thống được xây đựng dựa trên mô hình Retrieval-Augmented Generation (RAG), giúp hỗ trợ tra cứu học phần nhanh chóng, chính xác và hiệu quả. Dữ liệu đầu vào được trích xuất từ đề cương của ba chương trình đào tạo thuộc Khoa Toán – Tin học, bao gồm: Khoa học Dữ liệu, Toán Ứng dụng và Toán Tin. Hệ thống gồm ba thành phần: tiền xử lý dữ liệu và xây dựng tri thức học phần, truy xuất tài liệu liên quan theo truy vấn, và sinh câu trả lời tự động bằng mô hình ngôn ngữ lớn. Ngoài mô hình RAG cơ bản, luận văn mở rộng với kiến trúc RAG tích hợp đồ thị, sử dụng mô hình HippoRAG-2 để khai thác liên kết giữa các học phần như: điều kiện tiên quyết hoặc kiến thức liên quan. Cuối cùng, hai mô hình được đánh giá so sánh dựa trên tập câu hỏi kiểm thử, nhằm làm rõ khả năng trả lời truy vấn phức tạp, hướng đến ứng dụng thực tiễn trong môi trường giáo dục đại học, cụ thể tại Trường Đại học Khoa học tự nhiên, ĐHQG-HCM.

Abstract

This thesis focuses on the design and development of a course information question-answering system for university students. The system is built upon the Retrieval–Augmented Generation (RAG) framework to support fast, accurate, and efficient course lookup. Input data is extracted from the module handbook of three academic programs offered by the Faculty of Mathematics and Computer Science: Data Science, Applied Mathematics, and Mathematics and Computer Science. The system comprises three main components: preprocessing and construction of course knowledge, retrieval of relevant documents based on user queries, and automatic answer generation using a large language model. In addition to the baseline RAG model, the thesis extends the architecture with a graph-integrated RAG variant, HippoRAG–2, to leverage inter-course relationships such as prerequisites and related knowledge. Finally, the two models are comparatively evaluated on a curated test set of questions to assess their ability to handle complex queries, to demonstrate the system's practical applicability in higher education, specifically at the University of Science, Vietnam National University, Ho Chi Minh City.

Mục lục

Danh sách hình vẽ

Danh sách bảng

Danh sách ký hiệu

AI	Artificial Intelligence Trí tuệ nhân tạo
AM	Applied Mathematics Toán Ứng dụng
API	Application Programming Interface Giao diện lập trình ứng dụng
BERTs	Bidirectional Encoder Representations from Transformers Các mô hình biểu diễn mã hóa hai chiều từ Transformer
BLEU	Bilingual Evaluation Understudy Phương pháp đánh giá song ngữ thay thế
DS	Data Science Khoa học Dữ liệu
DPR	Dense-Retrieval Passages Truy xuất dày đặc đoạn văn
GPTs	Generative Pre-trained Transformers Các mô hình Transformer Tiền huấn luyện để Sinh văn bản
GraphRAG	Graph-structured Retrieval-Augmented Generation Mô hình RAG tích hợp cấu trúc đồ thị
H-GraphRAG	Hybrid Graph-structured Retrieval-Augmented Generation Mô hình GraphRAG kết hợp
IB-GraphRAG	Index-based Graph-structured Retrieval-Augmented Generation Mô hình GraphRAG dựa trên chỉ mục
GRIT	Generative Representational Instruction Tuning Điều chỉnh theo hướng dẫn mang tính tạo sinh và biểu diễn
IR	Information Retrieval Truy hồi thông tin
KB-GraphRAG	Knowledge-based Graph-structured Retrieval-Augmented Generation Mô hình GraphRAG dựa trên tri thức
LLMs	Large Language Models Các mô hình ngôn ngữ lớn
LCS	Longest Common Subsequence Dãy con chung dài nhất
MCS	Mathematics for Computer Science Toán Tin
METEOR	Metric for Evaluation of Translation with Explicit ORdering

	Chỉ số đánh giá dịch máy có xét đến thứ tự từ
MD5	Message Digest Algorithm 5 Hàm băm mật mã 5
MTEB	Massive Text Embedding Benchmark Bộ đánh giá chuẩn cho các mô hình nhúng văn bản quy mô lớn
NER	Named Entity Recognition Nhận dạng thực thể
NLP	Natural Language Processing Xử lý ngôn ngữ tự nhiên
OpenIE	Open Information Extraction Khai thác thông tin mở
PPR	Personalized PageRank Thuật toán PageRank cá nhân hóa
RAG	Retrieval-Augmented Generation Tạo sinh có tăng cường truy xuất
ROUGE	Recall-Oriented Understudy for Gisting Evaluation Phương pháp đánh giá tóm tắt dựa trên độ bao phủ
TF-IDF	Term Frequency – Inverse Document Frequency Tần suất thuật ngữ – Tần suất nghịch đảo của tài liệu

Chương 1

Mở đầu

Ở chương này, độc giả sẽ nắm được tổng quan về bối cảnh và động lực nghiên cứu của chúng tôi, bao gồm lý do lựa chọn đề tài, các vấn đề còn tồn tại trong việc tra cứu thông tin học phần hiện nay, cũng như mục tiêu, phạm vi và phương pháp tiếp cận của luận văn. Ngoài ra, chúng tôi cũng giới thiệu cấu trúc tổng thể của luận văn nhằm định hướng cho người đọc trong các chương tiếp theo.

1.1 Đặt vấn đề

Trong môi trường học tập hiện đại nói chung và của trường Đại học Khoa học tự nhiên, ĐHQG-HCM nói riêng, chúng tôi – những sinh viên năm cuối ngành Khoa học dữ liệu thuộc Khoa Toán - Tin học, ngày càng nhận ra nhu cầu của các bạn/em sinh viên về việc tra cứu nhanh chóng và chính xác các thông tin quan trọng, liên quan đến học phần như học kỳ mở lớp, loại học phần, giảng viên phụ trách, nội dung giảng dạy, mục tiêu đầu ra và các điều kiện tiên quyết. Bởi lẽ, những thông tin này đóng vai trò thiết yếu trong việc lập kế hoạch học tập, lựa chọn môn học phù hợp với định hướng cá nhân cũng như chuẩn bị cho các kỳ đăng ký học phần. Tuy nhiên, không phải lúc nào sinh viên cũng có đủ thời gian và điều kiện để tự mình tìm hiểu tất cả các thông tin cần thiết qua các nguồn rải rác như trang web khoa, file PDF đề cương, hay trao đổi trực tiếp với cố vấn học tập.

Thực tế cho thấy, việc tra cứu thủ công các đề cương môn học hoặc gửi câu hỏi lên nền tảng hỗ trợ trực tuyến như nền tảng Zalo tư vấn của Khoa Toán - Tin còn nhiều bất cập. Những hạn chế bao gồm: độ trễ trong phản hồi do phụ thuộc vào cán bộ trực, thông tin phản hồi mang tính thủ công và không đồng nhất, cũng như khả năng tiếp nhận cùng lúc nhiều yêu cầu còn hạn chế. Điều này dẫn đến sự trì hoãn không mong muốn trong việc ra quyết định học tập của sinh viên, đặc biệt trong các giai đoạn cao điểm như thời gian đăng ký môn học.

Hiện nay, các hệ thống hỏi đáp tự động ứng dụng trí tuệ nhân tạo đang nổi lên như một giải pháp đầy tiềm năng trong lĩnh vực giáo dục đại học nói riêng và sự phát triển của các lĩnh vực xã hội nói chung. Khả năng xử lý ngôn ngữ tự nhiên kết hợp với truy xuất thông tin giúp các hệ thống này hỗ trợ và tương tác với sinh viên một cách tự nhiên, nhanh chóng và chính xác hơn. Mở ra hướng tiếp cận mới để xây dựng các hệ thống tư

vấn học phần thông minh và hiệu quả hơn trong bối cảnh chuyển đổi số giáo dục.

1.2 Mục tiêu nghiên cứu

Trước những hạn chế còn tồn tại trong quá trình sinh viên tra cứu thông tin học phần theo phương thức thủ công hoặc thông qua các kênh hỗ trợ trực tuyến truyền thống, chúng tôi hướng đến việc xây dựng một hệ thống hỏi đáp tự động chuyên biệt cho việc tư vấn học phần trong môi trường giáo dục đại học – nơi thông tin học phần không chỉ đa dạng về nội dung mà còn phong phú về hình thức biểu đạt. Cụ thể, chúng tôi đề xuất một hệ thống hỏi đáp sử dụng mô hình RAG nhằm tận dụng khả năng truy xuất tri thức hiệu quả từ kho dữ liệu học phần, đồng thời tạo sinh câu trả lời tự nhiên, dễ hiểu và phù hợp với ngữ cảnh câu hỏi của sinh viên.

Như đã trình bày trong phần tóm tắt nội dung của luận văn, chúng tôi sẽ tiến hành một quy trình đánh giá toàn diện đối với hai hệ thống tư vấn: mô hình RAG cơ bản và mô hình RAG tích hợp đồ thị tri thức. Quá trình đánh giá được thực hiện thông qua các chỉ số định lượng phổ biến, kết hợp với đánh giá định tính dựa trên các tình huống thực tế được thiết kế sẵn. Các tiêu chí này nhằm làm rõ phạm vi mà hệ thống có thể xử lý hiệu quả; đồng thời phản ánh tính ứng dụng, mức độ chính xác, và khả năng mở rộng của hai mô hình. Từ đó chúng tôi sẽ đề xuất mô hình phù hợp với dữ liệu đầu vào và bối cảnh chuyển đổi số giáo dục đại học hiện nay nói chung và Trường Đại học Khoa học tự nhiên, ĐHQG-HCM, khoa Toán - Tin học nói riêng.

1.3 Đối tượng và phạm vi nghiên cứu

Nghiên cứu này tập trung vào việc xây dựng và thử nghiệm hệ thống hỏi đáp tư vấn học phần, với đối tượng nghiên cứu là tập hợp toàn bộ các học phần thuộc ba chương trình đào tạo của Khoa Toán – Tin, Trường Đại học Khoa học tự nhiên, ĐHQG-HCM, khóa tuyển 2021: Khoa học dữ liệu, Toán Tin học và Toán Ứng dụng. Các học phần này được thu thập từ dữ liệu chính thức do khoa công bố, bao gồm các đề cương học phần, thông tin giảng viên, tài liệu học tập và các yếu tố liên quan như số tín chỉ, học phần tiên quyết, mục tiêu đầu ra và phương pháp giảng dạy cũng như đánh giá.

Phạm vi nghiên cứu của luận văn này tập trung vào giai đoạn truy xuất và phản hồi về thông tin học phần theo hình thức hỏi đáp tự động, dựa trên dữ liệu đầu vào có cấu trúc. Hệ thống được thiết kế để tiếp nhận các câu hỏi tự nhiên từ sinh viên và trả về câu trả lời phù hợp, chính xác, dễ hiểu. Hệ thống không mở rộng sang chức năng gợi ý hay khuyến nghị học phần tiếp theo dựa trên lịch sử học tập cá nhân hay mục tiêu nghề nghiệp, vốn là phạm trù riêng của các hệ thống tư vấn học tập mang tính dự đoán và cá nhân hóa.

Sự khác biệt giữa hai hướng tiếp cận này cần được làm rõ: trong khi các hệ thống gợi ý học phần tập trung vào việc phân tích hành vi học tập của sinh viên, để dự đoán lộ trình phù hợp trong tương lai. Hệ thống mà chúng tôi phát triển mang tính chất hỗ trợ truy xuất và phản hồi thông tin theo ngữ cảnh của câu hỏi từ sinh viên. Việc giới hạn phạm vi này là có chủ đích, chúng tôi muốn đảm bảo độ chính xác, bao phủ và tính ứng

dụng thực tế trong môi trường giáo dục đại học hiện nay, nơi mà nhu cầu nắm bắt thông tin nhanh chóng và chính xác luôn là yếu tố then chốt.

1.4 Đóng góp chính của luận văn

Trên nền tảng nhu cầu thực tiễn đã nêu và phạm vi nghiên cứu đã xác định, luận văn này hướng đến việc đề xuất sử dụng và phát triển một hệ thống hỏi đáp học phần tự động, ứng dụng mô hình RAG kết hợp với đồ thị tri thức trong bối cảnh giáo dục đại học. Đây là hướng tiếp cận còn tương đối mới trong lĩnh vực giáo dục, nơi các tác vụ hỏi đáp thường bị giới hạn bởi dữ liệu đầu vào phi cấu trúc, thông tin được truy xuất bị rời rạc và câu trả lời không được tự nhiên, thiếu chính xác.

Để có được bộ dữ liệu đầu vào sạch sẽ và có cấu trúc cho hệ thống tư vấn hỏi đáp học phần, chúng tôi đề xuất một chuỗi quy trình xử lý dữ liệu học phần đầu vào, được liên kết với nhau một cách hoàn chỉnh và tự động. Không những thế, chuỗi quy trình xử lý này còn mang tính ứng dụng tổng quát – tức có thể sử dụng cho nhiều bộ dữ liệu học phần không chỉ dừng lại ở khoa Toán - Tin học.

Một đóng góp quan trọng tiếp theo của luận văn là việc kết hợp đánh giá, phân tích và so sánh giữa các mô hình khác nhau trên cùng một bộ dữ liệu đầu vào đã được xử lý và làm sạch, bao gồm: (i) mô hình RAG cơ bản sử dụng truy xuất từ kho văn bản ngữ nghĩa, (ii) mô hình RAG tích hợp truy xuất dựa trên đồ thị tri thức. Hai mô hình này sẽ được thử nghiệm các mô hình nhúng như NV-embed-v2, text-embedding-3-small, GritLM-7B, Contriever và được so sánh với nhau. Việc so sánh sẽ dựa trên các chỉ số định lượng, nhằm đánh giá chính xác hiệu năng của thành phần truy xuất và tạo sinh ngôn ngữ cho câu trả lời của cả hai mô hình. Ngoài ra, chúng tôi còn kết hợp thêm đánh giá định tính thông qua các trường hợp nghiên cứu được nêu ở phần sau, nhằm đặt ra giới hạn để đảm bảo khả năng hoạt động tốt của cả hệ thống. Qua đó, luận văn sẽ cung cấp cái nhìn toàn diện về ưu và nhược điểm của từng hướng tiếp cận. Từ đó, chúng tôi sẽ đề xuất lựa chọn sử dụng mô hình phù hợp cho với mục tiêu nghiên cứu đã nêu.

1.5 Cấu trúc tổng thể của luận văn

Nhằm định hướng người đọc dễ nắm bắt được từ tổng thể đến chi tiết nội dung trong luận văn này. Chúng tôi sẽ tóm tắt nội dung chính ở từng chương. Bài luận này bao gồm tất cả là 6 chương, bao gồm:

Chương 1: Chúng tôi sẽ trình bày về mục tiêu nghiên cứu, cũng như các đóng góp chính của luận văn. Song, xác định đối tượng nghiên cứu cũng như các tình huống nghiên cứu được đặt ra cho hệ thống tư vấn học phần.

Chương 2: Độc giả sẽ nắm được một cách tổng quan về một hệ thống RAG cơ bản cùng với các thành phần chính. Ngoài ra, các nghiên cứu liên quan trong những năm gần đây cũng được đề cập.

Chương 3: Nhằm củng cố về mảng kiến thức học thuật, nội dung ở chương 3 sẽ tập trung trình bày chi tiết về cơ sở lý thuyết của mô hình RAG cơ bản và RAG tích hợp đồ

thị tri thức, cũng như các thuật toán và phương pháp đánh giá liên quan được sử dụng trong từng thành phần chính của cả hai mô hình.

Chương 4: Ở phần này, chúng tôi sẽ trình bày về chuỗi liên kết các quy trình xử lý dữ liệu học phần đầu vào. Song, chúng tôi sẽ trình bày về hai hệ thống RAG được sử dụng.

Chương 5: Chương này cung cấp thông tin tổng quan về qúa trình xây dựng bộ dữ liệu kiểm thử cho cả hai mô hình RAG được sử dụng. Đồng thời trình bày về các kết quả thực nghiệm: đánh giá khả năng truy xuất và tạo sinh của cả hai mô hình RAG là RAG cơ bản và RAG tích hợp đồ thị, khi sử dụng hai mô hình ngôn ngữ lớn khác nhau là GPT 4-o mini và Llama 8B.

Chương 6: Đây là phần kết thúc của luận văn, nơi chúng tôi sẽ đưa ra nhận định về việc lựa chọn hệ thống phù hợp, tóm tắt lại những đóng góp chính cũng như thảo luận về một số hạn chế gặp phải của hệ thống và trình bày hướng phát triển cho luận văn này.

1.6 Các tình huống nghiên cứu của hệ thống tư vấn học phần

Trong quá trình học tập và lập kế hoạch học phần, sinh viên thường đối mặt với nhiều câu hỏi mang tính học thuật lẫn hành chính như: *"Môn này có tiên quyết không?", "Tôi nên học những môn nào trước?"*, hay *"Các môn trong lĩnh vực X sẽ giúp tôi đạt được năng lực Y như thế nào?"*. Đây không những là thắc mắc cá nhân rời rạc, mà còn là biểu hiện của các tình huống câu hỏi được đặt ra cho hệ thống tư vấn hỏi đáp học phần cho sinh viên. Tuy nhiên, số lượng các tình huống câu hỏi sẽ rất lớn và chúng tôi không hướng tới một hệ thống "biết mọi thứ". Thay vào đó, chúng tôi lựa chọn một hướng tiếp cận thực tiễn hơn là xác định rõ các tình huống nghiên cứu mà hệ thống tư vấn học phần cần phải xử lý. Mỗi tình huống phản ánh một nhu cầu hỏi đáp phổ biến trong môi trường giáo dục, chẳng hạn như hỏi đáp về nội dung học phần, các điều kiện tiên quyết của chúng, hoặc là tổng hợp mục tiêu học tập đầu ra theo nhóm học phần.

Tóm lại, mục tiêu của hệ thống tư vấn hỏi đáp học phần không phải là trả lời mọi câu hỏi có thể xảy ra, mà là hỗ trợ truy xuất và trả lời thông tin chính xác, ổn định trong phạm vi những tình huống câu hỏi đã được định nghĩa rõ ràng. Bảng 1.1 bao gồm các tình huống nghiên cứu được trình bày cụ thể cùng với dạng câu hỏi đầu vào và kết quả đầu ra – nhằm làm rõ ranh giới hoạt động và khả năng của hệ thống tư vấn của chúng tôi. Ngoài ra, khi bàn về vấn đề ngôn ngữ, chúng tôi ưu tiên lựa chọn ngôn ngữ Tiếng Anh cho hệ thống của mình.

Bảng 1.1: Danh sách các tình huống được đặt ra cho hệ thống tư vấn hỏi đáp học phần.

	Tình huống	**Câu hỏi đầu vào**	**Câu trả lời đầu ra**
1	Truy xuất thông tin chi tiết của một học phần	Hỏi về 1 hoặc nhiều trường thông tin của một học phần (mã, tên học phần, học kỳ mở, tên giảng viên, loại học phần, danh sách các học phần tiên quyết, mục tiêu đầu ra, nội dung giảng dạy). **Ví dụ:** *"Which semester does the Ho Chi Minhs Ideology course available?"*	Đưa ra câu trả lời theo thông tin được hỏi về một học phần.
2	Truy xuất mục tiêu đầu ra theo kiến thức hoặc kỹ năng trong một học phần.	Tìm kiếm các mục tiêu đầu ra theo kiến thức hoặc kỹ năng của một học phần. **Ví dụ:** *"What software tool are students expected to use in the Linear Algebra course?"*	Đưa ra các mục tiêu đầu ra liên quan đến kiến thức hoặc kỹ năng của một học phần.
3	Truy xuất nội dung theo chủ đề trong một học phần.	Tìm kiếm nội dung theo chủ đề của một học phần. **Ví dụ:** *"How many chapters are there in General Environment course content?"*	Mô tả các phần nội dung liên quan đến chủ đề được hỏi của học phần đó.
4	Truy xuất và tổng hợp mục tiêu đầu ra của một nhóm các học phần (tối đa 5)	Mô tả các mục tiêu đầu ra liên quan đến kiến thức hoặc kỹ năng của một nhóm các học phần. **Ví dụ:** *"What are the key learning outcomes in terms of knowledge for Calculus 1A, Calculus 2A, and Analysis 3A?"*	Đưa ra các mục tiêu đầu ra liên quan đến kiến thức hoặc kỹ năng cho từng học phần.

Tiếp tục vào trang sau.

Bảng 1.1 – Tiếp tục từ trang trước.

	Tình huống	**Câu hỏi đầu vào**	**Câu trả lời đầu ra**
5	Truy xuất và tổng hợp mục tiêu đầu ra của một nhóm các học phần (tối đa 5) theo 1 trong 3 điều kiện: loại học phần, học kỳ mở, điều kiện tiên quyết.	Mô tả các mục tiêu đầu ra liên quan đến kiến thức và kỹ năng cho một nhóm các học phần theo điều kiện về loại học phần, học kỳ mở hoặc điều kiện tiên quyết. **Ví dụ:** *"What knowledge and skills are expected from Compulsory courses like Linear Algebra, Abstract Algebra, and Fundamentals of Computer Programming?"*	Đưa ra các mục tiêu đầu ra cho từng học phần đã được lọc theo điều kiện được hỏi.
6	Truy xuất nội dung và các học phần tiên quyết của nhóm học phần (tối đa 5) có nội dung liên quan.	Mô tả nội dung trọng tâm của một nhóm học phần theo chủ đề cụ thể và các học phần tiên quyết của nhóm học phần đó. **Ví dụ:** *"What topics are covered in Abstract Algebra and Abstract Algebra Practice, and what are their prerequisites?"*	Đối với từng học phần trong nhóm liên quan, đưa ra tên học phần, danh sách các học phần tiên quyết, mô tả nội dung chính.
7	Truy xuất quan hệ giữa các học phần tiên quyết trong một nhóm các học phần (tối đa 5).	Tìm kiếm các học phần tiên quyết trong nhóm các học phần cụ thể. Trình bày mối quan hệ tiên quyết giữa các học phần tiên quyết ấy. **Ví dụ:** *"What are the prerequisites for Abstract Algebra Practice and Algebra A2 and what is the relationship between these prerequisite courses?"*	Đối với từng học phần trong nhóm liên quan, đưa ra tên học phần, danh sách các học phần tiên quyết, mối liên hệ tiên quyết của danh sách này.
8	Truy xuất loại học phần và học kỳ mở của một nhóm các học phần (tối đa 5).	Tìm kiếm loại học phần và học kỳ mở của một nhóm các học phần. **Ví dụ:** *"In which semester are Analysis 1A and Calculus 1A offered, and what is their course type?"*	Đối với từng học phần trong nhóm liên quan, đưa ra loại học phần và học kỳ mở.

Tiếp tục vào trang sau.

Bảng 1.1 – Tiếp tục từ trang trước.

	Tình huống	**Câu hỏi đầu vào**	**Câu trả lời đầu ra**
9	Truy xuất danh sách các học phần theo các điều kiện: loại học phần, học kỳ mở, điều kiện tiên quyết.	Tìm kiếm danh sách các học phần theo loại học phần, được giảng dạy ở học kỳ nào và các điều kiện học phần tiên quyết. **Ví dụ:** *"What are the course type, semester offered, and prerequisites for Operations Research and Linear Programming?"*	Danh sách tên các học phần được lọc theo điều kiện được nêu.
10	Truy xuất tên của các giảng viên dạy một nhóm học phần (tối đa 5).	Tìm kiếm tên của các giảng viên dạy một nhóm các học phần. **Ví dụ:** *"Who are the instructors for General Economics, Psychology, and Learning Skills?"*	Danh sách tên của các giảng viên, và tên học phần họ dạy.

Chương 2

Tổng quan về RAG và các nghiên cứu liên quan

Trong chương này, chúng tôi sẽ trình bày tổng quan về mô hình RAG, bao gồm hoàn cảnh ra đời, các ứng dụng tiêu biểu và các thành phần chính trong RAG cơ bản. Tiếp theo, chúng tôi sẽ điểm qua một số nghiên cứu gần đây. Cuối cùng là giới thiệu một biến thể mở rộng của RAG với sự tích hợp đồ thị tri thức, như một hướng tiếp cận tiềm năng để nâng cao khả năng suy luận của mô hình.

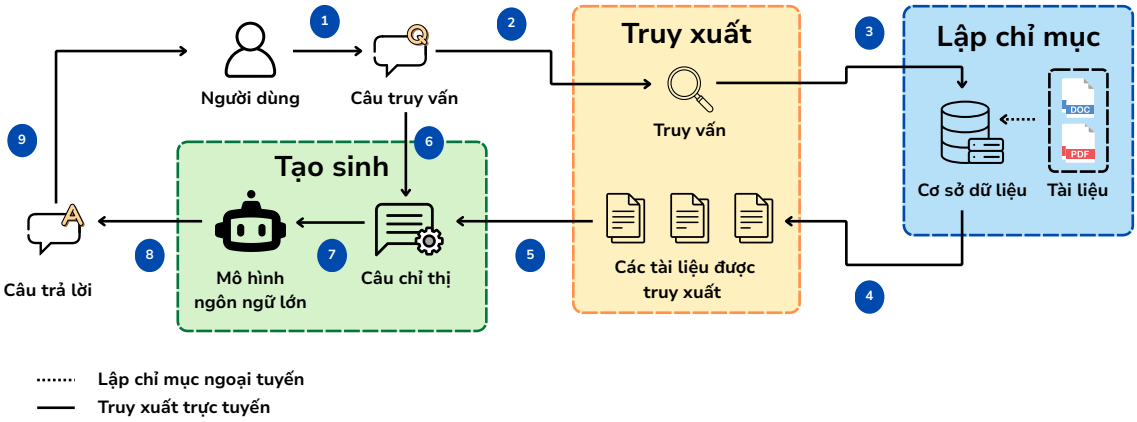
2.1 Tổng quan về RAG

Trong những năm gần đây, các mô hình ngôn ngữ lớn (Large Language Models - LLMs) được tiền huấn luyện và tinh chỉnh trên kho dữ liệu văn bản khổng lồ, đã chứng minh được khả năng lưu trữ và tái hiện kiến thức dưới dạng các tham số trong mô hình trong các tác vụ Xử lý ngôn ngữ tự nhiên (Natural Language Processing - NLP). Dù vậy, LLMs vẫn đối mặt với hai hạn chế nghiêm trọng khi được áp dụng vào các ứng dụng thực tiễn. Đầu tiên, LLMs không có khả năng tự cập nhật hay truy cập thông tin mới một cách nhanh chóng, nếu chúng không được huấn luyện lại. Mặt khác, việc mở rộng tham số lưu trữ cũng làm tăng chi phí tính toán đáng kể. Hạn chế tiếp theo của LLMs là hiện tượng ảo giác[1], khi chúng tiếp nhận các câu truy vấn về kiến thức mới, chưa được cập nhật và huấn luyện, thì LLMs sẽ có xu hướng đưa ra câu trả lời thiếu sự chính xác, hoặc thậm chí là sai sự thật (Lewis et al., 2020, Gao et al., 2023, Li et al., 2025b, Xu et al., 2024a).

Xuất phát từ thực tế trên, nhóm nghiên cứu tại Facebook AI Research (Lewis et al., 2020) đã đề xuất mô hình RAG, kết hợp sức mạnh của các mô hình tạo sinh ngôn ngữ với sự linh hoạt của các mô-đun truy xuất. Cụ thể, RAG hoạt động bằng cách truy xuất thông tin liên quan từ các cơ sở tri thức bên ngoài trước khi một mô hình ngôn ngữ lớn tạo sinh câu trả lời. Phương pháp này đã được nhóm nghiên cứu chứng minh là giúp cải thiện đáng kể độ chính xác của câu trả lời, giảm thiểu hiện tượng ảo giác, đặc biệt trong các tác vụ yêu cầu nhiều kiến thức chuyên sâu và chính xác. Đồng thời, RAG còn tạo điều kiện cho việc cập nhật kiến thức hiệu quả và tăng tính minh bạch bằng cách trích dẫn nguồn tài liệu, từ đó nâng cao độ chính xác của câu trả lời đầu ra. Nhờ việc tách

[1] hallucination

biệt kiến thức thực tế khỏi các tham số huấn luyện của LLMs, RAG cung cấp một giải pháp hiệu quả cho vấn đề thiếu hụt kiến thức và khả năng cập nhật kém của LLMs được tham số hóa[2].



Hình 2.1: Minh họa nguyên lý hoạt động tổng quát của các thành phần chính trong RAG với hai luồng hoạt động là: Lập chỉ mục ngoại tuyến và Truy xuất trực tuyến.

Hình 2.1 minh họa một mô hình RAG cơ bản sẽ được cấu thành từ ba thành phần cốt lõi lần lượt là (i) lập chỉ mục, (ii) truy xuất và (iii) tạo sinh ngôn ngữ thành câu trả lời. Ở giai đoạn đầu tiên, hệ thống sẽ bắt đầu lưu trữ một kho tri thức, thông qua việc tách các tài liệu đầu vào thành các đoạn văn bản nhỏ hơn, biểu diễn chúng dưới dạng các vector và lưu trữ trong cơ sở dữ liệu chuyên dụng. Qúa trình này được gọi là lập chỉ mục ngoại tuyến. Sau đó, ở giai đoạn trực tuyến, là khi hệ thống nhận được một câu truy vấn thì hệ thống sẽ tiến hành quá trình truy xuất các tài liệu liên quan trong cơ sở dữ liệu chuyên dụng. Các tài liệu được truy xuất, kết hợp với ngữ cảnh của câu truy vấn và câu chỉ thị được thiết kế sẽ giúp định hướng cho LLM tạo sinh câu trả lời cuối cùng.

Tóm lại, RAG là một mô hình trí tuệ nhân tạo tiên tiến, tích hợp truy xuất thông tin với các mô hình tạo sinh ngôn ngữ để nâng cao độ chính xác và tin cậy trong câu phản hồi. Bởi lẽ, các hệ thống truy hồi thông tin[3] có thể định vị hiệu quả các tài liệu có liên quan nhưng không thể tạo ra nội dung mới, trong khi LLMs tạo ra được văn bản trôi chảy nhưng phải dựa vào kiến thức được đào tạo trước.

2.2 Các nghiên cứu liên quan

Trong thời gian gần đây, cộng đồng nghiên cứu về RAG nhận ra rằng hiệu quả của RAG không chỉ phụ thuộc vào khả năng truy xuất các thông tin, tài liệu liên quan, mà còn phụ thuộc vào các thông tin, tài liệu đó được tích hợp và khai thác trong quá trình tạo sinh

[2]Parameterized language models
[3]Information Retrieval - IR

câu trả lời. Chính vì vậy, các nghiên cứu học thuật về RAG bắt đầu phân chia thành hai luồng tư duy rõ rệt. Luồng nghiên cứu thứ nhất tập trung vào việc nâng cao chất lượng truy xuất phù hợp với ngữ cảnh của câu truy vấn. Luồng nghiên cứu thứ hai thì lại chú trọng hơn vào việc cải thiện chất lượng tạo sinh câu trả lời. Ở phần tiếp theo, chúng tôi sẽ trình bày chi tiết về hai luồng nghiên cứu này.

2.2.1 Cải tiến chất lượng truy xuất

Một thành phần then chốt trong hệ thống RAG là mô-đun truy xuất, bởi lẽ chất lượng truy xuất sẽ quyết định mức độ phù hợp và hữu ích của ngữ cảnh đầu vào cho LLM. Do đó, nhiều nghiên cứu gần đây tập trung cải thiện giai đoạn truy xuất nhằm cung cấp thông tin chính xác, giảm nhiễu, và tăng độ bao phủ cho các truy vấn. Các hướng tiếp cận tiêu biểu bao gồm tích hợp mô-đun xếp hạng, nén và tinh lọc ngữ cảnh dựa trên tính nhất quán, kết hợp nhiều chiến lược truy xuất bổ sung nhau, hoặc thiết kế hệ thống truy xuất chuyên biệt cho các miền kiến thức cụ thể. Những cải tiến này không chỉ nâng cao độ chính xác của câu trả lời tạo sinh mà còn góp phần giảm thiểu hiện tượng ảo giác và chi phí xử lý không cần thiết.

Nghiên cứu của Yu et al. (2024) giới thiệu một khuôn[4] RAG mới lạ, được gọi là RankRAG. RankRAG tích hợp trực tiếp một mô-đun xếp hạng ngữ cảnh vào quy trình hoạt động của RAG, với mục tiêu là nâng cao mức độ liên quan của các tài liệu được truy xuất trước khi tạo sinh câu trả lời. Ở mô-đun truy xuất, RankRAG tận dụng phương pháp học để xếp hạng[5] để chấm điểm và sắp xếp lại các ngữ cảnh, dựa trên mức độ hữu dụng của chúng đối với LLM, từ đó dẫn đến chất lượng tạo sinh được cải thiện. Đóng góp chính của công trình nghiên cứu này nằm ở việc thống nhất hai giai đoạn là: truy xuất và xếp hạng, thành một hệ thống đồng nhất được tối ưu hóa đầu cuối[6], cho phép tạo ra các phản hồi giàu thông tin và phù hợp hơn về mặt ngữ cảnh. Một trong những thế mạnh của RankRAG là khả năng thích ứng, bởi lẽ hệ thống này có thể được huấn luyện để căn chỉnh việc xếp hạng tài liệu với thiên hướng phù hợp với câu trả lời tạo sinh của LLM, qua đó giảm thiểu vấn đề truy xuất các tài liệu tương đồng bề ngoài nhưng không hữu ích. Tuy nhiên, phương pháp này gây ra chi phí tính toán bổ sung cho giai đoạn xếp hạng; đồng thời, đòi hỏi dữ liệu được gán nhãn hoặc các tín hiệu giám sát mạnh[7] để có thể tinh chỉnh một cách hiệu quả. Bất chấp những hạn chế này, RankRAG đã đại diện cho một bước tiến đầy hứa hẹn hướng tới các hệ thống truy xuất được căn chỉnh tốt hơn với các tác vụ tạo sinh.

Nghiên cứu của Xu et al. (2024a) giới thiệu một bộ nén đa giai đoạn[8] được thiết kế để tinh lọc nội dung được truy xuất trước khi đưa vào một LLM. Đầu tiên, hệ thống áp dụng một bộ trích xuất tương phản[9] được huấn luyện để phân biệt các câu văn liên quan khỏi các câu nhiễu, bằng cách đối chiếu các đoạn văn bản hữu ích và liên quan với các đoạn không liên quan. Tiếp theo là giai đoạn chắt lọc dựa trên tính nhất quán[10], nơi đầu

[4] framework
[5] learning-to-rank
[6] end-to-end
[7] strong supervision signals
[8] multi-stage compressor
[9] contrastive extractor
[10] consistency-based distillation - đây là một trong những hình thức của chắt lọc tri thức (knowledge

ra của bộ nén được căn chỉnh với các biểu diễn ngữ nghĩa của một mô hình giáo viên, để đảm bảo sự mạch lạc và tính trung thực. Khuôn học nhất quán hai giai đoạn này tạo ra các bản tóm tắt súc tích, trung thực về mặt ngữ nghĩa, hỗ trợ tốt hơn cho các tác vụ hỏi-đáp ở giai đoạn sau, cải thiện độ chính xác precision và hiệu quả so với các phương pháp tiêu chuẩn[11]. Thế mạnh chính của hệ thống nằm ở việc giảm thiểu hiệu quả ngữ cảnh không liên quan hoặc gây hiểu lầm, qua đó nâng cao tính xác thực và sự tập trung của các câu trả lời được tạo sinh ra. Tuy nhiên, sự phức tạp gia tăng, bắt nguồn từ các hàm mất mát tương phản[12], chắt lọc giáo viên-học sinh, và huấn luyện tăng cường tính nhất quán—đòi hỏi thêm tài nguyên huấn luyện và các tín hiệu giám sát. Hơn nữa, trong các truy vấn đa khía cạnh, việc nén thông tin cần thiết có thể sẽ cắt tỉa những sắc thái hữu ích hoặc ngữ cảnh rộng hơn. Nhìn chung, phương pháp này mang lại một sự cải tiến hấp dẫn về chất lượng truy xuất và độ chính xác của câu trả lời, đổi lại bằng sự phức tạp của hệ thống tăng lên và những đánh đổi tiềm tàng về ngữ cảnh.

Nghiên cứu của Sawarkar et al. (2024) trình bày một phương pháp truy xuất mới lạ, được gọi là BlendedRAG. Hệ thống hợp nhất các kỹ thuật tìm kiếm ngữ nghĩa dựa trên từ khóa, vector dày đặc[13], và bộ mã hóa học thưa được huấn luyện bởi Elastic (2024) (Elastic Learned Sparse Encode - ELSER), thành một chiến lược truy xuất lai thống nhất. Cụ thể, hệ thống sử dụng: BM25 để đối chiếu thuật ngữ chính xác, các bộ nhúng dày đặc cho độ tương đồng ngữ nghĩa, và ELSER, kết hợp các thế mạnh bổ sung của ba thành phần này thông qua các truy vấn lai[14] tích hợp nhiều tín hiệu truy xuất. Bộ truy xuất kết hợp này đạt được những cải tiến đáng kể trên các bộ dữ liệu tiêu chuẩn như Natural Questions, TREC-COVID, và SQuAD, vượt trội hơn một số phương pháp tiêu chuẩn đã được tinh chỉnh. Ưu điểm chính của nó nằm ở việc cung cấp khả năng truy xuất chính xác và mạnh mẽ hơn trên các dạng câu truy vấn đa dạng bằng cách tận dụng các phương pháp truy xuất bổ sung cho nhau. Tuy nhiên, sự phức tạp gia tăng trong việc duy trì nhiều loại chỉ mục và thiết kế cẩn thận các truy vấn lai làm tăng chi phí hệ thống và đòi hỏi siêu dữ liệu phong phú cùng với sự tinh chỉnh để phát huy hết tiềm năng. Ngoài ra, hiệu quả của bộ truy xuất kết hợp phụ thuộc vào chất lượng, sự tinh chỉnh và sự cân bằng giữa các thành phần truy xuất cấu thành. Nhìn chung, BlendedRAG cung cấp một hướng đi mạnh mẽ hướng tới các hệ thống RAG đáng tin cậy và chính xác hơn, mặc dù đi kèm với sự phức tạp về kiến trúc và yêu cầu tinh chỉnh cao hơn.

Tiếp đến, trong nghiên cứu của Li et al. (2025a) với mô hình đề xuất là BiomedRAG. Đây một hệ thống RAG được thiết kế riêng cho các ứng dụng y sinh, có cơ chế đưa trực tiếp các đoạn văn bản (được phân tách từ các tài liệu đầu vào) vào một LLM, qua đó bỏ qua được các đoạn văn nhiễu. Hệ thống này sử dụng một bộ chấm điểm đoạn văn được giám sát bởi LLM để xác định ngữ cảnh hữu ích nhất nhằm cải thiện các dự đoán. Trong các đánh giá trên tám bộ dữ liệu bao gồm các tác vụ như trích xuất bộ ba[15],

distillation) được giới thiệu bởi Hinton et al. (2015). Đây là một trong những phương pháp thuộc Học chuyển giao (Transfer Learning), lấy ý tưởng chính từ quá trình học tập ở người khi mà kiến thức được truyền đạt từ giáo viên có hiểu biết tốt hơn tới người học có hiểu biết kém hơn. Trong kỹ thuật knowledge distillation thì một mô hình lớn hơn sẽ đóng vai trò là gíao viên (teacher) nhằm chuyển giao kiến thức sang mô hình nhỏ hơn đóng vai trò là học sinh (student).

[11] baselines

[12] contrastive loss functions

[13] dense-vector

[14] hybrid queries

[15] triple extraction

trích xuất quan hệ, phân loại và hỏi-đáp, BiomedRAG đã đạt được mức tăng hiệu suất trung bình ấn tượng khoảng 9.95% và vượt qua các phương pháp tiêu chuẩn. Các ưu điểm của BiomedRAG bao gồm sự đơn giản trong thiết kế, không cần đến các cơ chế cross-attention chuyên biệt, nhưng vẫn đảm bảo khả năng giảm nhiễu hiệu quả thông qua đầu vào cấp độ phân chia tài liệu, giúp giảm thiểu thông tin không liên quan. Về mặt hạn chế, việc phụ thuộc vào phương pháp phân chia tài liệu có thể bỏ lỡ các phạm vi ngữ cảnh rộng hơn, và bộ chấm điểm phân chia tài liệu được thiết kế riêng làm phát sinh thêm sự phức tạp trong huấn luyện và đòi hỏi tài nguyên trong quá trình truy xuất và chấm điểm. Nhìn chung, BiomedRAG cung cấp một phương pháp mạnh mẽ và có khả năng thích ứng, giúp nâng cao đáng kể mức độ liên quan và độ chính xác cho các thông tin truy xuất, từ đó hỗ trợ quy trình tạo sinh câu trả lời từ LLM ở phía sau.

2.2.2 Cải tiến chất lượng tạo sinh ngôn ngữ

Trong các nghiên cứu cải thiện về chất lượng mô-đun tạo sinh ngôn ngữ trong RAG, các phương pháp hầu như tập trung vào việc làm giàu ngữ cảnh đầu vào, tối ưu hóa chiến lược kết hợp giữa truy xuất và tạo sinh, hoặc khai thác hiệu quả khả năng xử lý ngữ cảnh dài của LLMs hiện đại. Mỗi hướng tiếp cận đều mang đến những ưu điểm riêng, từ tăng độ chính xác, giảm ảo giác, đến cải thiện chi phí triển khai và khả năng tổng quát hóa. Ba nghiên cứu tiêu biểu dưới đây minh họa rõ nét các chiến lược cải tiến đó, đồng thời phản ánh sự dịch chuyển từ các mô hình tạo sinh truyền thống sang các kiến trúc lai linh hoạt và thực tiễn hơn.

Nghiên cứu của Huang et al. (2023) đề xuất một phương pháp mới để cải thiện chất lượng tạo sinh trong hệ thống RAG bằng cách làm giàu ngữ cảnh được truy xuất với các biểu diễn câu trả lời có cấu trúc. Thay vì xem truy xuất và tạo sinh là hai bước riêng biệt, hệ thống này nhúng các câu trả lời ứng viên, chẳng hạn như các thực thể, nhãn, hoặc đoạn văn bản, trực tiếp vào các đoạn văn được truy xuất, cho phép LLM xác định và chú ý tốt hơn đến nội dung liên quan trong quá trình tạo sinh câu trả lời. Cơ chế mã hóa phong phú này hướng dẫn LLM tạo ra các câu trả lời chính xác và trung thực hơn, đặc biệt trong các tác vụ đòi hỏi nhiều kiến thức như hỏi-đáp và nhận dạng thực thể. Một ưu điểm lớn của phương pháp này là khả năng giảm thiểu hiện tượng ảo giác bằng cách làm nổi bật một cách tường minh các tín hiệu liên quan đến câu trả lời ngay trong đầu vào. Tuy nhiên, phương pháp này làm phát sinh thêm sự phức tạp, đòi hỏi các bộ mã hóa có cấu trúc và có thể cần đến sự giám sát để mã hóa các câu trả lời ứng viên, điều này có thể hạn chế khả năng tổng quát hóa sang các lĩnh vực hoặc tác vụ mới. Nhìn chung, việc mã hóa câu trả lời phong phú mang lại một sự cải tiến hấp dẫn cho các quy trình trong RAG, cân bằng giữa việc cải thiện độ chính xác với chi phí thiết kế và huấn luyện gia tăng.

Ngoài ra, nghiên cứu của Ram et al. (2023) giới thiệu một phương pháp đơn giản nhưng hiệu quả để cải thiện chất lượng tạo sinh bằng cách nối các tài liệu được truy xuất trực tiếp vào câu chỉ thị đầu vào của một mô hình ngôn ngữ được huấn luyện đóng băng[16], mà không yêu cầu bất kỳ sự tinh chỉnh mô hình hay thay đổi kiến trúc nào. Phương pháp này, được gọi là In-Context RALM, tận dụng các bộ truy xuất có sẵn (như BM25 hoặc các bộ truy xuất dày đặc) để cung cấp ngữ cảnh liên quan, giúp tăng cường

[16]frozen pre-trained language model

khả năng của mô hình ngôn ngữ trong việc tạo ra các câu trả lời chính xác và giàu thông tin hơn. Nghiên cứu cho thấy chiến lược này làm giảm đáng kể độ phức tạp[17], đạt được những cải tiến tương đương với việc tăng quy mô mô hình lên 2–3 lần, trong khi vẫn dễ dàng triển khai trên nhiều tác vụ khác nhau. Một thế mạnh lớn của phương pháp này là khả năng tương thích với bất kỳ LLM nào dựa trên API hoặc có trọng số đóng[18], khiến nó trở nên rất thực tiễn cho việc triển khai trong thực tế. Tuy nhiên, hiệu quả của phương pháp này phụ thuộc nhiều vào chất lượng truy xuất và các chiến lược tái xếp hạng, và nó có thể dẫn đến tăng độ trễ hoặc gặp giới hạn cửa sổ ngữ cảnh[19] khi có nhiều tài liệu được nối vào. Nhìn chung, In-Context RALM cung cấp một giải pháp gọn nhẹ nhưng mạnh mẽ để tăng cường khả năng tạo sinh câu trả lời.

Trong bài nghiên cứu của nhóm tác giả Li et al. (2024) cũng trình bày một so sánh chi tiết giữa các hệ thống RAG cơ bản và LLMs có ngữ cảnh dài (Long-Context LLM - LCLLM) hiện đại như GPT-4 và Gemini 1.5 về mặt chất lượng sinh. Mặc dù các LCLLM thường đạt hiệu suất cao hơn nhờ xử lý trực tiếp các cửa sổ văn bản lớn, nghiên cứu cho thấy RAG có thể tạo ra kết quả tương tự cho hơn 60% truy vấn, với chi phí tính toán thấp hơn đáng kể. Để thu hẹp khoảng cách này, các tác giả đề xuất SELF-ROUTE, một hệ thống lai có khả năng quyết định một cách linh hoạt xem nên trả lời bằng ngữ cảnh được truy xuất hay chuyển sang xử lý bằng ngữ cảnh dài khi cần thiết. Phương pháp này duy trì độ chính xác câu trả lời cao trong khi giảm lượng token sử dụng và độ trễ, giúp nó hiệu quả hơn về mặt chi phí cho các ứng dụng quy mô lớn. Lợi ích chính là khả năng thích ứng hệ thống, tận dụng thế mạnh của cả RAG và LCLLM tùy thuộc vào độ phức tạp của truy vấn. Tuy nhiên, hệ thống cũng làm phát sinh thêm logic định tuyến[20], sự phức tạp của hệ thống, và vẫn kế thừa những hạn chế của RAG đối với các câu hỏi đòi hỏi suy luận đa bước[21] hoặc kiến thức chuyên sâu.

2.2.3 RAG tích hợp đồ thị tri thức

Với mục tiêu nâng cao khả năng truy xuất tri thức chính xác và tạo sinh ngôn ngữ mạch lạc, nhiều nghiên cứu gần đây đã tập trung mở rộng RAG bằng cách tích hợp đồ thị tri thức[22] vào quy trình hoạt động. Các hệ thống tích RAG hợp đồ thị tri thức cho phép khai thác tri thức có cấu trúc để mô hình hóa mối quan hệ giữa các thực thể, từ đó hỗ trợ quá trình suy luận đa bước và giảm thiểu hiện tượng ảo giác trong tạo sinh câu trả lời. Những cải tiến này đặc biệt hữu ích trong các lĩnh vực như y sinh, dịch vụ khách hàng hay truy vấn chuyên biệt, nơi đòi hỏi hệ thống không chỉ sinh nội dung hợp ngữ pháp mà còn phải đảm bảo tính chính xác và khả năng giải thích. Trong phần này, một số hướng tiếp cận tiêu biểu sẽ được trình bày nhằm làm rõ cách thức đồ thị tri thức được tích hợp vào hệ thống RAG và hiệu quả mà chúng mang lại trong các ứng dụng thực tế.

Nhóm nghiên cứu (Matsumoto et al., 2024) đã giới thiệu KRAGEN là một khuôn RAG được tăng cường bằng cách tích hợp đồ thị tri thức. Hệ thống được phát triển để cải thiện khả năng giải quyết vấn đề trong lĩnh vực y sinh bằng cách kết hợp LLMs với

[17] perplexity
[18] closed-weight
[19] context window
[20] routing logic
[21] multi-hop reasoning
[22] knowledge graph

dữ liệu có cấu trúc. Hệ thống này chuyển đổi các đồ thị tri thức y sinh thành một cơ sở dữ liệu vector và tích hợp nó vào quy trình RAG để truy xuất các dữ kiện liên quan, cho phép tạo ra các câu trả lời chính xác và nhận thức rõ hơn về ngữ cảnh. Tận dụng các kỹ thuật ra chỉ thị tiên tiến như graph-of-thoughts (GoT), KRAGEN phân rã một cách linh hoạt các truy vấn y sinh phức tạp thành các vấn đề con nhỏ hơn, truy xuất chuyên biệt cho từng vấn đề, và tổng hợp các kết quả, qua đó giảm thiểu hiện tượng ảo giác và nhiễu thường gặp trong các phương pháp truy xuất tiêu chuẩn. Bằng cách tích hợp suy luận trên đồ thị, KRAGEN có thể xác định các mối quan hệ đa bước và các kết nối tiềm ẩn giữa các thực thể y sinh mà chỉ riêng việc truy xuất dựa trên văn bản thường bỏ lỡ. Một ưu điểm của hệ thống là khả năng tạo ra các kết quả đầu ra đáng tin cậy và có thể giải thích được, được hỗ trợ bởi các trực quan hóa đồ thị suy luận trên giao diện tương tác người dùng[23]. Tuy nhiên, hiệu quả của KRAGEN phụ thuộc nhiều vào tính đầy đủ và chính xác của đồ thị tri thức, và quy trình xây dựng cũng như tích hợp đồ thị làm phát sinh thêm sự phức tạp và chi phí tính toán. Bất chấp những thách thức này, KRAGEN đã cho thấy nhiều hứa hẹn trong các lĩnh vực y sinh nơi tri thức có cấu trúc là yếu tố then chốt để có được khả năng suy luận và sinh văn bản chất lượng cao.

Nghiên cứu của Dong et al. (2024) thì giới thiệu một khuôn RAG nâng cao, tích hợp các cấu trúc đồ thị để cải thiện khả năng suy luận tri thức phức tạp và sinh văn bản. Bằng cách nhúng các đồ thị có cấu trúc, chẳng hạn như đồ thị tri thức hoặc đồ thị ngữ nghĩa[24], vào quy trình truy xuất và tạo sinh, hệ thống có thể mô hình hóa các mối quan hệ giữa các thực thể một cách hiệu quả hơn, cho phép suy luận đa bước và hiểu biết ngữ cảnh sâu sắc hơn. Phương pháp tăng cường bằng đồ thị này giúp LLMs truy xuất các đường dẫn thông tin phù hợp hơn và tạo ra các câu trả lời mạch lạc, có cơ sở dữ kiện thực tế, đặc biệt là trong các lĩnh vực đòi hỏi nhiều tri thức chuyên sâu. Một ưu điểm lớn là khả năng nắm bắt tri thức liên kết chéo và giảm thiểu hiện tượng ảo giác bằng gắn nền LLMs vào các dữ kiện có cấu trúc. Tuy nhiên, việc xây dựng và duy trì các đồ thị chất lượng cao có thể tốn nhiều tài nguyên, và việc tích hợp chúng vào các quy trình tạo sinh làm tăng thêm sự phức tạp về mặt kiến trúc. Bất chấp những đánh đổi này, phương pháp này cho thấy tiềm năng mạnh mẽ trong việc nâng cao cả khả năng suy luận và độ chính xác về mặt dữ kiện của các mô-đun tạo sinh.

Nghiên cứu của nhóm tác giả Xu et al. (2024c) tại LinkedIn cải tiến phương pháp RAG cơ bản bằng cách xây dựng một đồ thị tri thức hai tầng từ các phiếu hỗ trợ[25] trong quá khứ, bảo toàn cả cấu trúc bên trong một phiếu và các mối quan hệ giữa các phiếu, thay vì xem chúng như những đoạn văn bản riêng lẻ. Trong quá trình truy vấn, hệ thống phân tích ý định của người dùng và ánh xạ nó tới các đồ thị con[26] liên quan trong đồ thị tri thức, sử dụng phương pháp truy xuất dựa trên không gian vector, để xác định các nút phiếu phù hợp. Các đồ thị con này sau đó được đưa vào một LLM, giúp tạo ra các câu trả lời mạch lạc và có cơ sở dữ kiện thực tế hơn, tránh được các vấn đề phát sinh từ việc phân đoạn văn bản. Trong các đánh giá, mô hình tăng cường bằng đồ thị đạt được mức tăng 77.6% về chỉ số Mean Reciprocal Rank và tăng 0.32 điểm BLEU so với các phương pháp tiêu chuẩn, tương đương với trung vị thời gian giải quyết sự cố nhanh hơn 28.6% khi triển khai thực tế trên LinkedIn. Điểm mạnh của phương pháp này nằm ở độ chính

[23]User Interface
[24]semantic graphs
[25]support tickets
[26]subgraphs

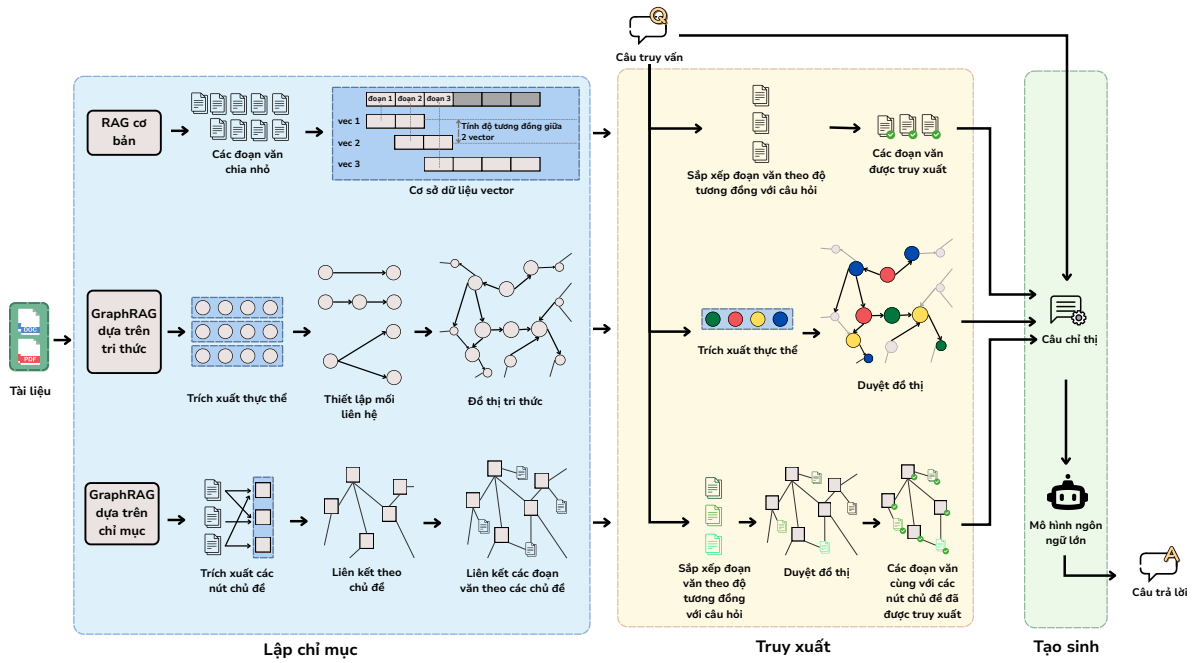
xác truy xuất được cải thiện, tính đầy đủ của câu trả lời và hiệu quả trong thực tế. Về mặt hạn chế, hệ thống hiện tại chưa có khả năng cập nhật đồ thị tri thức một cách linh hoạt theo thời gian thực dựa trên các truy vấn mới.

Chương 3

Cơ sở lý thuyết

Chương này trình bày các cơ sở lý thuyết làm nền tảng cho việc xây dựng và triển khai hệ thống tư vấn hỏi–đáp dựa trên mô hình RAG. Mở đầu, chúng tôi giới thiệu kiến trúc tổng thể cùng các thành phần chính của mô hình RAG cơ bản, cũng như những phiên bản mở rộng có tích hợp đồ thị tri thức. Tiếp theo, nhằm giúp độc giả hiểu rõ cách vận hành và liên kết giữa các thành phần trong hệ thống, từ giai đoạn lập chỉ mục, truy xuất đến tạo sinh ngôn ngữ, chúng tôi sẽ trình bày các thuật toán và phương pháp lý thuyết được ứng dụng trong từng mô-đun.

3.1 Kiến trúc tổng thể của mô hình RAG



Hình 3.1: Quy trình hoạt động tổng quan với ba giai đoạn chính của 3 mô hình biến thể của RAG lần lượt là: RAG cơ bản, GraphRAG dựa trên tri thức và GraphRAG dựa trên chỉ mục.

3.1.1 RAG cơ bản

Hãy hình dung bạn đang trò chuyện với một trợ lý ảo thông minh. Khi bạn đặt một câu hỏi như: *"Những kiến thức và kỹ năng gì được giảng dạy trong hai học phần Nhập môn máy học và Nhập môn trí tuệ nhân tạo?"*, và điều bạn mong đợi không chỉ là một câu trả lời chính xác mà còn là một câu trả lời được viết trôi chảy, có ngữ cảnh, thậm chí được tham chiếu từ những tài liệu đáng tin cậy. Đó chính là mục tiêu của mô hình RAG, một sự kết hợp giữa khả năng truy xuất thông tin và tạo sinh ngôn ngữ.

Như đã trình bày ở Phần 2.1, để xây dựng một hệ thống RAG cơ bản, chúng ta sẽ đi qua ba "chặng đường" chính: (i) Lập chỉ mục, (ii) Truy xuất tài liệu phù hợp, và (iii) Tạo sinh câu trả lời (Hình 2.1).

Lập chỉ mục

Hình 3.1 cho chúng ta thấy giai đoạn đầu tiên và nền tảng của một hệ thống RAG cơ bản là lập chỉ mục. Với mục tiêu là chuyển đổi văn bản thô thành một định dạng có cấu trúc để có thể dễ dàng tìm kiếm, từ đó tạo điều kiện thuận lợi cho việc truy xuất thông tin một cách hiệu quả và chính xác. Quá trình này được thực hiện thông qua bốn bước quan trọng.

1. **Tiền xử lý dữ liệu:** Trong lĩnh vực giáo dục, các tài liệu như sách giáo khoa (Gong et al., 2024), giáo trình (Taneja et al., 2024) và đề bài (Neumann et al., 2025) thường tồn tại dưới nhiều định dạng khác nhau như PDF, PowerPoint hay Word. Để chuẩn hóa các đầu vào này, quá trình tiền xử lý sẽ chuyển đổi chúng thành văn bản thuần túy hoặc có thể là dạng dữ liệu có cấu trúc, đồng thời loại bỏ các yếu tố nhiễu như: ký tự đặc biệt, khoảng trắng thừa hoặc dữ liệu không cần thiết. Bước này đảm bảo rằng các thao tác xử lý tiếp theo sẽ hoạt động trên một biểu diễn văn bản nhất quán và chất lượng cao.

2. **Phân đoạn văn bản:** Sau khi văn bản được tiền xử lý, bước tiếp theo là phân chia tài liệu thành các đoạn nhỏ hơn gọi là *chunk*. Mục tiêu của bước này là đảm bảo tính hiệu quả và chính xác cho giai đoạn truy xuất thông tin trong mô hình RAG, đồng thời tuân thủ giới hạn về độ dài cửa sổ ngữ cảnh của LLMs, vốn chỉ xử lý được một số lượng giới hạn token trong mỗi lần truy vấn.

3. **Vector hóa:** Sau khi tài liệu được phân chia thành các đoạn văn bản, mỗi đoạn sẽ được chuyển đổi thành một biểu diễn vector thông qua quá trình vector hóa. Quá trình này sử dụng các mô hình nhúng hiện đại như NVIDIA/NV-Embed-v2, Contriever, hoặc các phiên bản được phát triển bởi OpenAI (ví dụ: text-embedding-3-small, text-embedding-3-large). Các mô hình này ánh xạ nội dung văn bản vào một không gian vector, được sử dụng để tính toán độ tương đồng ngữ nghĩa giữa các đoạn văn, từ đó làm nền tảng cho giai đoạn truy xuất tiếp theo.

4. **Lưu trữ chỉ mục:** Sau khi mỗi đoạn văn bản được biểu diễn dưới dạng vector, hệ thống sẽ tiến hành lưu trữ một cách có hệ thống các vector này trong một cơ sở dữ liệu vector chuyên biệt. Mỗi vector sẽ được gán một mã định danh (chỉ mục) duy nhất, dùng để ánh xạ trở lại với nội dung văn bản gốc. Quá trình này không chỉ

đơn thuần là gán mã định danh, mà còn bao gồm cả việc xây dựng một cách tối ưu hóa cấu trúc dữ liệu, từ đó giúp quá trình truy xuất diễn ra nhanh chóng và hiệu quả hơn.

Giai đoạn truy xuất

Hình 3.1 cho thấy rằng sau khi dữ liệu đã được lập chỉ mục một cách có hệ thống, hệ thống RAG cơ bản sẽ bắt đầu nhận vào một câu truy vấn bên ngoài. Khi này, giai đoạn truy xuất sẽ đóng vai trò then chốt trong việc xác định và lấy ra các đoạn văn bản có liên quan nhất với câu truy vấn, từ trong cơ sở dữ liệu vector. Chất lượng và hiệu quả của quá trình này sẽ ảnh hưởng trực tiếp đến độ chính xác của câu trả lời được tạo sinh bởi LLM. Các phương pháp truy xuất có thể được phân loại thành hai nhóm chính, lần lượt là: truy xuất thưa[1] và truy xuất dày đặc[2]. Cụ thể:

1. **Truy xuất thưa:** Là một mô hình truy hồi thông tin hoạt động trên không gian vector đa chiều[3]. Trong đó, hầu hết các phần tử có giá trị bằng không (được gọi là "thưa") và chúng thường dựa vào các đặc trưng từ vựng[4] như Tần suất thuật ngữ - Tần suất nghịch đảo của tài liệu (Term Frequency – Inverse Document Frequency, viết tắt là TF-IDF). Một ví dụ nổi bật khác là Best Match 25 (BM25), một hàm xếp hạng được sử dụng rộng rãi, mở rộng từ TF-IDF bằng cách tích hợp thêm cơ chế chuẩn hóa độ dài tài liệu và mô hình hóa sự tương quan dựa trên xác suất[5] (Robertson and Zaragoza, 2009). Nhờ vào hiệu quả và khả năng diễn giải cao, BM25 vẫn được xem là một phương pháp tiêu chuẩn trong các nghiên cứu về RAG (Jiang et al., 2023, Gutiérrez et al., 2024, Gutiérrez et al., 2025). Ngoài BM25, Ngoài phương pháp đối sánh từ vựng truyền thống, K-Lân cận gần nhất (K-Nearest Neighbors - KNN) mang lại một cách tiếp cận linh hoạt hơn bằng cách tính toán độ tương đồng vector giữa câu truy vấn và tài liệu, ngay cả khi sự trùng lặp về từ vựng là rất nhỏ. Các nghiên cứu gần đây đã tận dụng phương pháp truy xuất dựa trên KNN để nâng cao hiệu suất trong giai đoạn truy xuất thông tin (Alon et al., 2022, Borgeaud et al., 2022).

2. **Truy xuất dày đặc:** Ngược lại với truy xuất thưa, phương pháp truy xuất dày đặc dựa trên các biểu diễn vector dày đặc[6], trong đó cả truy vấn và tài liệu đều được mã hóa bằng các mô hình nhúng ngữ nghĩa hiện đại - là các mô hình học sâu, vốn được xây dựng và phát triển dựa trên kiến trúc học sâu, chẳng hạn như Transformers. Nhờ khả năng biểu diễn ngữ nghĩa phong phú và không phụ thuộc vào sự trùng khớp từ vựng, phương pháp truy xuất dày đặc ngày càng đóng vai trò quan trọng trong cải thiện chất lượng và hiệu quả của các hệ thống RAG. Một ví dụ điển hình cho phương pháp này, chính là Truy xuất văn bản dày đặc (Dense

[1] sparse retrieval
[2] dense retrieval
[3] high-dimensional vector spaces
[4] lexical features
[5] probabilistic relevance modeling
[6] Dense Vector là vector chứa các mảng số thực và hầu hết hoặc toàn bộ phần tử đều khác không. So với vector thưa, vector dày đặc sẽ chứa nhiều thông tin hơn tại cùng một mức độ chiều, vì mỗi chiều đều mang giá trị có ngữ nghĩa.

Passage Retrieval - DPR). Nghiên cứu của Karpukhin et al. (2020) đã sử dụng kiến trúc bộ mã hóa kép, trong đó: một bộ mã hóa sẽ xử lý truy vấn, và bộ còn lại sẽ mã hóa tài liệu. Mức độ tương đồng giữa các vector sẽ được xác định bằng cách sử dụng tích vô hướng hoặc độ tương đồng cosine, cho phép truy xuất ngữ nghĩa tinh tế hơn. Ngoài DPR, các mô hình khác như ColBERTv2 và Contriever cũng đã được phát triển cho các tác vụ truy xuất dày đặc (Izacard and Grave, 2021, Santhanam et al., 2022, Borgeaud et al., 2022). Mặc dù có nhiều ưu điểm, truy xuất dày đặc vẫn đối mặt với những thách thức tính toán đáng kể, đòi hỏi cập nhật liên tục các biểu diễn vector và tính toán độ tương đồng trong một không gian vector đa chiều. Điều này dẫn đến sự tăng đáng kể trong chi phí tính toán và tài nguyên bộ nhớ.

Giai đoạn tạo sinh câu trả lời

Giai đoạn cuối cùng trong hệ thống RAG cơ bản là tạo sinh câu trả lời (Hình 3.1) bởi LLM. Để đưa ra được câu trả lời chính xác và phù hợp với ngữ cảnh của câu truy vấn, LLM sẽ nhận được các dữ liệu đầu vào, bao gồm: các đoạn văn đã được truy xuất, câu truy vấn và câu lệnh chỉ thị. Giai đoạn cuối cùng này sẽ bao gồm hai khía cạnh chính, lần lượt là:

1. **Xử lý và hợp nhất đoạn văn được truy xuất:** Hiệu quả của RAG phụ thuộc vào việc thông tin được truy xuất sẽ được xử lý và tích hợp tốt như thế nào vào quá trình tạo sinh. Một cách tiếp cận đơn giản là tạo sinh dựa trên phương pháp kết nối chuỗi[7] - nơi các đoạn văn bản được truy xuất sẽ kết nối trực tiếp với câu truy vấn, tạo thành một chuỗi văn bản dài để đưa vào LLM (Lewis et al., 2020). Mặc dù đơn giản, phương pháp này có thể dẫn đến sự dư thừa hoặc vượt quá độ dài cửa sổ ngữ cảnh của LLM khi xử lý một lượng lớn các tokens. Để cải thiện hạn chế này, một số cách tiếp cận khác, chẳng hạn như: lọc theo độ liên quan (Asai et al., 2024), nén thông tin (Glass et al., 2022) và sắp xếp theo thứ hạng và chọn ra k đoạn văn được truy xuất (Xu et al., 2024b, Gutiérrez et al., 2024, Gutiérrez et al., 2025). Một kỹ thuật khác là Hợp nhất trong Bộ giải mã[8], xử lý các đoạn văn đã truy xuất một cách riêng biệt. Kỹ thuật này cho phép mô hình cân nhắc đóng góp của các đoạn văn một cách linh hoạt trong quá trình tạo sinh, từ đó tổng hợp câu trả lời mạch lạc hơn (Izacard and Grave, 2021). Ngoài ra, các phương pháp Hợp nhất thích ứng[9] tận dụng cơ chế Attention hoặc chiến lược sắp xếp, nhằm kiểm soát một cách linh động ảnh hưởng của các đoạn văn đã truy xuất trong quá trình giải mã. Từ đó giúp giảm thiểu hiện tượng ảo giác (Guu et al., 2020).

2. **Các chiến lược tối ưu hóa cho LLMs:** Để nâng cao chất lượng tạo sinh câu trả lời, nhiều phương pháp tối ưu hóa được áp dụng cho LLM. Trong đó, một phương pháp cơ bản là thiết kế câu chỉ thị, hay còn gọi là *prompt engineering*. Mục tiêu là thiết kế các câu chỉ thị có cấu trúc, nhằm hướng dẫn LLM cách sử dụng các thông tin đã truy xuất một cách chính xác và hiệu quả. Ví dụ như trong hệ thống In-Context RALM, được phát triển bởi nhóm nghiên cứu Ram et al. (2023), sẽ duy

[7] Concatenation-based generation
[8] Fusion-in-Decoder
[9] Adaptive Fusion

trì các tham số của LLM ở trạng thái đóng băng. Song, hệ thống sẽ tích hợp đoạn văn đã truy xuất thông qua các kỹ thuật hợp nhất với câu chỉ thị. Một cách tiếp cận khác là tinh chỉnh - nơi LLM sẽ được huấn luyện trên các bộ dữ liệu chuyên biệt để nắm bắt tốt hơn các thuật ngữ chuyên ngành và đảm bảo tính liên quan theo ngữ cảnh (Cheng et al., 2023, Karpukhin et al., 2020).

Những hạn chế của hệ thống RAG cơ bản

Mặc dù hệ thống RAG cơ bản đã được nghiên cứu, phát triển và ứng dụng rộng rãi trong những năm gần đây, nhưng hệ thống này vẫn bộc lộ hai hạn chế nhất định (Gao et al., 2023, Procko and Ochoa, 2024, Liu et al., 2025, Li et al., 2025b):

1. **Sự phân tán và mất mát ngữ cảnh trong kiến thức chuyên ngành:** Các truy vấn trong lĩnh vực chuyên sâu thường chứa các thuật ngữ đặc thù, đòi hỏi phải được diễn giải trong bối cảnh ngữ nghĩa phù hợp để đảm bảo độ chính xác. Tuy nhiên, những kiến thức chuyên ngành này thường sẽ có đặc điểm phân bố rời rạc, trải rộng trên nhiều tài liệu và nguồn dữ liệu không đồng nhất. Để xử lý đặc tính phân tán này, các mô hình RAG cơ bản thường áp dụng chiến lược chia nhỏ tài liệu thành các đoạn nhỏ nhằm phục vụ cho việc đánh chỉ mục và truy xuất hiệu quả hơn. Tuy nhiên, quá trình này phải đánh đổi tính toàn vẹn ngữ cảnh và làm mất đi các kết nối giữa các vùng thông tin quan trọng. Hệ quả là các mô hình truy xuất có thể bỏ sót các đoạn văn bản thiết yếu, từ đó khiến LLM dễ rơi vào trạng thái ảo giác vì không đủ thông tin và dữ kiện để trả lời câu truy vấn.

2. **Giới hạn suy luận và sự lệ thuộc quá mức vào đoạn văn truy xuất:** Trong giai đoạn tạo sinh, LLM tiếp nhận thông tin từ các đoạn văn đã được truy xuất. Tuy nhiên, trong nhiều trường hợp, LLM có xu hướng lệ thuộc quá mức vào các đoạn văn này và bỏ qua các logic nội tại hoặc ngữ cảnh của câu truy vấn. Đặc biệt là khi các truy vấn yêu cầu hệ thống phải suy luận đa bước hoặc tổng hợp tri thức từ nhiều nguồn tài liệu. Điều này khiến cho câu trả lời của LLM thiếu tính lập luận, không đáp ứng được yêu cầu của các tác vụ phức tạp trong các lĩnh vực như giáo dục, y sinh, kỹ thuật, hoặc luật – nơi mà việc hiểu sâu, lập luận chính xác và kết nối dữ kiện là yếu tố then chốt.

3.1.2 RAG tích hợp cấu trúc đồ thị

Trong các phần trước, chúng tôi đã trình bày kiến trúc tổng thể của một mô hình RAG cơ bản, được thiết kế với một chu trình hợp lý, và trong nhiều trường hợp, đã mang lại hiệu quả đáng kể. Tuy nhiên, giống như việc đọc một cuốn sách bị xé rời từng trang, hạn chế của hệ thống sử dụng RAG cơ bản là xử lý dữ kiện một cách rời rạc và thiếu kết nối ngữ nghĩa sâu sắc, đặc biệt là khi đối mặt với các truy vấn cần đa bước suy luận.

Chính vì vậy, sự kết hợp giữa RAG và cấu trúc đồ thị, hay còn được gọi là *GraphRAG* xuất hiện như một bước tiến chiến lược nhằm khắc phục những hạn chế của hệ thống RAG cơ bản. GraphRAG sử dụng cấu trúc đồ thị để tổ chức, liên kết và lưu trữ thông tin một cách logic và có hệ thống. Tùy theo cách mà đồ thị được tích hợp vào mô hình, Gao

et al. (2023) đã phân chia GraphRAG thành ba nhóm chính: (i) GraphRAG dựa trên tri thức; (ii) GraphRAG dựa trên chỉ mục; và (iii) GraphRAG kết hợp. Mỗi loại GraphRAG là một cách định hình khác nhau về việc sử dụng cấu trúc đồ thị trong RAG. Ngoài ra, Hình 3.1 đã minh họa rõ ràng về quy trình hoạt động tổng quát của mô hình (i) và (ii).

GraphRAG dựa trên tri thức

Mô hình GraphRAG dựa trên tri thức (Knowledge-based GraphRAG, viết tắt là KB-GraphRAG), là một mô hình RAG tích hợp cấu trúc đồ thị, trong đó: các nút (thực thể) đại diện cho các danh từ, cụm danh từ và các cạnh liên kết sẽ biểu diễn mối liên hệ giữa các nút tương ứng. Từng giai đoạn trong mô hình KB-GraphRAG được minh họa ở Hình 3.1, và trình bày ngắn gọn, so sánh với mô hình RAG cơ bản thông qua Bảng 3.1.

GraphRAG dựa trên chỉ mục

Nếu như KB-GraphRAG là hành trình kiến tạo một bản đồ tri thức có cấu trúc - nơi mỗi thực thể đều được định vị và liên kết bằng logic rõ ràng, thì GraphRAG dựa trên chỉ mục (Index-based GraphRAG, viết tắt là IB-GraphRAG) lại chọn một con đường khác. IB-GraphRAG sẽ trích xuất các chủ đề trong tập tài liệu đầu vào. Có hai kiểu nút được lưu trữ trong đồ thị là: nút đại diện cho chủ đề và nút đại diện cho đoạn văn. Các nút chủ đề sẽ được liên kết với nhau theo độ tương đồng về chủ đề. Tiếp đến, các nút đoạn văn sẽ được liên kết với các nút chủ đề tương ứng. Từng giai đoạn trong mô hình IB-GraphRAG được minh họa ở Hình 3.1, và trình bày ở Bảng 3.1.

Bảng 3.1: Bảng so sánh 3 quy trình hoạt động tổng quát của mô hình RAG cơ bản, KB-GraphRAG và IB-GraphRAG.

Giai đoạn	**RAG cơ bản**	**KB-GraphRAG**	**IB-GraphRAG**
Lập chỉ mục	Chuyển đổi văn bản thô thành các đoạn văn bản nhỏ, sau đó vector hóa và lưu trữ chỉ mục trong cơ sở dữ liệu vector.	Chuyển hóa tri thức từ dạng phi cấu trúc thành cấu trúc đồ thị rõ ràng (nút là thực thể, cạnh là mối liên hệ ngữ nghĩa). Sử dụng Trích xuất thực thể (Named Entity Recognition - NER), hoặc Khai thác thực thể mở (Open Information Extraction - OpenIE) để tạo bộ ba tri thức.	Giữ lại sự phong phú của văn bản gốc bằng cách chia văn bản thành đoạn nhỏ, mã hóa vector, sau đó trích xuất nút chủ đề và tạo cạnh liên kết dựa trên sự tương đồng chủ đề chung. Tiếp đến là lưu trữ nút đoạn văn và liên kết chúng với các nút chủ đề tương ứng.

Tiếp tục vào trang sau.

Bảng 3.1 - Tiếp tục từ trang trước.

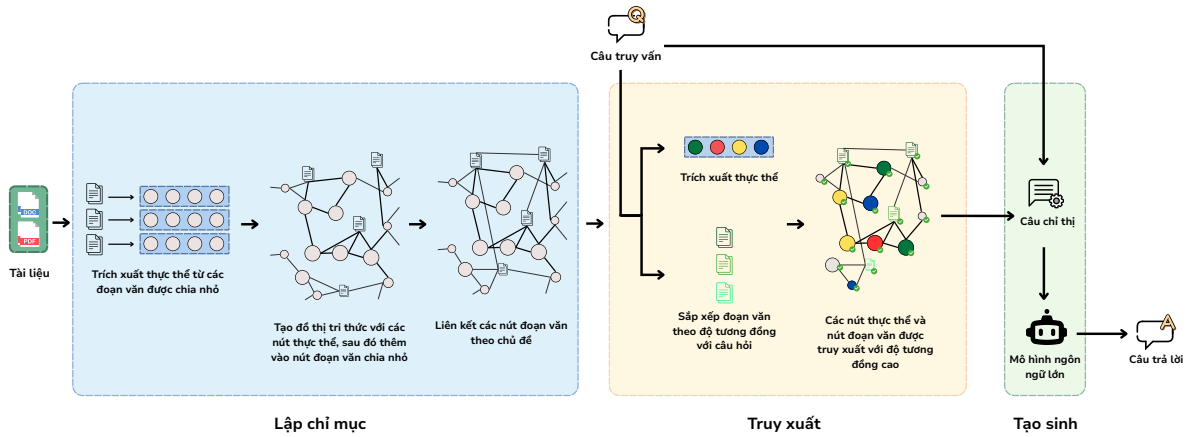
Giai đoạn	RAG cơ bản	KB-GraphRAG	IB-GraphRAG
Truy xuất	Xác định và lấy các đoạn văn bản liên quan nhất từ cơ sở dữ liệu đã chỉ mục thông qua truy xuất thưa hoặc truy xuất dày.	Thực hiện suy luận trong đồ thị tri thức. Gắn nhãn thực thể trong truy vấn để xác định nút xuất phát, sau đó sử dụng các thuật toán duyệt đồ thị như Tìm kiếm theo chiều sâu (Depth-first Search - DFS), Tìm kiếm theo chiều rộng (Breadth-first Search - BFS) (Wu et al., 2025) để thu thập thông tin và hỗ trợ suy luận đa bước.	Mã hóa truy vấn thành vector sau đó tìm Top-K các đoạn văn có độ tương đồng cao nhất. Tiếp đến, kết hợp mở rộng truy xuất bằng cách duyệt đồ thị chủ đề. (Liu et al., 2025).
Tạo sinh văn bản	Tích hợp thông tin được truy xuất cùng với câu chỉ thị và câu truy vấn, làm dữ liệu đầu vào cho LLM tạo sinh câu trả lời.	Đầu vào của LLM là một bộ các nút và cạnh đại diện cho các chuỗi logic, kết hợp với câu chỉ thị và câu truy vấn, để tạo sinh câu trả lời logic.	Đâu vào của LLM là một bộ các nút chủ đề và đoạn văn, kết hợp với câu chỉ thị và câu truy vấn, để tạo sinh câu trả lời đầy đủ ngữ nghĩa.

GraphRAG kết hợp

Mặc dù KG-GraphRAG sở hữu khả năng lập luận đa bước mạnh mẽ nhờ vào cấu trúc quan hệ rõ ràng giữa các thực thể, nhưng lại gặp hạn chế ở khả năng bao phủ thông tin về chủ đề giữa các đoạn văn hoặc tài liệu. Mặt khác, IB-GraphRAG lại thể hiện thế mạnh trong việc bao phủ thông tin về chủ đề, nhưng lại thiếu đi khả năng suy luận đa bước một cách logic và chuyên sâu. GraphRAG kết hợp (Hybrid GraphRAG, viết tắt là H-GraphRAG) ra đời như một giải pháp kết hợp hai điểm mạnh của KB-GraphRAG và IB-GraphRAG. Hình 3.2 cho thấy rằng trong quy trình xây dựng đồ thị của H-GraphRAG, sẽ có hai loại nút được tạo thành, lần lượt là: thực thể và đoạn văn. Các nút thực thể sẽ được liên kết với nhau thông qua các mối liên hệ về ngữ nghĩa hoặc từ đồng nghĩa. Sau đó, các nút đoạn văn sẽ được liên kết với nhau thông qua mối liên hệ về chủ đề. Chi tiết quy trình tổng quát của H-GraphRAG diễn ra như sau:

Giai đoạn lập chỉ mục: Trong H-GraphRAG, quá trình lập chỉ mục bắt đầu từ việc chia văn bản gốc thành các đoạn văn nhỏ để trích xuất các thực thể. Việc xây dựng đồ thị từ các thực thể sẽ tương tự với KB-GraphRAG, nhưng ở H-GraphRAG sẽ tích hợp thêm vào các nút đoạn văn nhằm liên kết các thực thể liên quan với chúng. Điều này sẽ củng cố mối quan hệ chặt chẽ và bao phủ hơn khi tiếp tục tạo các cạnh liên kết giữa các nút đoạn văn này với nhau theo chủ đề.

Giai đoạn truy xuất: Khi người dùng gửi truy vấn, H-GraphRAG sẽ thực hiện



Hình 3.2: Quy trình hoạt động tổng quan với ba giai đoạn chính của mô hình H-GraphRAG.

đồng thời hoặc lần lượt hai tác vụ là: trích xuất các thực thể từ câu truy vấn và mã hóa câu truy vấn để tìm Top-K các đoạn văn có độ tương đồng cao với câu truy vấn trong không gian vector. Sau khi đã xác định được các thực thể cũng như các đoạn văn liên quan nhất, H-GraphRAG tiến hành truy vết trên đồ thị để tìm ra một bộ các nút thực thể và đoạn văn liên quan nhất với câu truy vấn. Chuẩn bị cho giai đoạn cuối cùng.

Giai đoạn tạo sinh văn bản: Sau khi hoàn tất truy xuất từ đồ thị, hệ thống bước vào giai đoạn cuối cùng là tạo sinh câu trả lời. Một bộ các nút thực thể và đoạn văn có chung chủ đề và ngữ nghĩa liên quan nhất với câu truy vấn, cùng với câu chỉ thị để hướng dẫn LLM tiếp nhận, tổng hợp và khai thác tất cả các thông tin này một cách logic và có hệ thống để tạo sinh ra câu trả lời.

3.2 Các phương pháp đo tương đồng chuỗi

Bước đầu tiên trong giai đoạn lập chỉ mục (Phần 3.1) chính là tiền xử lý dữ liệu đầu vào. Mục tiêu của bước này chính là chuyển hóa và làm sạch dữ liệu thô ban đầu thành dữ liệu có cấu trúc. Trong phần này, chúng tôi sẽ trình bày về 4 thuật toán đo độ tương đồng giữa hai chuỗi, lần lượt là: Khoảng cách Levenshtein, Dãy con chung dài nhất (Longest Common Subsequence - LCS), hệ số Jaccard và Jaro-Winkler. Đây đều là những thuật toán mà chúng tôi đã được học trong học phần Khai phá Dữ liệu. Vì vậy, chúng tôi muốn ứng dụng các thuật toán này vào giai đoạn tiền xử lý dữ liệu đầu vào cho hệ thống tư vấn hỏi-đáp học phần. Mục tiêu là để so khớp các tên học phần tiên quyết bị lỗi (do quá trình nhập thủ công và được trình bày chi tiết trong Phần 4.1.1) với lần lượt từng tên học phần trong danh sách các tên học phần chuẩn. Từ đó, dựa vào cặp so khớp nào có độ tương đồng cao nhất, ta có thể xác định chính xác tên học phần tiên quyết tương ứng.

3.2.1 Các phương pháp dựa trên quy hoạch động

Trong quá trình học tập và nghiên cứu về khai phá dữ liệu, chúng tôi đã tiếp cận nhiều thuật toán kinh điển, trong đó có hai thuật toán nổi bật là Khoảng cách Levenshtein và LCS. Cả hai đều được trình bày chi tiết trong giáo trình của Aggarwal (2015). Một điểm chung của hai thuật toán này là chúng đều dựa trên quy hoạch động. Đây là một kỹ thuật lập trình tối ưu mạnh mẽ, được thiết kế để giải quyết các bài toán có cấu trúc con lặp lại[10] và tính chất tối ưu cấu trúc con[11]. Thay vì giải bài toán lớn một cách trực tiếp, quy hoạch động chia nhỏ vấn đề thành các bài toán con, giải từng bài toán một lần duy nhất và lưu trữ kết quả để tái sử dụng.

Một trong những điều quan trọng khi nói về hai thuật toán Khoảng cách Levenshtein và LCS chính là hướng tiếp cận. Nếu như Khoảng cách Levenshtein tập trung trả lời câu hỏi *"Chuỗi này khác chuỗi kia bao nhiêu?"*, thì LCS sẽ tập trung vào câu hỏi *"Chuỗi này giống chuỗi kia bao nhiêu?"*. Dù cả hai thuật toán đều giải quyết bài toán về so sánh hai chuỗi, nhưng Khoảng cách Levenshtein là một hàm đo khoảng cách, tức nhấn mạnh vào sự khác biệt giữa hai chuỗi. Mặt khác, LCS là một hàm đo độ tương đồng, tức nhấn mạnh vào sự giống nhau giữa hai chuỗi.

Khoảng cách Levenshtein

> *"Khoảng cách chỉnh sửa được định nghĩa là độ đo giữa hai dãy ký tự, phản ánh lượng "nỗ lực" (hoặc chi phí) tối thiểu cần thiết để chuyển đổi một dãy thành dãy kia thông qua một chuỗi các thao tác chuyển đổi, gọi là các chỉnh sửa. Thuật ngữ này còn được biết đến với tên gọi khoảng cách Levenshtein.[12]"*
>
> *- Aggarwal (2015) trang 82.*

Vào những năm 1965 và 1966, Levenshtein (1965) đề xuất một mô hình toán học để đo độ khác biệt giữa hai chuỗi ký tự nhị phân nhằm giải quyết bài toán viễn thông về sửa lỗi tín hiệu, bằng cách xác định số lượng tối thiểu các phép toán cần thiết như chèn, xóa và thay thế ký tự, để biến đổi một chuỗi thành chuỗi còn lại. Bài báo này đã đặt một cột mốc về sự khai sinh của khái niệm *Khoảng cách Levenshtein*, hay còn được gọi với cái tên khác là *Khoảng cách chỉnh sửa*[13].

Điều đáng chú ý là, dù trải qua hơn năm thập kỷ tồn tại, Khoảng cách Levenshtein vẫn còn giữ được giá trị trong các nghiên cứu hiện đại. Một ví dụ gần đây là công trình của Logan et al. (2022) giới thiệu một phiên bản tối ưu hóa của thuật toán Levenshtein để phân cụm dữ liệu từ các công nghệ giải trình tự thế hệ thứ ba, như PacBio và Oxford Nanopore trong lĩnh vực y-sinh học. Ngoài ra, Da et al. (2022) áp dụng mô hình học sâu Levenshtein Transformation của Gu et al. (2019) trong nhận dạng ký tự quang học, đặc

[10] overlapping subproblems
[11] optimal substructure
[12] The edit distance defines the distance between two strings as the least amount of "effort" (or cost) required to transform one sequence into another by using a series of transformation operations, referred to as "edits." The edit distance is also referred to as the Levenshtein distance.
[13] Edit distance

biệt là trong việc xử lý các lỗi nhận dạng bằng cách sử dụng khoảng cách chỉnh sửa để cải thiện độ chính xác.

Như đã đề cập ở trên, khoảng cách Levenshtein là một phương pháp định lượng số bước cần thiết để chuyển đổi một chuỗi ký tự thành chuỗi khác, sử dụng ba thao tác cơ bản: chèn, xóa, và thay thế một ký tự. Mỗi thao tác mang một chi phí, và thuật toán sẽ tìm đường đi có chi phí nhỏ nhất giữa hai chuỗi. Đây là một ví dụ điển hình của quy hoạch động, nơi mỗi bước được tính dựa trên các bước con trước đó.

Giả sử ta có hai chuỗi $\overline{X} = (x_1 x_2 \dots x_m)$, và $\overline{Y} = (y_1 y_2 \dots y_n)$. Với m, n lần lượt là độ dài của chuỗi $\overline{X}, \overline{Y}$. Ta định nghĩa Edit$(i, j)$ là chi phí tối thiểu để chuyển đổi đoạn đầu tiên của chuỗi $\overline{X}$ (bao gồm các ký tự tính từ x_1 đến x_i) thành đoạn đầu tiên của chuỗi $\overline{Y}$ (bao gồm các ký tự tính từ y_1 đến y_j). Khi đó, ba trường hợp có thể sẽ xảy ra cho ký tự cuối cùng trong hai chuỗi con $\overline{X_i}$ và $\overline{Y_j}$ như sau:

1. **Xóa ký tự cuối cùng của $\overline{X}$:** Trường hợp này tương ứng với việc bỏ qua x_i, và ta chỉ cần so khớp $\overline{X_{i-1}}$ với $\overline{Y_j}$. Chi phí sẽ là [Edit$(i, j-1)$ + Chi phí xóa].

2. **Chèn ký tự vào cuối $\overline{X_i}$:** Ta chèn y_j vào cuối chuỗi $\overline{X}$ để khớp với phần tử cuối cùng của $\overline{Y_j}$. Khi đó, chi phí chuyển đổi là [Edit$(i, j-1)$ + Chi phí chèn].

3. **Thay thế (hoặc giữ nguyên nếu giống nhau):** Nếu $x_i = y_j$, chi phí sẽ là 0; nếu khác nhau, cần thay thế với chi phí tương ứng. Khi đó, ta tiếp tục xử lý phần còn lại $\overline{X_{i-1}}$ và $\overline{Y_{j-1}}$ với chi phí [Edit$(i-1, j-1) + I_{ij}$]. Trong đó, I_{ij} là 0 nếu $x_i = y_j$ và là 1 nếu ngược lại.

Như vậy, công thức quy hoạch động được xây dựng như sau:

$$Edit(i,j) \quad = \quad \min \left\{ \begin{array}{l} Edit(i-1,j) + \text{Chi phí xóa}, \\ Edit(i,j-1) + \text{Chi phí chèn}, \\ Edit(i-1,j-1) + I_{ij} \end{array} \right\}. \tag{3.1}$$

Công thức (3.1) phản ánh quá trình lựa chọn phép biến đổi tối ưu ở từng bước - dựa trên lịch sử chuyển đổi của các phần tử trước đó, và tiến dần đến lời giải toàn cục, thông qua việc xây dựng bảng giá trị $Edit(i, j)$ tương đương với ma trận $E \in \mathbb{Z}^{(m+1)\times(n+1)}$.

Trong thực tế, bởi lẽ chi phí chèn và xóa thường được giả định là bằng nhau, dẫn đến hàm tính khoảng cách sẽ trở nên đối xứng. Khi đó, khoảng cách biến đổi từ chuỗi $\overline{X}$ đến $\overline{Y}$ cũng chính là khoảng cách biến đổi từ chuỗi $\overline{Y}$ đến $\overline{X}$, và không thay đổi tổng chi phí (Aggarwal, 2015, trang 83). Ngoài ra, vì Khoảng cách Levenshtein đo lường mức độ khác nhau giữa hai chuỗi, nên giá trị số nguyên $Edit(i, j)$ càng nhỏ thì hai chuỗi càng giống nhau, và $Edit(i, j) = 0$ khi hai chuỗi trùng nhau.

LCS

> *"Một dãy con của một dãy là một tập hợp các ký hiệu được lấy ra từ dãy ban đầu theo cùng thứ tự xuất hiện. Dãy con khác với chuỗi con ở chỗ các phần tử trong dãy con không nhất thiết phải liên tiếp nhau, trong khi các phần tử*

trong chuỗi con phải liên tiếp nhau.[14] "

- *Aggarwal (2015) trang 84.*

Ý tưởng cốt lõi của thuật toán LCS xoay quanh việc so sánh từng phần tử của hai chuỗi đầu vào là $\overline{X} = (x_1 x_2 \dots x_m)$, và $\overline{Y} = (y_1 y_2 \dots y_n)$. Với m, n lần lượt là độ dài của chuỗi $\overline{X}, \overline{Y}$. Ta định nghĩa $LCS(i, j)$ là độ dài LCS giữa đoạn đầu tiên của chuỗi $\overline{X}$ (bao gồm các ký tự tính từ x_1 đến x_i) và đoạn đầu tiên của chuỗi $\overline{Y}$ (bao gồm các ký tự tính từ y_1 đến y_j). Từ đó, hai trường hợp có thể xảy ra:

1. **Nếu** $x_i = y_j$**:** thì ta có $LCS(i, j) = LCS(i-1, j-1) + 1$

2. **Nếu** $x_i \neq y_j$**:** thì phần tử cuối cùng của ít nhất một trong hai chuỗi cần phải bị xóa với giả định rằng nó không thể xuất hiện trong phép khớp. Trong trường hợp này, tùy thuộc vào chuỗi nào được lựa chọn để xóa, thì giá trị của $LCS(i, j)$ có thể là $LCS(i, j-1)$ hoặc là $LCS(i-1, j)$

Như vậy, công thức quy hoạch động được xây dựng như sau:

$$LCS(i,j) = \max \left\{ \begin{array}{ll} LCS(i-1, j-1) + 1 & \text{chỉ khi } x_i = y_j, \\ LCS(i-1, j) & \text{ngược lại (không khớp với } x_i), \\ LCS(i, j-1) & \text{ngược lại (không khớp với } y_j) \end{array} \right\}. \quad (3.2)$$

Bảng giá trị $LCS(i, j)$ tương đương với ma trận $L \in \mathbb{Z}^{(m+1)\times(n+1)}$ và kết quả của thuật toán sẽ là số nguyên ≥ 0, biểu diễn độ dài của dãy con chung dài nhất giữa 2 chuỗi đầu vào. Bởi lẽ chỉ quan tâm đến độ dài của các ký tự chung có thứ tự giống nhau và không ảnh hưởng đến kết quả cuối cùng, nên LCS được xem là có mang tính chất đối xứng, cụ thể $LCS(\overline{X}, \overline{Y}) = LCS(\overline{Y}, \overline{X})$.

Ý nghĩa điều kiện biên xuất phát từ bản chất của thuật toán là *tìm độ dài dãy con chung dài nhất giữa 2 chuỗi X và Y*. Vì vậy, nếu một chuỗi là rỗng, thì LCS luôn có độ dài bằng 0, tức không có ký tự chung nào để so khớp. Ngược lại, giá trị $LCS(\overline{X}, \overline{Y})$ càng lớn thì hai chuỗi càng giống nhau.

3.2.2 Hệ số Jaccard

"Chỉ số tương đồng do Paul Jaccard đề xuất cách đây 120 năm đã trở thành một trong những hệ số được biết đến rộng rãi nhất trong sinh thái học thống kê và các lĩnh vực nghiên cứu khác, nơi các đối tượng cần so sánh được mô tả dựa trên sự có mặt hoặc vắng mặt của nhiều ký tự.[15] *."*

[14] A subsequence of a sequence is a set of symbols drawn from the sequence in the same order as the original sequence. A subsequence is different from a substring in that the values of the subsequence need not be contiguous, whereas the values in the substring need to be contiguous.

[15] The similarity index suggested by Paul Jaccard 120 years ago has been one of the best-known coefficients in statistical ecology and other research fields in which the objects to be compared are described in terms of the presence or absence of many characters.

- Podani (2021) trang 1.

Theo nghiên cứu của Podani (2021), hệ số tương đồng Jaccard, được đặt theo tên nhà thực vật học người Thụy Sĩ Paul Jaccard. Hệ số này lần đầu tiên được giới thiệu trong bài báo năm 1901 bởi Jaccard (1901). Trong công trình này, Jaccard đã phát triển một hệ số, mà ông gọi là *hệ số cộng đồng*[16], nhằm định lượng mức độ tương đồng giữa các tập hợp loài thực vật trong các khu vực địa lý khác nhau. Mục tiêu chính của hệ số này là đánh giá mức độ chia sẻ loài giữa hai khu vực, từ đó hiểu rõ hơn về sự phân bố và đa dạng sinh học.

Kể từ đó, hệ số Jaccard đã được áp dụng rộng rãi trong nhiều lĩnh vực khoa học và kỹ thuật. Trong những năm gần đây, nhiều nghiên cứu về mảng xử lý ngôn ngữ tự nhiên, đã tận dụng hệ số này để giải quyết các vấn đề phức tạp trong đo lường độ tương đồng giữa các văn bản. Chẳng hạn, Diana and Ulfa (2019) sử dụng độ đo này để đo lường mức độ tương đồng giữa hai văn bản thông qua việc chuyển nội dung của chúng thành các tập hợp token (từ vựng) với các bộ q-grams khác nhau từ 1-grams, 2-grams cho đến 3-grams, sau đó tính toán tỷ lệ giữa phần giao và phần hợp của các tập này. Ngoài ra, nghiên cứu của Esteban et al. (2020) đánh giá hệ số Jaccard là thước đo độ tương đồng tốt nhất cho tiêu chí về giáo sư và năng lực trong hệ thống gợi ý lai đa tiêu chí.

Về ý tưởng chung, ta có hai tập hợp A và B. Khi đó, hệ số Jaccard được định nghĩa là tỷ lệ giữa số phần tử chung và tổng số phần tử phân biệt trong hai tập hợp:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}, \tag{3.3}$$

trong đó:

- $|A \cap B|$ là số lượng phần tử chung giữa hai tập,
- $|A \cup B|$ là tổng số phần tử phân biệt trong cả hai tập.

Giá trị của $J(A,B)$ nằm trong đoạn $[0,1]$. Cụ thể, giá trị $J(A,B)$ tiến dần về 1 khi số lượng phần tử chung giữa hai tập A và B chiếm tỷ lệ lớn hơn nhiều so với tổng phần tử khác biệt, và ngược lại khi giá trị $J(A,B)$ tiến dần về 0. Hay nói cách khác, $J(A,B) = 1$ khi $A \equiv B$ và $J(A,B) = 0$ khi $A \cap B = \emptyset$.

Trong bài toán về sử dụng độ đo tương đồng giữa hai chuỗi tên học phần, chúng tôi quyết định sử dụng đơn vị token là các ký tự trong một chuỗi. Giả sử, ta có hai chuỗi ký tự $\overline{X} = (x_1 x_2 \dots x_m)$, $\overline{Y} = (y_1 y_2 \dots y_n)$ và một số nguyên $q \geq 1$. Với m và n lần lượt là độ dài của chuỗi $\overline{X}, \overline{Y}$. Khi đó, hệ số Jaccard được định nghĩa là tỷ lệ giữa số phần tử chung và tổng số phần tử phân biệt trong hai chuỗi như sau:

$$J_q(\overline{X},\overline{Y}) = \frac{|Q_q(\overline{X}) \cap Q_q(\overline{Y})|}{|Q_q(\overline{X}) \cup Q_q(\overline{Y})|}, \tag{3.4}$$

trong đó, $Q_q(\cdot)$ tập hợp tất cả các chuỗi con liên tiếp có độ dài đúng bằng q trong chuỗi tương ứng:

[16] coefficient de communauté

$$Q_q(\overline{X}) = \{x_i x_{i+1} \dots x_{i+q-1} \,|\, 1 \leq i < m - q + 1\}\,. \tag{3.5}$$

Khi sử dụng hệ số Jaccard với q-gram, việc thay đổi giá trị q ảnh hưởng trực tiếp đến độ tương đồng giữa hai chuỗi. Giá trị q càng nhỏ thì sẽ giúp phát hiện sự tương đồng tổng quát. Từ đó, hệ số Jaccard giữa hai chuỗi có xu hướng tăng. Ngược lại, giá trị q càng lớn thì hệ số Jaccard giữa hai chuỗi có xu hướng giảm. Điều này sẽ giúp thuật toán so sánh một cách chính xác và phân biệt rõ hơn giữa hai chuỗi. Nghiên cứu của Diana and Ulfa (2019) cũng đã nhấn mạnh lại sự tác động trong việc lựa chọn q-gram, đồng thời chỉ ra rằng 2-gram mang lại chỉ số chính xác (precision) là 0.983, tốt hơn so với 1-gram và 3-gram trong tác vụ xác định các văn bản đạo văn.

3.2.3 Độ tương đồng Jaro-Winkler

> *"Đối với các thuộc tính chứa giá trị tên (chẳng hạn như tên riêng, họ, tên đường, tên địa danh hoặc tên bang/vùng lãnh thổ), các nghiên cứu đã chỉ ra rằng hàm so sánh chuỗi xấp xỉ Jaro-Winkler cho hiệu quả cao hơn so với các hàm so sánh khác.*[17] *"*
>
> *- Christen (2012) trang 139.*

Vào thập niên 80 đến 90, nước Mỹ bước vào thời kỳ số hóa dữ liệu dân số, các nhà nghiên cứu tại Cục Điều tra Dân số Hoa Kỳ[18] phải đối mặt với một vấn đề nan giải: *Làm sao để ghép nối chính xác hàng triệu bản ghi chứa tên người — trong khi sai lệch chính tả, ký tự bị đảo lộn hay nhập liệu thiếu sót là chuyện xảy ra như cơm bữa?*. Từ thực tiễn ấy, thuật toán Jaro đã ra đời bởi Jaro (1989) như một giải pháp đầu tiên, được thiết kế riêng cho việc so sánh tên gọi. Nó đánh giá độ tương đồng dựa trên số lượng ký tự trùng khớp và số lần hoán vị cần thiết để biến một chuỗi thành chuỗi kia – phản ánh sự gần gũi về mặt cấu trúc giữa hai cái tên.

Tuy nhiên, sau nhiều năm ứng dụng thực tế, các chuyên gia phát hiện một quy luật thú vị rằng con người có xu hướng nhận diện tên gọi chủ yếu dựa vào những ký tự đầu tiên. Những lỗi xảy ra ở phần cuối chuỗi thường ít ảnh hưởng hơn đến việc nhận diện so với lỗi ở phần đầu. Dựa trên kinh nghiệm này, thuật toán Jaro-Winkler được phát triển bởi Winkler (1990) như một cải tiến trực tiếp từ Jaro. Thuật toán mới bổ sung trọng số cho các ký tự trùng nhau ở đầu chuỗi, giúp tăng điểm tương đồng cho những cái tên có cùng tiền tố — từ đó cải thiện đáng kể độ chính xác trong các hệ thống ghép bản ghi, tìm kiếm mờ và xử lý dữ liệu thiếu chuẩn hóa.

Vào những năm gần đây, thuật toán Jaro-Winkler tiếp tục được nghiên cứu và cải tiến để nâng cao hiệu suất và độ chính xác. Trong nghiên cứu của Cherifi et al. (2011), là xây dựng mạng lưới dịch vụ web dựa trên ba phép đo tương đồng chuỗi là Levenshtein, Jaro và Jaro-Winkler, kết quả công bố cho thấy Jaro-Winkler tìm thấy nhiều tương đồng

[17] For attributes that contain name values (such as given names, surnames, street names, location/town names or state/territory names), it has been shown that the Jaro-Winkler approximate string comparison function performs better compared with other comparison functions, Yancey (2005)

[18] US Census Bureau

phù hợp hơn và có thể được sử dụng ở ngưỡng cao hơn. Không những vậy, nghiên cứu của Shareef et al. (2023) đã đề xuất phương pháp kết hợp giữa Jaro-Winkler và khoảng cách Manhattan để ưu tiên các trường hợp kiểm thử trong phát triển phần mềm, giúp cải thiện khả năng phát hiện lỗi.

Giả sử, ta có 2 chuỗi $\overline{X} = (x_1 x_2 \dots x_m)$ có độ dài là m và $\overline{Y} = (y_1 y_2 \dots y_n)$ có độ dài là n. Ta lần lượt thực hiện các bước tính toán như sau:

Bước 1: Tính độ tương đồng Jaro

Gọi M là số ký tự trùng nhau giữa $\overline{X}$ và $\overline{Y}$. Một cặp ký tự được coi là trùng nhau nếu chúng cùng xuất hiện trong chuỗi còn lại trong khoảng cách:

$$d = \left\lfloor \frac{\max(m, n)}{2} \right\rfloor - 1,$$

trong đó, $\lfloor x \rfloor$ là giá trị số nguyên lớn nhất và nhỏ hơn x. Ví dụ: x_i được xem là trùng với y_j nếu $j \in [\max(0, i-d),\ \min(i+d+1, n))$ và $x_i = y_j$.

Gọi T là một nửa số lượng các cặp ký tự trong M mà có thứ tự vị trí xuất hiện không giống nhau. Khi đó:

$$T = \frac{\text{số lượng ký tự trùng nhau mà không cùng thứ tự vị trí xuất hiện trong M}}{2}.$$

Khi đó, độ tương đồng Jaro được tính bằng công thức:

$$\text{Jaro}(X, Y) = \frac{1}{3}\left(\frac{M}{m} + \frac{M}{n} + \frac{M - T}{M}\right). \tag{3.6}$$

Bước 2: Tính độ tương đồng Jaro-Winkler

Gọi l là độ dài của chuỗi tiền tố chung giữa hai chuỗi X và Y (tối đa 4, tính từ ký tự đầu tiên). Gọi p là hệ số mở rộng (thường lấy $p = 0.1$), để kiểm soát mức độ ảnh hưởng của phần tiền tố giống nhau Khi đó, độ tương đồng Jaro-Winkler được tính như sau:

$$\text{Jaro-Winkler}(X, Y) = \text{Jaro}(X, Y) + l \cdot p \cdot (1 - \text{Jaro}(X, Y)). \tag{3.7}$$

Trình bày về lý do về giới hạn l, tác giả Winkler (1990) đã nhấn mạnh trong phần nghiên cứu của mình rằng các cặp tên thực tế thường chỉ giống nhau ở tối đa bốn ký tự đầu tiên. Dựa trên quan sát này, ông đề xuất giới hạn số ký tự tiền tố dùng để tăng điểm tương đồng trong công thức (3.7). Giới hạn này không chỉ phản ánh đặc trưng ngôn ngữ học của tên riêng, mà còn giúp tránh việc các chuỗi dài có tiền tố giống nhau chi phối quá mức kết quả so khớp. Đồng thời, thuật toán còn đảm bảo kết quả điểm tương đồng luôn nằm trong khoảng [0,1], càng gần về 1 thì hai chuỗi càng tương đồng nhau và ngược lại.

3.3 Các thuật toán và mô hình nhúng khả dụng trong RAG

Như đã đề cập chi tiết các bước trong quy trình tổng quát của mô hình RAG cơ bản (Hình 3.1) trong Phần 3.1, độc giả đã nắm được vai trò của các mô hình nhúng trong cả ba giai đoạn. Cụ thể: ở giai đoạn lập chỉ mục, mô hình nhúng đóng vai trò chuyển hóa các đoạn văn thành các biểu diễn vector. Tiếp đến là giai đoạn truy xuất, mô hình nhúng cũng đồng thời chuyển hóa câu truy vấn từ dạng văn bản sang biểu diễn vector, và thực hiện truy xuất dày đặc (Phần 3.1.1). Ngược lại, hệ thống RAG cơ bản có thể sử dụng truy xuất thưa bằng cách sử dụng thuật toán TF-IDF hoặc BM25 (Phần 3.1.1). Trong phần này, chúng tôi sẽ trình bày về hai thuật toán này, cũng như thông tin tổng quan về các mô hình nhúng hiện đại, được ứng dụng phổ biến.

3.3.1 Thuật toán BM25

Khi số lượng tài liệu và dữ liệu số ngày càng tăng nhanh chóng, việc tìm kiếm một văn bản phù hợp giữa hàng nghìn, thậm chí hàng triệu tài liệu trở thành một thách thức lớn. Bài toán truy hồi thông tin thường đặt ra câu hỏi quan trọng: *"Làm thế nào để xếp hạng các tài liệu sao cho những văn bản liên quan nhất đến truy vấn của người dùng được ưu tiên xuất hiện đầu tiên?"*. Một trong những lời giải đầu tiên đến từ thuật toán TF-IDF, nhưng chúng cũng bộc lộ những giới hạn khi phải xử lý các văn bản có độ dài khác nhau, hoặc khi cần xem xét mức độ liên quan phi tuyến giữa tần suất và độ quan trọng của từ (Phần 3.1.1). Chính trong bối cảnh đó, thuật toán BM25 ra đời bởi Robertson and Zaragoza (2009), là sự cải tiến của TF-IDF và thể hiện cho một triết lý: *"Tính liên quan không chỉ đến từ số lần xuất hiện, mà còn từ ngữ cảnh và đặc điểm tổng thể của cả tài liệu."*.

Thuật toán TF-IDF truyền thống

Để hiểu về BM25, chúng tôi sẽ bắt đầu từ TF-IDF. Trong hành trình phát triển của các mô hình truy hồi văn bản, thuật toán TF-IDF đã trở thành một công cụ kinh điển và cực kỳ phổ biến. Được đề xuất lần đầu tiên bởi Salton and Yu (1974), TF-IDF là một phương pháp thống kê nhằm lượng hóa tầm quan trọng của một từ khóa đối với một tài liệu cụ thể trong toàn bộ tập hợp văn bản.

Ý tưởng cốt lõi của TF-IDF rất tường minh: nếu một từ xuất hiện thường xuyên trong một tài liệu cụ thể nhưng lại hiếm khi xuất hiện ở các tài liệu còn lại, thì từ đó mang tính phân biệt cao, tức là có khả năng "đại diện" cho nội dung của tài liệu. Chính vì đặc tính này, TF-IDF đã trở thành nền tảng của nhiều hệ thống phân loại văn bản, truy vấn tìm kiếm và khai phá tri thức. TF-IDF bao gồm hai thành phần chính, bao gồm:

TF - Tần suất xuất hiện của từ

TF đại diện cho mức độ thường xuyên của một từ trong một tài liệu. Tần suất càng cao, từ đó càng có khả năng mang ý nghĩa quan trọng trong tài liệu đó.

$$\mathrm{TF}(t,d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}}, \tag{3.8}$$

trong đó:

- $f_{t,d}$ là số lần từ t xuất hiện trong tài liệu d,
- $\sum_{t' \in d} f_{t',d}$ là tổng số lần xuất hiện của các từ vựng trong tài liệu d.

IDF - Tần suất nghịch đảo theo tài liệu

Đo lường mức độ "phổ biến" của một từ trong toàn bộ tập hợp tài liệu. Một từ càng ít xuất hiện trong các tài liệu khác, thì IDF của nó càng cao, đồng nghĩa với việc từ đó càng mang tính đặc trưng.

$$\mathrm{IDF}(t) = \log\left(\frac{N}{n_t + 1}\right), \tag{3.9}$$

trong đó:

- N là tổng số tài liệu trong tập dữ liệu,
- n_t là tổng số tài liệu có chứa từ t.

Trong công thức (3.9), việc cộng thêm 1 trong mẫu số là để tránh phép chia cho 0 (trường hợp từ t không xuất hiện ở bất kỳ tài liệu nào). Khi kết hợp công thức (3.8) và (3.9), ta có công thức tổng quát của TF-IDF là:

$$\text{TF-IDF}(t,d) = \mathrm{TF}(t,d) \times \mathrm{IDF}(t). \tag{3.10}$$

Tính toán BM25

Như đã đề cập ở trên, thuật toán BM25 kế thừa trực tiếp tinh thần của TF-IDF, nhưng đi xa hơn bằng cách đưa vào hai yếu tố quan trọng là: mức độ bão hòa của tần suất từ và chuẩn hóa theo độ dài tài liệu. Điều này giúp BM25 không chỉ đánh giá tốt các từ khóa quan trọng, mà còn làm giảm tác động thiên lệch từ các văn bản quá ngắn hoặc quá dài.

Giả sử ta có một truy vấn q chứa nhiều từ, thuật toán BM25 tính điểm mức độ phù hợp giữa tài liệu d và truy vấn đó như sau:

$$\mathrm{BM25}(d,q) = \sum_{t \in q} IDF(t) \cdot \frac{f_{t,d} \cdot (k_1 + 1)}{f_{t,d} + k_1 \cdot (1 - b + b \cdot \frac{|d|}{\text{avgdl}})}, \tag{3.11}$$

trong đó:

- $f_{t,d}$ là số lần từ t xuất hiện trong tài liệu d,

- $|d|$ là tổng số từ trong tài liệu d,
- avgdl là độ dài trung bình của các tài liệu trong tập tài liệu,
- k_1 là ham số điều chỉnh độ bão hòa của tần suất từ – kiểm soát mức độ TF ảnh hưởng đến điểm tổng (thường trong khoảng [1.2, 2.0]),
- b là tham số điều chỉnh ảnh hưởng của độ dài tài liệu (thường là 0.75),
- IDF(t) là tần suất ngược của từ t, được tính dựa theo công thức cải tiến (3.12):

$$\text{IDF}(t) = \log\left(\frac{N - n_t + 0.5}{n_t + 0.5} + 1\right), \tag{3.12}$$

với N là tổng số tài liệu trong tập tài liệu và n_t là tổng số tài liệu chứa từ t.

Thay vì sử dụng một tỉ lệ đơn giản như trong TF-IDF truyền thống, BM25 điều chỉnh theo cơ chế bão hòa. Nghĩa là sau một ngưỡng nhất định, việc tăng tần suất của từ sẽ không làm tăng nhiều điểm số – phản ánh đúng hành vi ngôn ngữ tự nhiên. Hay nói một cách khác, văn bản dài có xu hướng chứa nhiều từ khóa chỉ vì nó dài hơn, không phải vì nội dung của nó có liên quan hơn. Thành phần $\frac{|d|}{\text{avgdl}}$ trong công thức giúp giảm thiểu hiện tượng này. Nếu tài liệu dài hơn mức trung bình, mẫu số sẽ tăng, làm giảm tác động của TF.

Ngoài ra, công thức IDF (3.12) của BM25 được cải tiến nhằm tránh các giá trị cực đoan khi n_t quá nhỏ hoặc bằng 0 (gây chia cho 0 hoặc log không xác định). Việc cộng thêm 0.5 vào tử và mẫu là một kỹ thuật làm mượt.

Trong bối cảnh các hệ thống RAG ngày càng được phát triển mạnh mẽ, BM25được xem như một phương pháp tiêu chuẩn để đánh giá khả năng truy xuất của hệ thống. Trong nghiên cứu của Jiang et al. (2023) với mô hình Active RAG, BM25 được sử dụng là phương pháp chuẩn để đánh giá hiệu quả của chiến lược truy xuất chủ động[19] mà nhóm tác giả đề xuất. Trong khi đó, khi phát triển mô hình ngôn ngữ In-context Retrieval-Augmented, các tác giả cũng sử dụng BM25 như một phương pháp truy xuất hiệu quả và tiết kiệm tài nguyên, giúp lựa chọn các tài liệu phù hợp để đưa vào lời nhắc đầu vào của LLM (Ram et al., 2023). Gần đây, nghiên cứu của Cheng et al. (2023) đã áp dụng BM25 để xây dựng bộ nhớ tri thức tự động cập nhật, nơi BM25 giúp xác định các đoạn văn bản liên quan trong quá trình tạo sinh câu trả lời có tính ngữ cảnh dài hạn.

3.3.2 Các mô hình nhúng phổ biến

Trong hệ thống RAG nói riêng và các biến thể nói chung, mô hình sinh chỉ là một nửa câu chuyện. Để truy xuất được những đoạn văn bản có liên quan đến truy vấn đầu vào, cần đến những mô hình nhúng có khả năng ánh xạ văn bản sang không gian vector sao cho các ngữ nghĩa tương đồng được đưa về gần nhau. Mỗi mô hình nhúng là một cách nhìn nhận khác nhau về ngôn ngữ, về cách biểu diễn nghĩa, và ảnh hưởng sâu sắc đến

[19] active retrieval

chất lượng của truy xuất. Trong mục này, chúng tôi sẽ trình bày một số mô hình nhúng phổ biến được sử dụng rộng rãi trong các hệ thống RAG hiện đại.

Contriever: được giới thiệu bởi nhóm nghiên cứu tại Facebook AI (hiện là Meta AI) Izacard et al. (2021). Thay vì phụ thuộc vào các cặp câu hỏi–đoạn văn được gán nhãn, Contriever tận dụng kỹ thuật học tương phản[20] với các cặp văn bản tự sinh từ bộ dữ liệu Wikipedia, cho phép huấn luyện trên quy mô lớn mà không cần nhãn. Về mặt kiến trúc, Contriever sử dụng bộ mã hóa dựa trên BERTs[21] với các bộ dữ liệu huấn luyện cho cả văn bản và truy vấn riêng biệt, và được huấn luyện với mục tiêu kéo các văn bản có ngữ nghĩa gần nhau lại gần nhau trong không gian vector, trong khi đẩy xa các cặp không liên quan. Nhờ kỹ thuật huấn luyện này, Contriever đạt hiệu suất ấn tượng trong bộ dữ liệu truy xuất chuẩn như Natural Questions (Kwiatkowski et al., 2019). Theo kết quả trong báo cáo của tác giả, Contriever đạt được Recall@20 lên đến 85.1% trên tập Natural Questions.

GritLM/GritLM-7B: Nếu Contriever là đại diện tiêu biểu cho hướng tiếp cận truy xuất dựa trên học tương phản không giám sát, thì GritLM lại mở ra một lối đi mới - khai thác trực tiếp một LLM để tảo ra các biểu diễn ngữ nghĩa mạnh mẽ, mà không cần đến một mô hình nhúng riêng biệt (ví dụ như BERTs). GritLM được phát triển bởi nhóm nghiên cứu tại Allen Institute for AI trong nghiên cứu Muennighoff et al. (2024). Trong số các phiên bản, GritLM-7B là mô hình với 7 tỷ tham số và 4096 chiều, được huấn luyện bằng phương pháp Tinh chỉnh hướng dẫn biểu diễn tạo sinh (Generative Representational Instruction Tuning - GRIT). Đây là một phương pháp sử dụng một LLM để thực hiện cả hai tác vụ là: tạo sinh ngôn ngữ và tạo các vector từ chính các tầng ẩn trung gian, thông qua câu chỉ thị. GritLM-7B được tinh chỉnh dựa trên LLM là Mistral 7B (Mistral AI, 2023) và sử dụng hai bộ dữ liệu tương ứng cho từng tác vụ, lần lượt là: Tülu 2 (Ivison et al., 2023) và E5S[22]. GritLM-7B đạt được các chỉ số vượt trội trên bộ Massive Text Embedding Benchmark (MTEB) (Muennighoff et al., 2022), với điểm trung bình tổng thể đạt 66.76, điểm trung bình cho RAG đạt 62.33, và điểm trung bình cho nhiệm vụ đo độ tương đồng ngữ nghĩa[23] đạt 83.35. Những kết quả này cho thấy GritLM-7B không chỉ vượt trội hơn tất cả các mô hình tạo sinh có cùng quy mô, mà còn sánh ngang hoặc vượt qua các mô hình lớn hơn đáng kể, chẳng hạn như LLaMA 2 70B (Meta AI, 2023), trong nhiều tác vụ nhúng văn bản và làm theo câu chỉ thị.

Nvidia/NV-Embed-v2: Với sự tham gia của các tập đoàn công nghệ lớn vào cuộc chơi, đặt mục tiêu tối ưu hóa hiệu suất trong toàn bộ chuỗi xử lý AI, NV-Embed-v2 là một trong những sản phẩm của NVIDIA (Lee et al., 2024). NV-Embed-v2 là một mô hình nhúng đa năng[24] dựa trên LLM, được thiết kế cho các tác vụ nhúng và truy xuất hiệu suất cao. Kiến trúc của NV-Embed-v2 giới thiệu một lớp attetion tiềm ẩn[25] cho các biểu diễn nhúng gộp [26], mang lại độ chính xác vượt trội so với các phương pháp gộp trung

[20] contrastive learning
[21] Bidirectional Encoder Representations from Transformers
[22] kết hợp giữa E5 (Wang et al., 2023) và S2ORC (Lo et al., 2019)
[23] Semantic Textual Similarity
[24] generalist embedding model
[25] latent attention layer
[26] pooled embeddings

bình hoặc token cuối[27] thông thường trong LLMs chỉ có bộ giải mã[28]. NV-Embed-v2 là mô hình với 7 tỷ tham số với số chiều có thể lên đến 4096, được huấn luyện bằng quy trình tinh chỉnh hướng dẫn tương phản[29]. Mô hình tận dụng các bộ dữ liệu được tuyển chọn[30] đa dạng, bao gồm các bộ dữ liệu đánh giá truy xuất công khai[31] để tối ưu hóa hiệu suất. Theo kết quả trong bài nghiên cứu của Lee et al. (2024), NV-Embed-v2 đã giành vị trí số 1 trên MTEB với điểm kỷ lục mới là 72.31 (tính đến ngày 30 tháng 8 năm 2024) trên 56 tác vụ nhúng; đạt điểm cao nhất trong phần Long Doc (74.78) vượt mặt cả mô hình text-embedding-3-large của OpenAI (68.77).

OpenAI/text-embedding-3-small, text-embedding-3-large: Tiếp tục với sự cạnh tranh giữa các tập đoàn công nghệ lớn, *text-embedding-ada-002* — mô hình nhúng thành công nhất trong năm 2022 của OpenAI đã được nâng cấp thanh hai phiên bản là: *text-embedding-3-small* và *text-embedding-3-large* nhằm cải thiện độ chính xác và hiệu quả (OpenAI, 2024). Hai mô hình mới thuộc dòng *text-embedding-3* được thiết kế cho các tác vụ như tìm kiếm ngữ nghĩa, phân cụm, và đề xuất. Theo thông tin được cung cấp trên trang web của OpenAI (2024), cả hai mô hình này đều được huấn luyện trên tập dữ liệu văn bản quy mô rất lớn, bao gồm cả dữ liệu đa ngôn ngữ, để phục vụ tốt hơn cho các tác vụ tìm kiếm, phân loại và so sánh văn bản. Về hiệu suất, text-embedding-3-large đạt điểm 64.6% trên bộ MTEB, vượt trội so với text-embedding-ada-002 (vốn chỉ đạt 61.0%). Trong khi đó, text-embedding-3-small dù nhẹ hơn nhưng vẫn đạt đến 60.5% trên MTEB, cho thấy hiệu quả tốt ngay cả với chi phí thấp. Đặc biệt, cả hai mô hình đều hỗ trợ tùy chỉnh chiều vector đầu ra (từ 256 đến 3072 chiều) và được thiết kế để cân bằng giữa hiệu năng và chi phí cho các ứng dụng thực tế.

3.3.3 Chuẩn L2 và độ tương đồng cosine

Trong cả giai đoạn lập chỉ mục và truy xuất thông tin của RAG, việc đo lường mức độ tương đồng giữa các vector đóng vai trò then chốt. Chúng giúp cho hệ thống nắm bắt được các sợi dây liên kết ngữ nghĩa giữa các đoạn văn, hoặc thực thể, hoặc bộ ba được trích xuất. Từ đó, hệ thống có thể cải thiện được khả năng suy luận đa bước một cách logic. Một trong những công cụ toán học phổ biến và hiệu quả được sử dụng là độ tương đồng cosine, được tính bằng cách nhân tích vô hướng của các vector được chuẩn hóa L2[32] (Aggarwal, 2015, trang 90).

Chuẩn L2

Chuẩn L2 (L2-norm), còn gọi là chuẩn Euclid, là một phép đo độ dài của vector trong không gian nhiều chiều. Trong không gian n chiều, giả sử ta có một vector $\mathbf{v} = (v_1, v_2, ..., v_n)$, chuẩn L2 cho vector $\mathbf{v}$ được định nghĩa như sau:

[27] last-token methods
[28] decoder-only LLMs
[29] contrastive instruction-tuning
[30] curated datasets
[31] public retrieval benchmarks
[32] L2-normalization

$$\|\mathbf{v}\|_2 = \sqrt{v_1^2 + v_2^2 + \cdots + v_n^2} = \sqrt{\sum_{i=1}^{n} v_i^2}. \tag{3.13}$$

Chuẩn hóa một vector theo chuẩn L2 là thao tác biến đổi vector đó về dạng đơn vị, tức là vector có độ lớn bằng 1 nhưng vẫn giữ nguyên hướng. Vector $\mathbf{v}$ sau khi chuẩn hóa sẽ có dạng:

$$\mathbf{v}_{norm} = \frac{\mathbf{v}}{\|\mathbf{v}\|_2}. \tag{3.14}$$

Việc chuẩn hóa này đặc biệt hữu ích trong các mô hình mà chỉ hướng của vector là quan trọng, còn độ lớn tuyệt đối thì không, chẳng hạn như khi tính độ tương đồng giữa các vector.

Độ tương đồng cosine giữa hai vector

Độ tương đồng cosine là một trong các độ đo phổ biến dùng để đánh giá mức độ tương đồng giữa hai vector trong không gian n chiều, dựa trên góc giữa chúng. Cho hai vector $\mathbf{a} = (a_1, a_2, ..., a_n), \mathbf{b} = (b_1, b_2, ..., b_n)$, độ tương đồng giữa chúng được tính bởi công thức:

$$\text{cosine\_similarity}(\mathbf{a}, \mathbf{b}) = \cos(\theta) = \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\|_2 \cdot \|\mathbf{b}\|_2} = \frac{\sum_{i=1}^{n} a_i b_i}{\sqrt{\sum_{i=1}^{n} a_i^2} \cdot \sqrt{\sum_{i=1}^{n} b_i^2}}. \tag{3.15}$$

Trong công thức (3.15), θ là góc giữa hai vector $\mathbf{a}, \mathbf{b}$ trong không gian n chiều. Bên cạnh đó, giá trị $\cos(\theta)$ nằm trong khoảng $[-1, 1]$, với:

- $\cos(\theta) = 1$ khi $\theta = 0°$ (hai vector đồng hướng),
- $\cos(\theta) = 0$ khi $\theta = 90°$ (hai vector vuông góc),
- $\cos(\theta) = -1$ khi $\theta = 180°$ (hai vector ngược hướng).

Khi các vector đã được chuẩn hóa bằng chuẩn L2 (tức là $\|\mathbf{a}\|_2 = \|\mathbf{b}\|_2 = 1$), công thức (3.15) được đơn giản hóa thành:

$$\cos(\theta) = \mathbf{a} \cdot \mathbf{b} = \sum_{i=1}^{n} a_i b_i. \tag{3.16}$$

Như vậy, công thức (3.16) cho thấy ta có thể tính độ tương đồng cosine bằng cách chuẩn hóa vector và sau đó nhân tích vô hướng giữa chúng.

3.4 Thuật toán Personalized PageRank

> *"Thuật toán Topic-sensitive PageRank tính toán nhiều vector PageRank, mỗi vector được thiên lệch theo một chủ đề cụ thể, từ đó cho phép xếp hạng kết quả truy vấn phù hợp hơn với sở thích chủ đề của người dùng.*[33] *"*
>
> *- Haveliwala (2002).*

Giả sử một sinh viên đang sử dụng hệ thống tư vấn hỏi-đáp học phần H-GraphRAG để tìm kiếm học phần phù hợp cho kỳ tới, với truy vấn: *"Which elective courses teach about time series?"*. Khi này, hệ thống không những phải tìm ra được chính xác các thông tin liên quan đến *"time series"*, mà còn phải xếp hạng các thông tin đó dựa trên ngữ cảnh sự liên kết về các học phần tiên quyết. Điều này đòi hỏi một cơ chế truy xuất không đơn thuần dựa vào sự tương đồng văn bản, mà phải tận dụng được các mối quan hệ trong đồ thị học phần - với hai loại nút lần lượt là nút thực thể (các danh từ hoặc cụm danh từ) và nút đoạn văn, cùng với các cạnh biểu diễn mối liên hệ giữa chúng (Hình 3.2). Trong bối cảnh này, thuật toán Personalized PageRank (PPR) đóng vai trò tính toán trọng số cho tất cả các nút trong đồ thị. Các nút có trọng số càng cao tức càng liên quan đến câu truy vấn, và ngược lại. Từ đó, hệ thống có thể dễ dàng xếp hạng các nút tiềm năng theo độ liên quan ngữ nghĩa và cấu trúc, bằng cách chọn ra top-k nút có trọng số cao nhất.

Thuật toán PageRank ban đầu, được phát triển bởi Brin and Page (1998), mô hình hóa quá trình một người dùng "du hành ngẫu nhiên" trong mạng lưới web bằng cách nhấp vào các liên kết trong một trang web. Thuật toán giả định rằng một trang web càng được nhiều trang khác trỏ tới, thì nó càng quan trọng. Mô hình này ví người dùng như một "khách du lịch ngẫu nhiên" - họ lướt qua các trang web bằng cách nhấp vào liên kết, và đôi khi họ sẽ "nhảy" sang một trang khác bất kỳ mà không theo liên kết nào cả. Tuy nhiên, cách tiếp cận này không phân biệt người dùng, không ưu tiên theo chủ đề hoặc nhu cầu cụ thể.

Chính từ khoảng trống đó, thuật toán Topic-Sensitive PageRank, hay còn gọi là Personalized PageRank (PPR), được nghiên cứu bởi Haveliwala (2002), mục tiêu là tùy biến kết quả xếp hạng theo một tập ưu tiên[34] - theo chủ đề, sở thích người dùng, hoặc truy vấn cụ thể. Hay nói một cách khác, mỗi lần người dùng "nhảy" ngẫu nhiên[35], người dùng sẽ có xu hướng quay trở về các trang web có liên quan đến truy vấn gốc với xác suất cao hơn.

Hệ thống chúng tôi đề xuất sử dụng (Phần 4.2) có ứng dụng thuật toán PPR trong giai đoạn truy xuất trực tuyến trên đồ thị tri thức, bắt đầu bằng các nút "hạt giống" (Hình 4.5). Cách tiếp cận này cho phép hệ thống không chỉ tìm chính xác các thông tin thực thể và đoạn văn liên quan đến câu truy vấn, mà còn ưu tiên các thông tin đóng vai trò quan trọng theo ngữ cảnh cụ thể.

[33] Topic-sensitive PageRank computes multiple PageRank vectors, each biased towards a particular topic, enabling more relevant query-time rankings based on the user's topic preference.

[34] personalization set

[35] teleport

3.4.1 Thuật toán PageRank

Như đã đề cập, thuật toán PageRank mô hình hóa hành vi lan truyền trọng số cho các nút trong đồ thị. Cụ thể, thuật toán sẽ bắt đầu bằng các nút "hạt giống", di chuyển theo liên kết giữa các nút khác (với xác suất α) hoặc nhảy đến một nút bất kỳ (với xác suất $1-\alpha$). Quá trình này được mô hình hóa bằng công thức sau:

$$\mathbf{r}^{k+1} = \alpha \cdot P^T \mathbf{r}^k + (1-\alpha) \cdot \mathbf{v}, \tag{3.17}$$

trong đó:

- $\mathbf{r}^{k+1} \in \mathbb{R}^n$: vector PageRank tại vòng lặp thứ $(k+1)$. Trong đó, mỗi phần tử r_i biểu diễn mức độ quan trọng của nút i sau khi thuật toán hội tụ.
- $P \in \mathbb{R}^{n \times n}$: ma trận xác suất chuyển tiếp[36] trong đồ thị có hướng.
- $\alpha \in [0, 1]$: hệ số tắt dần[37].
- $\mathbf{v} \in \mathbb{R}^n$: vector "nhảy", biểu diễn xác suất nhảy đến từng nút trong trường hợp không đi theo liên kết.

Công thức (3.17) là một phương trình đệ quy. Trong đó, ma trận chuyển tiếp P với kích thước $(n \times n)$ với n là tổng số nút có trong đồ thị. Bên cạnh đó, nếu có liên kết từ nút i đến nút j thì giá trị $P_{ij} = 1/n_i$, với n_i là tổng số liên kết đi ra từ nút i. Bên cạnh đó, ở vòng lặp đầu tiên $(k = 0)$, vector $\mathbf{r}^0$ sẽ được khởi tạo với phân phối đồng đều:

$$\mathbf{r}^0 = [\frac{1}{n}]_{n \times 1}.$$

Ngoài ra, $\mathbf{v}$ cũng là vector đồng nhất, tức tất cả các nút có xác suất bằng nhau và cố định ở tất cả các vòng lặp $\mathbf{v} = [\frac{1}{n}]_{n \times 1}$. Nhưng trong PPR, đây chính là yếu tố dẫn tới sự "ưu tiên" cho các nút thật sự quan trọng.

Hệ số tắt dần

Một trong những yếu tố tưởng chừng nhỏ bé nhưng lại có ảnh hưởng sâu sắc đến chất lượng kết quả trong PageRank và cả PPR chính là hệ số tắt dần α. Việc lựa chọn giá trị α không chỉ là một thao tác kỹ thuật, mà thực chất là một quyết định mang tính định hướng chiến lược. Nếu α được đặt quá cao (tiền dần về 1), thuật toán chủ yếu lan truyền một cách "lang thang" theo các liên kết trong đồ thị. Ngược lại, khi α tiến dần về 0, trọng số được phân phối gần như ngẫu nhiên theo vector nhảy $\mathbf{v}$, mặc dù có thể khiến cho kết quả truy xuất các nút mang ý nghĩa gần hơn với câu truy vấn, nhưng sẽ làm hạn chế khả năng khám phá diện rộng trong đồ thị.

[36] transition probability matrix
[37] damping factor

3.4.2 Tính toán vector PPR

Ý tưởng cốt lõi của PPR là: thay vì lan truyền trọng số đồng đều cho tất cả các nút trong đồ thị như PageRank, thuật toán PPR chỉ khởi đầu việc lan truyền từ một nhóm nút ưu tiên — tức những nút có liên quan trực tiếp đến câu truy vấn về một chủ đề cụ thể.

Giả sử ta có một truy vấn đề cập đến chủ đề t (ví dụ như: "time series"), ta xác định một tập S_t gồm các nút thực thể và nút đoạn văn liên quan đến chủ đề t. Từ đó, vector nhảy $\mathbf{v}$ được khởi tạo như sau:

$$v_i = \begin{cases} \frac{1}{|S_t|} & \text{nếu } i \in S_t, \\ 0 & \text{ngược lại} \end{cases} \tag{3.18}$$

Sau đó, thuật toán sẽ được lặp theo công thức (3.17). Qúa trình này được lặp cho đến khi $\mathbf{r}$ hội tụ, tức là sự thay đổi giữa hai vòng lặp nhỏ hơn một ngưỡng ϵ. Kết quả thu được là một vector phân phối $\mathbf{r}$ với các trọng số theo chủ đề t, cho phép xếp hạng các nút thực thể và đoạn liên quan đến truy vấn một cách ngữ cảnh hóa.

3.5 Kỹ thuật ra chỉ thị

Sự ra đời và phát triển mạnh mẽ của LLMs như GPT-3, GPT-4 đã mở ra một kỷ nguyên mới cho trí tuệ nhân tạo tạo sinh[38]. Tuy nhiên, cùng với sức mạnh ngày càng gia tăng của các mô hình này, một vấn đề mới cũng dần trở nên rõ nét: *"Làm thế nào để con người có thể giao tiếp hiệu quả với AI nhằm đạt được kết quả như mong đợi?"*

Từ nhu cầu đó, kỹ thuật ra chỉ thị đã xuất hiện như một cầu nối thiết yếu giữa con người và mô hình ngôn ngữ. Không còn là những lệnh rời rạc, chỉ thị giờ đây được xem như những hướng dẫn có cấu trúc, được thiết kế cẩn thận để truyền đạt rõ ràng ý định của người dùng đến AI. Nhờ vào đó, AI không chỉ "hiểu đúng" nhiệm vụ được giao mà còn có thể thực hiện một cách chính xác và nhất quán hơn.

Kỹ thuật ra chỉ thị đóng vai trò trung tâm trong việc định hình hành vi của LLMs (Izacard et al., 2023, Han et al., 2024, Brown et al., 2020). Một chỉ thị tốt có thể:

- Giúp AI hiểu đúng và hành động đúng theo yêu cầu nhờ cách diễn đạt rõ ràng, cụ thể và có tính định hướng cao;
- Tăng mức độ kiểm soát đầu ra, hạn chế các phản hồi sai lệch, vô nghĩa hoặc không phù hợp với ngữ cảnh;
- Cải thiện hiệu suất và trải nghiệm người dùng, khi AI phản hồi nhanh hơn, chính xác hơn, và nhất quán hơn trong các lần tương tác;
- Làm nền tảng để phát triển các ứng dụng AI thông minh, linh hoạt và dễ mở rộng – từ trợ lý ảo cá nhân, hệ thống hỏi đáp, cho đến các công cụ hỗ trợ sáng tạo nội dung.

[38] generative AI

Khi nói đến kỹ thuật ra chỉ thị, hai phương pháp nền tảng là *zero-shot learning* và *few-shot learning* thường được xem như những viên gạch đầu tiên để khai thác sức mạnh của LLMs. Mỗi phương pháp đại diện cho một cách tiếp cận khác nhau trong việc hướng dẫn AI thực hiện nhiệm vụ, với những ưu điểm và phạm vi ứng dụng riêng biệt.

3.5.1 Zero-shot Learning

> *"Học không mẫu hoạt động bằng cách cung cấp cho mô hình một mô tả nhiệm vụ dưới dạng ngôn ngữ tự nhiên thay vì bất kỳ ví dụ nào; mô hình sau đó được kỳ vọng sẽ tự động sinh ra phần hoàn chỉnh phù hợp với yêu cầu.*[39] "
>
> - *Brown et al. (2020).*

Zero-shot learning (Học không mẫu) là minh chứng điển hình cho khả năng "tự suy luận" của mô hình AI. Trong cách tiếp cận này, người dùng không cần cung cấp bất kỳ ví dụ cụ thể nào. Thay vào đó, họ chỉ cần mô tả nhiệm vụ một cách rõ ràng, súc tích – chẳng hạn như: "Dịch câu sau sang tiếng Anh" hoặc "Tóm tắt đoạn văn sau". Mô hình sau đó sẽ dựa hoàn toàn vào vốn kiến thức đã được huấn luyện từ trước để đưa ra phản hồi phù hợp. Điều này không khác gì việc yêu cầu một người đọc hiểu đề bài và tự giải quyết mà không cần nhìn ví dụ minh họa. Một số đặc điểm nổi bật của zero-shot learning gồm:

- **Không cần ví dụ trong câu lệnh chỉ thị:** Mô hình được giao nhiệm vụ trực tiếp, không có mẫu tham khảo.
- **Dựa vào khả năng tổng quát hóa mạnh mẽ:** AI phải vận dụng những gì đã học trong quá trình huấn luyện để tự suy luận và giải quyết vấn đề.
- **Tiết kiệm thời gian và công sức:** Người dùng không cần soạn thảo các ví dụ thủ công, điều này đặc biệt hữu ích khi triển khai hàng loạt hoặc xử lý các tác vụ phổ biến.
- **Phù hợp với các nhiệm vụ quen thuộc:** Zero-shot thường hoạt động tốt với những bài toán phổ thông – chẳng hạn như phân loại cảm xúc, trả lời câu hỏi thông thường, hay dịch ngôn ngữ, nơi mô hình đã từng tiếp xúc nhiều lần trong giai đoạn huấn luyện.

Tuy nhiên, khi nhiệm vụ trở nên phức tạp hơn hoặc có tính chuyên biệt cao, việc bổ sung ví dụ minh họa – tức chuyển sang few-shot learning, có thể là lựa chọn phù hợp hơn. Phần tiếp theo sẽ trình bày rõ hơn về phương pháp này và cách nó mở rộng khả năng của mô hình trong các tình huống khó.

[39] Zero-shot learning works by giving a natural language description of the task instead of any examples, with the model expected to provide the completion

3.5.2 Few-shot Learning

> *"Học vài mẫu, tức là bài toán học từ một số lượng rất nhỏ ví dụ, đã thu hút sự quan tâm mạnh mẽ trong lĩnh vực xử lý ngôn ngữ tự nhiên trong thời gian gần đây, đặc biệt nhờ sự xuất hiện của các mô hình tiền huấn luyện quy mô lớn – vốn thể hiện năng lực học vài mẫu một cách nổi bật và tự phát.*[40] "
>
> - *Izacard and Grave (2021).*

Trái ngược với Zero-shot , nơi mô hình phải tự hình dung cách làm chỉ từ lời mô tả thì Few-shot learning (Học vài mẫu) đưa ra một cách tiếp cận giàu tính định hướng hơn. Trong kỹ thuật này, người dùng cung cấp cho mô hình một số ví dụ minh họa (thường là từ 1 đến 5 ví dụ) trong phần chỉ thị, nhằm giúp mô hình hiểu rõ hơn về kỳ vọng của nhiệm vụ.

Hãy hình dung một giáo viên không chỉ giao đề bài mà còn đưa cho học sinh vài bài làm mẫu. Nhờ những mẫu này, học sinh có thể nhanh chóng nắm bắt cách trình bày, cách suy luận và phong cách trả lời mong đợi. Mô hình AI trong few-shot learning cũng hoạt động tương tự như vậy: nó học từ ví dụ để mô phỏng và tổng quát hóa cách thực hiện nhiệm vụ mới.

Một số đặc điểm chính của few-shot learning gồm:

- **Cung cấp ví dụ minh họa trong câu lệnh chỉ thị:** Những ví dụ này có thể là các cặp câu hỏi–trả lời, đầu vào–đầu ra, hoặc bất kỳ dạng thể hiện nào giúp mô hình hiểu được cấu trúc và nội dung cần tạo.
- **Học từ ngữ cảnh cụ thể:** Mô hình sử dụng các ví dụ như một dạng "bộ nhớ tạm thời" để suy luận ra quy luật và áp dụng vào trường hợp mới.
- **Tăng độ chính xác trong các tác vụ phức tạp:** Với các nhiệm vụ chưa quen thuộc, có ngữ cảnh chuyên biệt hoặc nhiều biến thể, việc có ví dụ đóng vai trò quan trọng trong việc giảm thiểu sai lệch và tăng tính phù hợp của đầu ra.
- **Đánh đổi giữa chất lượng và chi phí:** Dù few-shot thường cho kết quả tốt hơn zero-shot trong nhiều trường hợp, nhưng nó yêu cầu người dùng đầu tư công sức để soạn ví dụ phù hợp, cũng như chấp nhận độ dài prompt dài hơn, ảnh hưởng đến chi phí xử lý.

3.6 Các phương pháp đánh giá ứng dụng RAG

Sau khi độc giả đã nắm bắt được quy trình hoạt động tổng quát của một hệ thống RAG, cũng như các phương pháp được ứng dụng trong từng thành phần chính. Trong phần này, chúng tôi sẽ trình bày về ba phương pháp đánh giá phổ biến trong ứng dụng RAG,

[40] Few-shot learning, the task of learning from very few examples, has recently seen an explosion of interest in NLP with the arrival of large pre-trained models, which exhibit emergent few-shot learning abilities

lần lượt là: (i) Đánh giá hiệu suất truy xuất thông tin, (ii) Đánh giá hiệu suất của tạo sinh ngôn ngữ và (iii) Đánh giá dựa trên AI (Gutiérrez et al., 2024, Gutiérrez et al., 2025, Sarthi et al., 2024, Sawarkar et al., 2024, Ram et al., 2023). Phương pháp (ii) thường được chia ra thành hai mảng là: Đánh giá dựa trên nhiệm vụ đóng và Đánh giá dựa trên tham chiếu. Ở từng phương pháp (i) và (ii), chúng tôi cũng sẽ trình bày về các chỉ số định lượng liên quan được ứng dụng.

3.6.1 Đánh giá hiệu suất truy xuất thông tin

Một trong cã thành phần quan trọng của một hệ thống RAG là mô-đun truy xuất (Phần 3.1). Kết quả của mô-đun này sẽ quyết định đến câu trả lời của hệ thống có chính xác, đầy đủ và liên quan đến câu truy vấn hay không. Do đó, việc đánh giá hiệu xuất truy xuất là vô cùng cần thiết và một trong những chỉ số định lượng được sử dụng phổ biến chính là Recall@top-k.

Recall@top-k

Recall@top-k, hay độ phủ tại k vị trí hàng đầu, là một độ đo phổ biến để đánh giá khả năng của một hệ thống truy xuất thông tin trong việc tìm thấy tất cả các tài liệu liên quan (so với câu truy vấn) trong số k tài liệu hàng đầu mà nó trả về (Gutiérrez et al., 2024, Gutiérrez et al., 2025). Để tính Recall@top-k, ta cần:

- Một truy vấn Q.
- Một tập hợp các tài liệu liên quan thực sự cho truy vấn Q, ký hiệu là R_Q.
- Một danh sách các tài liệu được hệ thống truy xuất trả về cho truy vấn Q, được sắp xếp theo mức độ liên quan giảm dần và lấy ra k số tài liệu hàng đầu, ký hiệu là R_k.

Công thức tính Recall@top-k được định nghĩa như sau:

$$\text{Recall@top k} = \frac{\text{Số tài liệu liên quan trong top } k \text{ kết quả}}{\text{Tổng số tài liệu liên quan cho truy vấn}} = \frac{|R_k \cap R_Q|}{|R_Q|}. \tag{3.19}$$

Ví dụ, nếu một truy vấn có 5 tài liệu liên quan và hệ thống trả về 10 tài liệu, trong đó có 3 tài liệu liên quan nằm trong 5 vị trí hàng đầu (top 5), thì Recall@top 5 sẽ là $3/5 = 0.6$.

3.6.2 Đánh giá hiệu suất tạo sinh ngôn ngữ dựa trên nhiệm vụ đóng

Đánh giá dựa trên nhiệm vụ đóng là một phương pháp đánh giá dựa trên bộ dữ liệu kiểm tra, với các mẫu được chuẩn bị dưới dạng câu hỏi ngắn hoặc trắc nghiệm[41]. Hai độ đo

[41] multiple-choice

chính và phổ biến được sử dụng trong phương pháp này là: So khớp chính xác (Exact Match - EM) và điểm F1 (F1-score). Tại đây, các chuỗi câu trả lời được chuẩn hóa bằng cách chuyển sang chữ thường, loại bỏ các mạo từ và dấu câu, sau đó được mã hóa thành các từ dựa trên khoảng trắng. Với một câu trả lời dự đoán từ LLM là $\hat{a}$ và một câu trả lời tham chiếu a, bao gồm các token (chúng tôi sử dụng đơn vị token ở đây là từ vựng. Mỗi từ được cách nhau bởi khoảng trắng) $(\hat{a}_1\hat{a}_2 \dots \hat{a}_m)$ và $(a_1a_2 \dots a_n)$.

Exact Match

Điểm khớp chính xác EM được định nghĩa như sau:

$$\text{EM}(\hat{a}, a) = \begin{cases} 1 & \text{nếu } \hat{a} = a \\ 0 & \text{ngược lại} \end{cases}. \tag{3.20}$$

Có thể nói, EM là một độ đo nghiêm ngặt, vì câu trả lời dự đoán từ LLM phải khớp hoàn toàn các token với câu trả lời tham chiếu (bao gồm cả việc độ dài của hai chuỗi cũng phải bằng nhau: $|\hat{a}| = |a|$) thì mới được tính điểm.. Ví dụ, câu trả lời dự đoán là "King Charles III" sẽ không nhận được điểm nào cho câu trả lời tham chiếu là "Charles I". Do đó, một độ đo mềm hơn cũng thường được sử dụng, đó là F1-score.

F1-score

Cũng với câu trả lời dự đoán là $\hat{a} = (\hat{a}_1, \dots, \hat{a}_m)$ và câu trả lời tham chiếu $a = (a_1, \dots, a_n)$. Để tính được điểm F1, chúng ta cần phải tính độ chính xác (Precision) và độ bao phủ (recall). Trong đó, Precision được định nghĩa là tỷ lệ giữa số lượng token trong câu dự đoán mà khớp chính xác với câu tham chiếu, trên tổng số lượng token có trong câu dự đoán:

$$\text{Precision} = \frac{|\{\hat{a}_1\hat{a}_2 \dots \hat{a}_m\} \cap \{a_1a_2 \dots a_n\}|}{|\{\hat{a}_1\hat{a}_2 \dots \hat{a}_n\}|}. \tag{3.21}$$

Recall được định nghĩa là tỷ lệ giữa số lượng token trong câu dự đoán mà khớp chính xác với câu tham chiếu, trên tổng số lượng token có trong câu tham chiếu, tức là:

$$\text{Recall} = \frac{|\{\hat{a}_1\hat{a}_2 \dots \hat{a}_m\} \cap \{a_1, a_2 \dots a_n\}|}{|\{a_1a_2 \dots a_n\}|}. \tag{3.22}$$

Từ đó, chỉ số F1 sẽ là trung bình điều hòa giữa Precision và Recall, giúp cân bằng hai yếu tố chính xác và bao phủ:

$$\text{F1} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{3.23}$$

3.6.3 Đánh giá hiệu suất tạo sinh ngôn ngữ dựa trên tham chiếu

Đánh giá hiệu suất tạo sinh ngôn ngữ dựa trên tham chiếu là phương pháp so sánh kết quả trả lời từ hệ thống với một hoặc nhiều câu trả lời chuẩn (thường được soạn từ con người) để xác định mức độ tương đồng cả về mặt diễn đạt lẫn thông tin của câu trả lời. Các chỉ số phổ biến được sử dụng bao gồm: BLEU, ROUGE và METEOR

BLEU

.

> *"Làm thế nào để đo lường chất lượng bản dịch máy? Một bản dịch máy được xem là tốt hơn khi nó càng tiệm cận với bản dịch của người chuyên nghiệp. Đây chính là ý tưởng cốt lõi trong đề xuất của chúng tôi. Để đánh giá chất lượng của một bản dịch máy, người ta đo mức độ tương đồng của nó so với một hoặc nhiều bản dịch tham chiếu do con người thực hiện, dựa trên một thước đo định lượng cụ thể.*[42] "
>
> - *Papineni et al. (2002)*

Cuối những năm 1990 và đầu 2000, dịch máy bùng nổ nhưng vẫn thiếu phương pháp đánh giá nhanh gọn mà đáng tin cậy. Đến năm 2002, Papineni và các cộng sự tại IBM giới thiệu Bilingual Evaluation Understudy (BLEU), là một độ đo phổ biến để đánh giá chất lượng của văn bản được sinh ra bằng cách so sánh nó với một hoặc nhiều bản dịch tham chiếu do con người tạo ra (Papineni et al., 2002). Ý tưởng chính của BLEU là định lượng mức độ trùng khớp của các n-gram (chuỗi n tokens liên tiếp) giữa văn bản dự đoán và văn bản tham chiếu. Chẳng hạn như chúng ta có câu "I love NLP", với 2-gram = ["I love", "love NLP"]. Điểm BLEU càng cao cho thấy văn bản dự đoán càng gần với văn bản tham chiếu. Có càng nhiều n-gram khớp, điểm BLEU càng cao.

Công thức tổng quát của BLEU bao gồm tích hình học của độ chính xác n-gram cho các n khác nhau (thường từ $n = 1$ đến $n = 4$) và một yếu tố phạt độ ngắn[43] để tránh việc các bản dịch quá ngắn có điểm cao một cách không chính xác:

$$\text{BLEU} = \text{BP} \cdot \exp\left(\sum_{n=1}^{N} w_n \log P_n\right). \tag{3.24}$$

Trong đó:

- P_n là độ chính xác n-gram đã được cắt[44],

[42] How does one measure translation performance? The closer a machine translation is to a professional human translation, the better it is. This is the central idea behind our proposal. To judge the quality of a machine translation, one measures its closeness to one or more reference human translations according to a numerical metric.

[43] brevity penalty

[44] clipped precision

- w_n là trọng số cho độ chính xác n-gram (thường là $1/N$). Với N là số bậc tối đa của n-gram (Ví dụ: nếu $N = 4$ thì sẽ xét từ 1-gram đến 4-gram),
- BP là yếu tố phạt độ ngắn, được tính để giảm điểm nếu bản dịch dự đoán ngắn hơn bản dịch tham chiếu.

Độ chính xác P_n đo lường tỷ lệ các n-gram trong câu trả lời dự đoán $\hat{a}$ xuất hiện trong câu tham chiếu a, nhưng không vượt quá số lần xuất hiện tối đa trong câu tham chiếu. Hay nói một cách khác, nếu một n-gram được lặp nhiều lần trong câu dự đoán mà chỉ xuất hiện một lần trong tham chiếu, ta chỉ tính tối đa một lần để tránh đánh giá không chính xác:

$$P_n = \frac{\sum_{ngram \in \hat{a}} min(count_{\hat{a}}(ngram), count_a(ngram))}{\sum_{ngram \in \hat{a}} count_{\hat{a}}(ngram)}. \tag{3.25}$$

Trong đó:

- $count_{\hat{a}}(ngram)$ là số lần xuất hiện của n-gram trong câu dự đoán.
- $count_a(ngram)$ là số lần xuất hiện của n-gram trong câu tham chiếu.

Ngoài ra, vì BLEU không thưởng cho việc tạo sinh câu trả lời dự đoán quá ngắn nên đưa vào hệ số phạt BP như sau:

$$BP = \begin{cases} 1 & \text{nếu } |\hat{a}| > |a|, \\ exp\left(1 - \frac{|a|}{|\hat{a}|}\right) & \text{nếu } |\hat{a}| \leq |a|. \end{cases} \tag{3.26}$$

ROUGE

> *"Bộ chỉ số ROUGE bao gồm các phương pháp đánh giá tự động chất lượng của một bản tóm tắt bằng cách so sánh nó với một hoặc nhiều bản tóm tắt lý tưởng do con người tạo ra.*[45] *"*
>
> - *Lin (2004).*

Phương pháp Recall-Oriented Understudy for Gisting Evaluation (ROUGE), được đề xuất bởi Lin (2004), sử dụng để đánh giá hiệu suất của các tác vụ dịch máy và tóm tắt văn bản, với trọng tâm là đo lường độ phủ, tức là xác định bao nhiêu n-gram trong bản tham chiếu xuất hiện trong văn bản dự đoán được tạo ra. ROUGE được thiết kế nhằm khắc phục một số hạn chế của BLEU. Cụ thể, ROUGE nhấn mạnh hơn vào yếu tố độ phủ so với BLEU, đồng thời có khả năng phản ánh tốt hơn ý nghĩa nội dung của văn bản dự đoán. Trong đó, ROUGE-L đặc biệt dựa trên LCS giữa văn bản dự đoán và văn bản tham chiếu. Như đã đề cập ở Phần 3.2.1, LCS là độ dài của dãy con chung dài nhất

[45] ROUGE includes measures to automatically determine the quality of a summary by comparing it to other (ideal) summaries created by humans

trong cả hai chuỗi đầu vào. ROUGE-L được tính dựa trên độ chính xác (Precision), độ phủ (Recall) và điểm F1 của LCS như sau:

$$\text{Precision}_{LCS} = \frac{LCS(\hat{a}, a)}{|\hat{a}|}. \tag{3.27}$$

Độ chính xác Precision_{LCS} đo lường tỷ lệ độ dài của dãy con chung dài nhất giữa câu dự đoán và câu tham chiếu, trên độ dài của câu dự đoán. Độ phủ Recall_{LCS} đánh giá tỷ lệ độ dài của dãy con chung dài nhất giữa câu dự đoán và câu tham chiếu, trên độ dài của câu tham chiếu:

$$\text{Recall}_{LCS} = \frac{LCS(\hat{a}, a)}{|a|}. \tag{3.28}$$

Cuối cùng, chỉ số F1_{LCS} là trung bình điều hòa giữa độ chính xác và độ phủ LCS nhằm cung cấp một thước đo cân bằng giữa tính chính xác và tính bao phủ của hai chuỗi văn bản dự đoán và tham chiếu:

$$\text{F1}_{LCS} = 2 \cdot \frac{\text{Precision}_{LCS} \cdot \text{Recall}_{LCS}}{\text{Precision}_{LCS} + \text{Recall}_{LCS}}. \tag{3.29}$$

METEOR

> *"Chúng tôi giới thiệu METEOR, một chỉ số đánh giá tự động dành cho bài toán dịch máy, dựa trên khái niệm tổng quát về việc so khớp đơn vị từ đơn giữa bản dịch do máy tạo và các bản dịch tham chiếu do con người thực hiện. Các từ đơn này có thể được so khớp dựa trên hình thức bề mặt, gốc của từ, hoặc ý nghĩa.* [46] *"*
>
> - *Banerjee and Lavie (2005).*

Thực tế cho thấy BLEU và ROUGE đều mắc hạn chế khi gặp các biến thể ngôn ngữ: đồng nghĩa, biến đổi từ gốc, hoặc trật tự thay đổi. Năm 2005, phương pháp Metric for Evaluation of Translation with Explicit ORdering (METEOR) ra đời và được xem như là một độ đo được thiết kế để khắc phục một số hạn chế của BLEU và ROUGE, đặc biệt là việc nó tính đến các yếu tố ngữ nghĩa như từ đồng nghĩa và biến thể hình thái, cũng như trật tự từ (Banerjee and Lavie, 2005). METEOR được tính toán dựa trên trung bình điều hòa của độ chính xác và độ phủ của các đơn vị n-gram, sau đó được nhân với một yếu tố phạt cho trật tự từ không đúng:

$$\text{METEOR} = \text{Fmean} \cdot (1 - \text{Penalty}). \tag{3.30}$$

Trong đó:

[46] We describe METEOR, an automatic metric for machine translation evaluation that is based on a generalized concept of unigram matching between the machineproduced translation and human-produced reference translations. Unigrams can be matched based on their surface forms, stemmed forms, and meanings; furthermore, METEOR can be easily extended to include more advanced matching strategies.

- Fmean: Trung bình điều hòa của độ chính xác (Precision) và độ phủ (Recall) n-gram, với độ phủ thường được ưu tiên hơn.
- Penalty: Một yếu tố phạt được tính dựa trên số lượng *chunk*, phản ánh mức độ giữ đúng thứ tự từ trong câu dự đoán so với câu tham chiếu. Một chunk là một dãy n-gram khớp nhau và xuất hiện liên tiếp theo đúng thứ tự trong cả hai câu. Nếu các n-gram khớp nhưng phân tán hoặc sai thứ tự, chúng tạo thành nhiều chunk hơn. Số lượng chunk càng ít thì Penalty càng thấp, cho thấy câu tạo sinh dự đoán có trật tự tốt hơn và điểm METEOR sẽ cao hơn:

$$\text{Penalty} = \gamma \cdot (\frac{chunk}{m})^{\theta}, \tag{3.31}$$

trong đó:

- $\gamma \in [0, 1]$ là hệ số điều chỉnh cường độ phạt (thường lấy là 0.5),
- $\theta > 0$ là hệ số dốc phạt (thường lấy là 0.3).

Giá trị $(\frac{chunk}{m})^{\theta}$ càng lớn, tức các từ khớp càng rải rác, thì mức phạt càng cao.

Công thức tính độ chính xác và độ phủ đều tương tự với công thức (3.21) và (3.22). Tuy nhiên, thay vì dùng F1 tiêu chuẩn, METEOR sử dụng một công thức trung bình điều hòa có trọng số ưu tiên độ phủ để tính Fmean như sau:

$$\text{Fmean} = \frac{(1+\alpha) \cdot \text{Precision} \cdot \text{Recall}}{\alpha \cdot \text{Precision} + \text{Recall}}. \tag{3.32}$$

Trong công thức (3.32), $\alpha \in \mathbb{R}^+$, là hệ số điều chỉnh trọng số giữa độ chính xác và độ phủ. Trong nhiều nghiên cứu, $\alpha = 9$ giúp ưu tiên độ phủ hơn độ chính xác. Bởi lẽ, trong ngữ cảnh dịch máy hoặc hỏi đáp, việc bỏ sót thông tin thường nghiêm trọng hơn việc dư từ (Banerjee and Lavie, 2005, Denkowski and Lavie, 2010).

3.6.4 Đánh giá dựa trên AI

Với sự phát triển mạnh mẽ của LLMs, việc sử dụng AI để đánh giá chất lượng của các hệ thống AI khác đã trở thành một xu hướng mới và hứa hẹn. Phương pháp Đánh giá dựa trên AI tận dụng khả năng hiểu ngôn ngữ tự nhiên và đưa ra đánh giá phức tạp của các mô hình như GPT-4, Claude, hoặc LLMs khác để thay thế hoặc bổ sung cho việc đánh giá bằng con người trong nhiều tình huống. Đánh giá dựa trên AI mang lại nhiều lợi thế so với các phương pháp đánh giá truyền thống:

- **Khả năng mở rộng**: Có thể đánh giá hàng nghìn mẫu trong thời gian ngắn mà không cần nguồn lực lớn từ con người.
- **Tính nhất quán**: AI thường cho ra kết quả ổn định hơn so với các đánh giá viên con người, ít bị ảnh hưởng bởi tâm trạng hoặc thiên vị cá nhân.

- **Chi phí hiệu quả**: Giảm đáng kể chi phí so với việc thuê chuyên gia con người để đánh giá, đặc biệt quan trọng trong các nghiên cứu quy mô lớn.
- **Đánh giá đa chiều**: Có thể đồng thời đánh giá nhiều khía cạnh như độ chính xác, tính mạch lạc, phong cách, và tính sáng tạo.

Quá trình đánh giá dựa trên AI thường được thực hiện qua các bước sau:

1. **Thiết kế câu chỉ thị đánh giá**: Tạo ra các câu chỉ thị rõ ràng, chi tiết cho AI đánh giá, bao gồm tiêu chí đánh giá, thang điểm, và ví dụ minh họa.
2. **Cung cấp ngữ cảnh**: Đưa vào câu chỉ thị các thông tin cần thiết như câu hỏi gốc, câu trả lời cần đánh giá, và các tiêu chí chất lượng cụ thể.
3. **Thu thập đánh giá**: Sử dụng API của LLMs để thu thập điểm số và nhận xét chi tiết từ AI đánh giá.
4. **Xử lý kết quả**: Phân tích và tổng hợp các đánh giá, có thể bao gồm việc sử dụng nhiều AI đánh giá khác nhau để tăng độ tin cậy.

Mặc dù có nhiều ưu điểm, đánh giá dựa trên AI cũng đối mặt với một số thách thức:

- **Thiên vị mô hình**: AI đánh giá có thể có thiên vị đối với các phong cách viết hoặc cách tiếp cận tương tự như dữ liệu huấn luyện của chúng.
- **Độ tin cậy**: Cần được xác thực và đánh giá bằng cách so sánh với đánh giá của con người để đảm bảo tính chính xác.
- **Hiểu biết về lĩnh vực**: Trong các lĩnh vực chuyên môn cao, AI đánh giá có thể thiếu kiến thức sâu so với các chuyên gia con người.
- **Tính nhất quán trong câu chỉ thị**: Kết quả có thể thay đổi tùy thuộc vào cách thiết kế câu chỉ thị, đòi hỏi việc tinh chỉnh cẩn thận.

Chương 4

Triển khai hệ thống

Trong phần này, chúng tôi sẽ trình bày về quy trình tiền xử lý dữ liệu, để chuẩn bị dữ liệu đầu vào cho hệ thống RAG. Ngoài ra, chúng tôi cũng trình bày chi tiết về mô hình HippoRAG-2 được sử dụng.

4.1 Xây dựng chương trình xử lý dữ liệu học phần

4.1.1 Xác định vấn đề

Trong quá trình xây dựng hệ thống gợi ý môn học, chúng tôi nhận ra rằng bước đầu tiên – và cũng là một trong những bước quan trọng nhất, chính là chuẩn hóa dữ liệu đầu vào. Tập dữ liệu chúng tôi sử dụng được lấy từ website của Khoa Toán - Tin học. Tập dữ liệu này bao gồm ba file định dạng `.docx`, chứa thông tin về tất cả các học phần trong ba ngành học khác nhau thuộc Khoa Toán – Tin học, lần lượt là Khoa học dữ liệu[1], Toán Ứng dụng[2] và Toán Tin[3] . Mỗi file chứa nhiều bảng thông tin về các học phần, với cấu trúc gồm hai cột: cột bên trái liệt kê tên các trường, ví dụ như `Module designation`, `Semester(s) in which the module is taught`, v.v. như Bảng 4.2. Còn cột bên phải chứa thông tin tương ứng của từng học phần. Bảng 4.1 cho thấy thông tin về dung lượng và tổng số học phần có trong từng file ngành học.

File	Dung lượng	Tổng số học phần
Module-Handbook-DS-2021	3.86 MB	75
Module-Handbook-MCS-2021	3.90 MB	90
Module-Handbook-AM-2021	3.38 MB	94

Bảng 4.1: Danh sách các tập tin đầu vào và dung lượng tương ứng.

[1] DS
[2] AM
[3] MCS

15. Calculus 1A - MTH00011

Module designation	Calculus 1A
Semester(s) in which the module is taught	1st semester
Person responsible for the module	MSc. Nguyen Vu Huy
Language	Vietnamese
Relation to curriculum	Compulsory
Teaching methods	Lectures, group work, small group solving exercises
Workload (incl. contact hours, self-study hours)	150 Hours Contact hours: Lectures: 30 hours (in class); 30 exercise hours Private study: 90 hours (self-study)
Credit points	3 Credits (5 ECTS)
Required and recommended prerequisites for joining the module	None
Module objectives/intended learning outcomes	The objective of the module is to equip students with the basic knowledge of calculus as the foundation for specialized modules.
Content	The course covers the basics of continuity, limit, derivative, Riemann integral.
Examination forms	Midterm and final exam: written exams.
Study and examination requirements	Class attendance: at least 70%. Overall grade: minimum 5.0/10.0.
Reading list	1. Mathematica by Example, Academic Press, New York 2. Calculus, Harcourt Brace College Publishers, New York 3. Giáo trình Giải tích 1, Nhà xuất bản Thống Kê, TP. Hồ Chí Minh

Bảng 4.2: Bảng thông tin học phần MTH00011 trong Module-Handbook-AM-2021.docx.

Tuy nhiên, vì toàn bộ dữ liệu này đều được nhập tay, nên những sai sót là điều khó tránh khỏi. Những lỗi này không chỉ gây khó khăn trong việc phân tích mà còn ảnh hưởng trực tiếp đến độ chính xác của các hệ thống xử lý phía sau, đặc biệt là hệ thống tư vấn.

Cụ thể, chúng tôi phát hiện ra ba nhóm vấn đề chính:

a) **Không nhất quán về cấu trúc bảng:** Về nguyên tắc, mỗi bảng mô tả một học phần nên có đúng 14 hàng, tương ứng với 14 trường thông tin chuẩn. Tuy vậy, trong nhiều trường hợp, một số bảng vượt quá giới hạn này – có bảng lên đến 15 hoặc 16 hàng, và hầu hết những hàng dư này đôi khi là hàng trống hoặc người viết tự tạo thêm trường thông tin khác. Điều này dẫn đến việc trích xuất tự động gặp khó khăn.

b) **Thông tin giảng viên không đồng nhất:** Trường thông tin `Person responsible for the module` – tên người chịu trách nhiệm giảng dạy học phần, cũng xuất hiện nhiều bất nhất trong cách thể hiện. Cùng một giảng viên, nhưng tên của họ lại được nhập theo nhiều dạng khác nhau do người nhập thêm hoặc thay đổi học vị/danh xưng. Ví dụ: cùng một giảng viên có thể được ghi là *MSc. Nguyễn Văn A, M.Sc. Nguyễn Văn A, M.S. Nguyễn Văn A*, hoặc đơn giản là *Ms. Nguyễn Văn A*. Sự đa dạng trong cách viết này sẽ không chỉ gây khó khăn về mặt chi phí lưu trữ khi LLMs hiểu mỗi cách viết như một thực thể khác nhau, mà còn gây sai lệch trong việc nhận diện, gắn nhãn hoặc tổng hợp thông tin liên quan đến cùng một giảng viên.

c) **Thông tin học kỳ mở lớp quá đa dạng:** Trường `Semester(s) in which the module is taught` trong các học phần thường ghi theo thứ tự từ *1st, 2nd, 3rd,*..., đến *8th*. Ngoài ra, trên thực tế, số học kỳ tại trường còn được xác định bởi học kỳ chẵn và lẻ. Bởi lẽ, trong quá trình đăng ký học phần, hệ thống tư vấn chỉ cần xác định học kỳ hiện tại là chẵn hay lẻ, từ đó lọc ra các học phần phù hợp sẽ được mở trong học kỳ đó, áp dụng cho sinh viên ở nhiều khóa khác nhau.

d) **Học phần tiên quyết thiếu thống nhất:** `Required and recommended prerequisites for joining the module` là một trong những phần xử lý phức tạp nhất. Có những học phần ghi tên môn tiên quyết đầy đủ, nhưng lại dùng nhiều định dạng khác nhau, hoặc ghép nhiều môn trong cùng một dòng hoặc xuống hàng không theo cấu trúc rõ ràng. Đối với tên của các học phần tiên quyết, chúng tôi nhận thấy hầu hết các lỗi sai sẽ rơi vào hai trường hợp lần lượt là (i) lỗi sai chính tả, dư hoặc thiếu ký tự (Ví dụ như tên học phần đúng là *Analysis 1A*, nhưng người dùng sẽ nhập là *Analysist 1a*, hoặc *analysis i*); (ii) lỗi hoán vị từ (Ví dụ với trường hợp học phần *Introduction to Informatics*, tuy nhiên, người dùng lại nhập *Informatics introduction*). Chúng tôi đã có thống kê phần trăm thông tin tất cả các trường hợp tên học phần bị nhập sai cho lần lượt từng ngành là **11.25% ở DS, 8% ở AM và 8% ở MCS**.

Những thách thức này đặt ra yêu cầu về một chương trình xử lý dữ liệu môn học có khả năng tự động làm sạch, chuẩn hóa, và tái cấu trúc thông tin một cách chính xác và ổn định – đó chính là mục tiêu mà chúng tôi hướng đến trong phần tiếp theo.

4.1.2 Mục tiêu xử lý

Sau khi nhận diện rõ những bất cập trong dữ liệu học phần, mục tiêu hàng đầu của chúng tôi là thiết kế một chương trình xử lý dữ liệu cơ bản - chúng tôi gọi là *đường ống xử lý dữ liệu đầu vào*. Đường ống này có khả năng chuyển hóa toàn bộ dữ liệu đầu vào – vốn đang ở trạng thái rời rạc, không nhất quán, thành một tập dữ liệu đầu ra có cấu trúc rõ ràng, định dạng đồng nhất và thông tin tinh gọn. Ngoài ra, chương trình này có thể ứng dụng rộng rãi với các đề cương học phần ở các ngành học khác nhau, miễn dữ liệu đầu vào có định dạng tương tự DS, MCS hoặc AM.

Dữ liệu sau xử lý cần đảm bảo chỉ giữ lại những trường thông tin thực sự cần thiết, loại bỏ các phần dư thừa hoặc gây nhiễu. Cụ thể, mỗi ngành học sẽ được chuẩn hóa và biểu diễn dưới dạng một file JSON riêng biệt, với các trường quan trọng như mã học phần, tên học phần, tên giảng viên, các học phần tiên quyết, học kỳ mở, dự kiến chất lượng đầu ra và nội dung giảng dạy. Các giá trị trong từng trường đều được chuẩn hóa về mặt ngữ nghĩa và cú pháp để đảm bảo tính nhất quán xuyên suốt toàn bộ tập dữ liệu.

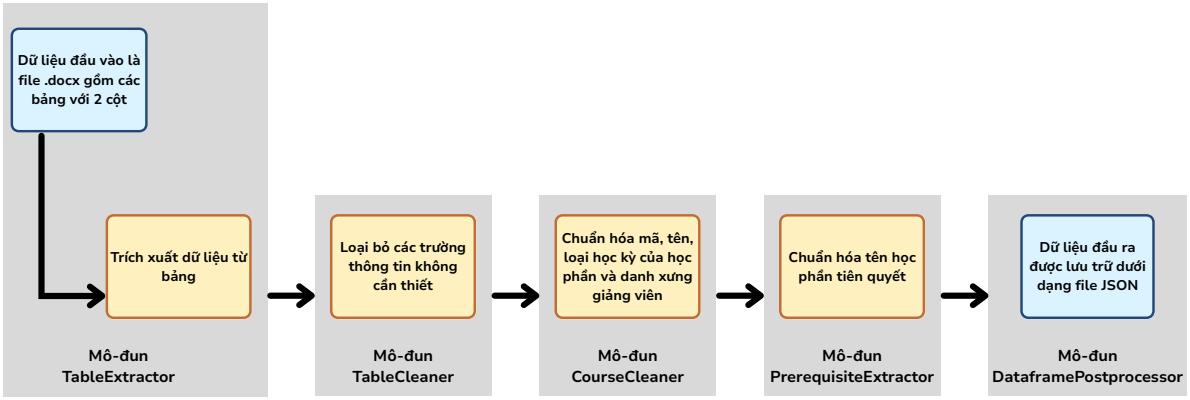
Đây không chỉ là bước xử lý dữ liệu đơn thuần, mà còn là tiền đề quan trọng cho hệ thống tư vấn của chúng tôi. Dữ liệu đầu ra là một dạng có cấu trúc được thiết kế theo hướng thân thiện với LLMs, giúp chúng nhận diện và trích xuất thông tin một cách chính xác, đồng thời tối ưu chi phí lưu trữ và truy vấn – đặc biệt khi hệ thống cần làm việc với hàng trăm học phần khác nhau.

Ví dụ dưới đây minh họa cấu trúc đầu ra của học phần trong Bảng 4.2 sau khi được xử lý:

JSON file

```
{
"MTH00010": [
    "course id: MTH00010",
    "course name: Analysis 1A",
    "semester: odd",
    "teacher name: Ong Thanh Hai",
    "course type: Compulsory",
    "required prerequisites: none",
    "learning outcomes: The objective of the module is to equip
    students with the basic knowledge of the foundation of calculus
    as the foundation for specialized modules.",
    "content: The course covers the basics of real numbers, sequences
    and series of real numbers."
    ]
}
```

4.1.3 Mô tả chương trình xử lý



Hình 4.1: Quy trình xử lý dữ liệu đầu vào thông qua các mô-đun.

Để chuyển hóa dữ liệu học phần từ định dạng `.docx` chưa chuẩn sang một cấu trúc rõ ràng, nhất quán và sẵn sàng cho hệ thống tư vấn, chúng tôi đã xây dựng một đường ống xử lý dữ liệu cơ bản gồm nhiều khối (mô-đun) thực hiện liền tiếp nhau, được tổ chức rõ ràng thành các mô-đun. Toàn bộ quy trình được minh họa qua Hình 4.1. Ngoài ra, chúng tôi sử dụng phiên bản python 3.10 cùng với các thư viện cơ bản và phổ biến như pandas và numpy.

Bước 1: Trích xuất bảng từ file .docx

Bước đầu tiên trong chương trình là trích xuất dữ liệu từ các file `.docx` chứa thông tin học phần. Chúng tôi đã xây dựng mô-đun `TableExtractor`, với nhiệm vụ đọc toàn bộ tài liệu và nhận diện từng bảng dữ liệu trong đó.

Điểm đặc biệt của file `.docx` đầu vào là các bảng không có trường thông tin về mã học phần. Do đó, chúng tôi đã xây dựng cơ chế tìm lại đoạn văn bản ngay trước mỗi bảng – chính là mã học phần, và gán nó như một trường đầu tiên có tên `course id`. Ở đây, chúng tôi sử dụng thư viện python-docx phiên bản 1.1.2. Thư viện này hỗ trợ định vị và xác định được nội dung trong dạng bảng từ file .docx.

Sau đó, từng bảng được trích xuất theo cấu trúc hàng – cột, mỗi ô được loại bỏ khoảng trắng đầu – cuối. Các bảng này được lưu trữ dưới dạng cấu trúc ba lớp danh sách lồng nhau, trong đó mỗi phần tử đại diện cho một học phần và chứa các cặp trường – giá trị đã được trích xuất, sẵn sàng cho bước làm sạch dữ liệu tiếp theo.

Định dạng dữ liệu sau khi trích xuất từ dạng bảng

```
[
    [   # Bảng 0
        ['Course id', 'BAA00101'],
        ['Module designation', 'Marxist-Leninist Philosophy'],
        ['Semester(s) in which the module is taught', 'odd'],
        ['Person responsible for the module', 'Assigned lecturers at
        University of Science, VNU-HCM'],
        ['Relation to curriculum', 'Compulsory'],
        ['Required and recommended prerequisites for joining the mo-
        dule', 'None'],
        ['Module objectives/intended learning outcomes',
         'The course equips students with the basic contents of the
         worldview...],
        ['Content', 'Marxist-Leninist philosophy is a...]
    ],
    [...],
    [...]
]
```

Bước 2: Lọc và làm sạch các trường không cần thiết

Sau khi trích xuất, chúng tôi xác định những trường thông tin không cần thiết (không nằm trong 10 tình huống nghiên cứu mà chúng tôi đề cập ở Bảng 1.1 như `Language`, `Teaching methods`, `Workload (incl. contact hours, self-study hours)`, `Credit points`, `Examination forms`, `Study and examination requirements`, `Reading list`. Do đó, mô-đun `TableCleaner` có tác vụ là lọc bỏ các dữ liệu này.

Bước 3: Chuẩn hóa thông tin học phần

Sau khi các bảng dữ liệu được lọc và làm sạch về mặt cấu trúc, bước tiếp theo là chuẩn hóa nội dung của từng trường thông tin. Lúc này, mô-đun `CourseCleaner` đảm nhận vai trò tinh chỉnh từng chi tiết nhỏ trong từng bảng học phần - giúp dữ liệu trở nên đồng nhất và dễ xử lý hơn trong các bước sau.

Trước hết, `CourseCleaner` sẽ cắt gọn mã học phần để chỉ giữ lại 8 ký tự cuối cùng - thường là phần định danh chính thức trong hệ thống đào tạo. Tiếp theo, trường học kỳ được quy đổi về dạng chữ là "odd" hoặc "even"[4].

Một điểm nhấn đáng chú ý nằm ở bước làm sạch tên giảng viên. `CourseCleaner` sẽ làm sạch và gom nhóm tên giảng viên: ví dụ, *MSc. Nguyen Van A, M.S. Nguyen Van A* đều được gom thành *Nguyen Van A*. Cuối cùng, hệ thống còn chuẩn hóa tên học phần và đặc biệt là trường điều kiện tiên quyết `prerequisites`, cụ thể là loại bỏ các ký tự

[4] lẻ hoặc chẵn

như *Recommended prerequisites:, Recommendation:, Course requirements: , Prerequisite courses:* đảm bảo chỉ giữ lại phần thông tin cốt lõi.

Kết quả sau bước này là một danh sách các bảng học phần với nội dung đã được tinh gọn, đồng nhất và sẵn sàng để trích xuất thông tin chuyên sâu hơn, như chuỗi điều kiện học phần.

Bước 4: Trích xuất và chuẩn hóa điều kiện học phần tiên quyết

Sau khi dữ liệu học phần đã được chuẩn hóa, chương trình bước vào giai đoạn trích xuất các học phần tiên quyết. Đây là một trường thông tin quan trọng giúp xác định mối quan hệ giữa các học phần.

Mô-đun `PrerequisiteExtractor` đảm nhiệm nhiệm vụ này thông qua chuỗi xử lý nhiều lớp. Đầu tiên, nó chuyển đổi các bảng đã làm sạch thành một `DataFrame` và tách nhỏ các điều kiện học phần tiên quyết (nếu có nhiều học phần được liệt kế cùng nhau - được phân cách bằng dấu phẩy `,`, chấm phẩy `;` hoặc dòng mới). Việc này đảm bảo mỗi điều kiện được xem xét một cách riêng biệt và được biểu diễn minh họa ở hình 4.2.

	course_id	course_name	prerequisites
0	BAA00012	English 2	english 1
1	MTH00040	Probibability and Statistics	calculus b1.
2	PHY00002	General Physics 2	calculus 1b
3	PHY00002	General Physics 2	general physics 1
4	MTH10336	Software Testing 2	software testing
5	MTH10336	Software Testing 2	object oriented programming
...	...	...	...

Hình 4.2: Minh họa một số dòng về các học phần tiên quyết được xử lý tách ra thành từng hàng riêng trong dataframe.

Để chuẩn hóa tên các học phần tiên quyết, mô-đun `PrerequisiteExtractor` sẽ so khớp, tìm độ tương đồng lần lượt giữa từng tên học phần tiên quyết trong dataframe, và danh sách tên học phần chuẩn. Ở đây, chúng tôi sử dụng thuật toán Jaccard - dựa trên kết quả được đánh giá và so sánh ở Bảng 4.4, cho phép chương trình tự động nhận diện các điều kiện ngay cả khi chúng bị viết sai chính tả hoặc hoán vị từ.

Để thực hiện tác vụ này, trước tiên chúng tôi chuyển toàn bộ tên các học phần tiên quyết về dạng chữ thường. Sau đó, một từ điển ánh xạ được xây dựng, trong đó khóa là tên học phần đã được chuyển sang chữ thường, và giá trị tương ứng là tên học phần chuẩn đúng. Thuật toán Jaccard sẽ lần lượt so khớp từng giá trị trong cột `prerequisites` của dataframe với các khóa trong từ điển này, nhằm tìm ra tên học phần chuẩn nếu chuỗi đầu vào đạt độ tương đồng cao nhất với một khóa trong từ điển.

Kết quả của thuật toán sẽ được lưu trữ tại cột `answer` như hình 4.3

	course_id	course_name	prerequisites	answer
0	BAA00012	English 2	english 1	English 1
1	MTH00040	Probibability and Statistics	calculus b1.	Calculus 1B
2	PHY00002	General Physics 2	calculus1b	Calculus 1B
3	PHY00002	General Physics 2	general physics 1	General Physics 1
4	MTH10336	Software Testing 2	software testing	Software Testing 1
5	MTH10336	Software Testing 2	object oriented programming	Object Oriented Programming
...	...	...	...	...

Hình 4.3: Tên các học phần tiên quyết đã được chuẩn hóa được lưu ở cột `answer`.

Bước 5: Hậu xử lý và xuất file JSON

Tiếp theo, hệ thống gộp các hàng theo mã học phần `course_id`, đồng thời tổng hợp tất cả các điều kiện tiên quyết thành một chuỗi duy nhất - mỗi tên học phần tiên quyết sẽ được cách nhau bởi dấu phẩy `,` loại bỏ trùng lặp và định dạng lại cho thống nhất. Cuối cùng là lưu trữ và xuất ra file JSON hoàn chỉnh.

Kết quả thực thi

File	Dung lượng đầu vào	Dung lượng đầu ra	Thời gian trung bình	ĐLC tổng thể
DS_output.json	3.86 MB	95 KB	0.82s	0.01
MCS_output.json	3.90 MB	113 KB	1.84s	0.02
AM_output.json	3.38 MB	112 KB	1.95s	0.01

Bảng 4.3: Kết quả thực thi chương trình xử lý dữ liệu học phần ở cả 3 tập dữ liệu đầu vào, thông qua dung lượng đầu vào-ra, trung bình thời gian thực thi và độ lệch chuẩn (ĐLC) tổng thể tương ứng trong 50 lần chạy thực nghiệm.

Bảng 4.3 trình bày kết quả thực thi chương trình xử lý dữ liệu học phần trên ba bộ dữ liệu đầu vào. Thời gian trung bình sau 50 lần chạy thử dao động từ 0.82s đến 1.95s, cho thấy tốc độ xử lý khá nhanh. Đặc biệt, độ lệch chuẩn của thời gian đều rất thấp (từ 0.01 đến 0.02), phản ánh sự ổn định cao trong quá trình thực thi. Kết quả này cho thấy chương trình vận hành hiệu quả và nhất quán, ngay cả khi xử lý các bộ dữ liệu có quy mô khác nhau.

4.1.4 Đánh giá và so sánh kết quả của các thuật toán đo độ tương đồng

Trước khi tiến hành đánh giá hiệu quả của các thuật toán đo độ tương đồng chuỗi, chúng tôi đã xây dựng 2 bộ dữ liệu kiểm tra chuyên biệt dành cho từng ngành DS, MCS và AM. Mục tiêu là để mô phỏng thực tế quá trình nhập thủ công tên các học phần tiên quyết với 2 dạng sai lệch khác nhau lần lượt là (i) lỗi hoán vị từ và (ii) lỗi chính tả bao

gồm dư hoặc thiếu ký tự (đã được đề cập trong Phần 4.1.1). Ngoài ra, 2 bộ dữ liệu kiểm tra này sẽ được thiết kế với độ "nhiễu" và tính thử thách cao hơn dữ liệu đầu vào thực tế, nhằm giúp chúng tôi kiểm định khả năng tổng quát hóa của từng thuật toán trong nhiệm vụ tìm kiếm tên học phần đúng.

Đối với bộ kiểm tra đầu tiên là lỗi hoán vị từ, DS sẽ có 75 bản ghi, MCS sẽ có 90 bản ghi và AM có 94 bản ghi. Chúng tôi chủ động xây dựng bộ kiểm tra với số lượng nhãn là bằng nhau, tương ứng với tổng số học phần trong mỗi ngành học. Ở mỗi bản ghi sẽ là một cặp dữ liệu: `prerequisites` - tên học phần bị làm nhiễu (giả lập lỗi do người dùng nhập sai); `ground_truth` - nhãn đúng tương ứng là tên học phần chuẩn.

Đối với bộ kiểm tra gồm các lỗi chính tả, số lượng bản ghi lần lượt ở mỗi ngành là: DS với 150 bản ghi, MCS với 180 bản ghi và AM với 188 bản ghi. Tương tự với bộ lỗi hoán vị từ, số lượng nhãn trong mỗi ngành sẽ đồng đều nhau, mỗi học phần sẽ có 2 mô phỏng lỗi sai chính tả, thiếu hoặc dư các ký tự. Mỗi bản ghi cũng sẽ có một cặp dữ liệu `prerequisites` và `ground_truth`.

Khi đã có 2 bộ dữ liệu kiểm tra, câu hỏi tiếp theo là *Chúng tôi nên đánh giá chất lượng khôi phục của các thuật toán này như thế nào cho đúng và công bằng?*. Tập kiểm tra được thiết kế dưới dạng bài toán phân loại đa nhãn[5] - nơi mỗi `prerequisites` cần được ánh xạ về đúng một `ground_truth` duy nhất trong tập đầu ra. Đặc biệt, do số lượng nhãn tương đối đồng đều, chúng tôi đã lựa chọn các chỉ số định lượng phù hợp (Grandini et al., 2020):

Accuracy

Độ chính xác Accuracy là một trong những chỉ số đánh giá phổ biến nhất trong các bài toán phân loại, bao gồm cả bài toán phân loại nhiều nhãn. Accuracy đo lường tỷ lệ dự đoán chính xác trên tổng số mẫu, tức là cho biết bao nhiêu phần trăm dự đoán của mô hình khớp hoàn toàn với nhãn thực tế.

Trong ngữ cảnh của bài toán hiện tại, mỗi chuỗi sai lệch được ánh xạ về đúng một nhãn `ground_truth` duy nhất, nên việc áp dụng Accuracy là hoàn toàn phù hợp để đánh giá hiệu năng tổng thể của thuật toán đo độ tương đồng.

$$\text{Accuracy} = \frac{1}{N}\sum_{i=1}^{N} 1(y_i = \hat{y}_i), \tag{4.1}$$

trong đó:

- N: Tổng số lượng mẫu trong tập kiểm tra,
- y_i: Nhãn thực tế của mẫu thứ i,
- $\hat{y}_i$: Nhãn mà thuật toán dự đoán cho mẫu thứ i,
- $1(\cdot)$: Hàm trả về 1 nếu điều kiện đúng, và 0 nếu ngược lại.

[5] multi-class classìication

Macro F1-score

Trong bài toán phân loại đa nhãn, hai chỉ số quan trọng là Precision và Recall được tính riêng biệt cho từng lớp trước khi tiến hành tổng hợp thành chỉ số Macro F1-Score. Cụ thể, Precision đo lường độ chính xác của thuật toán khi gán nhãn: nó được tính bằng tỷ lệ giữa số chuỗi được gán đúng nhãn (so với `ground_truth`) trên tổng số chuỗi được gán nhãn đó. Precision cao đồng nghĩa với việc thuật toán ít mắc lỗi khi đưa ra quyết định phân loại. Trong khi đó, Recall phản ánh khả năng phát hiện đầy đủ các nhãn thực sự tồn tại, được tính bằng tỷ lệ giữa số chuỗi được nhận diện đúng nhãn trên tổng số chuỗi thực sự thuộc nhãn đó trong `ground_truth`. Giá trị Recall cao cho thấy thuật toán ít bỏ sót các trường hợp đúng.

Bước 1: Tính Precision và Recall cho nhãn i:

$$\text{Precision}_i = \frac{TP_i}{TP_i + FP_i},$$

$$\text{Recall}_i = \frac{TP_i}{TP_i + FN_i},$$

trong đó:

- TP_i: Số mẫu thuộc nhãn i được lựa chọn đúng,
- FP_i: Số mẫu không thuộc nhãn i nhưng lại bị lựa chọn sai thành nhãn i,
- FN_i: Số mẫu thuộc nhãn i nhưng lại bị lựa chọn sai thành nhãn khác.

Sau đó tính trung bình cộng để thu được Macro Average Precision và Macro Average Recall, với N là tổng số nhãn. Cuối cùng là kết hợp hai giá trị này theo công thức F1 thông thường thì ta sẽ thu được kết quả của Macro F1 như sau

Bước 2: Tính Macro Average Precision và Macro Average Recall

$$\text{MacroAveragePrecision} = \frac{1}{N}\sum_{i=1}^{N}\text{Precision}_i, \tag{4.2}$$

$$\text{MacroAverageRecall} = \frac{1}{N}\sum_{i=1}^{N}\text{Recall}_i. \tag{4.3}$$

Bước 3: Tính Macro F1-Score

$$\text{Macro F1-Score} = 2 \times \left(\frac{\text{MacroAveragePrecision} \times \text{MacroAverageRecall}}{\text{MacroAveragePrecision}^{-1} + \text{MacroAverageRecall}^{-1}}\right). \tag{4.4}$$

Do cách tính trung bình không có trọng số này của chỉ số Macro F1-Score như trong công thức (4.4), các nhãn – dù xuất hiện nhiều hay ít đều sẽ có đóng góp ngang nhau vào

Bảng 4.4: So sánh kết quả của các thuật toán đo độ tương đồng chuỗi giữa hai tập dữ liệu kiểm tra có lỗi hoán vị từ và không có lỗi hoán vị từ ở cả 3 ngành, thông qua các chỉ số Accuracy (Acc.) và Macro F1-Score (MacroF1). Các kết quả vượt trội sẽ được tô đậm.

	DS		**MCS**		**AM**		**Average**	
	Acc.	**MacroF1**	**Acc.**	**MacroF1**	**Acc.**	**MacroF1**	**Acc.**	**MacroF1**
Tập kiểm tra có hoán vị từ								
Levenshtein	0.18	0.13	0.22	0.16	0.18	0.14	0.19	0.14
LCS	0.32	0.24	0.31	0.23	0.17	0.14	0.26	0.20
Jaccard	**0.96**	**0.95**	**0.96**	**0.94**	**0.97**	**0.96**	**0.96**	**0.95**
Jaro-Winkler	0.41	0.34	0.47	0.39	0.32	0.26	0.40	0.33
Tập kiểm tra không có hoán vị từ								
Levenshtein	**0.99**	**0.99**	**0.98**	**0.97**	**0.99**	**0.99**	**0.99**	**0.98**
LCS	0.99	0.99	0.97	0.95	0.98	0.98	0.98	0.97
Jaccard	**0.99**	**0.99**	**0.98**	**0.97**	**0.99**	**0.99**	**0.99**	**0.98**
Jaro-Winkler	0.99	0.99	0.97	0.96	0.98	0.98	0.98	0.98

kết quả cuối cùng. Trong ngữ cảnh của bài toán hiện tại, như đã trình bày ở trên, cả 2 bộ kiểm tra đã được chúng tôi thiết kế với số lượng nhãn phân bố đồng đều, đi cùng với mục tiêu là kiểm tra xem thuật toán có nhận diện đúng nhãn tương ứng với `ground_truth` hay không. Vì vậy, Macro-F1 là lựa chọn phù hợp, cho phép đánh giá khả năng phân biệt và nhận diện nhãn của thuật toán một cách toàn diện trên mọi nhãn, bất kể độ phổ biến của nhãn đó.

Bảng 4.4 trình bày kết quả so sánh hiệu suất của bốn thuật toán đo độ tương đồng chuỗi trên hai loại tập kiểm tra: (i) có lỗi hoán vị từ và (ii) không có lỗi hoán vị từ, áp dụng trên ba ngành học khác nhau gồm DS, MCS và AM. Ở tập kiểm tra có lỗi hoán vị từ, thuật toán Jaccard thể hiện hiệu quả vượt trội với Accuracy và Macro F1 cao nhất trên cả ba ngành (trung bình đạt 0.96 và 0.95). Trong khi đó, Levenshtein và LCS lại cho kết quả thấp đáng kể trong bối cảnh này, cho thấy khả năng xử lý lỗi hoán vị từ của chúng còn hạn chế do không xem xét hoán đổi vị trí từ trong chuỗi. Ngược lại, khi áp dụng lên tập kiểm tra không có lỗi hoán vị từ (chỉ chứa lỗi chính tả, thiếu/thừa ký tự), Levenshtein cho kết quả áp đảo với độ chính xác gần như tuyệt đối (trung bình đạt 0.99 cho Accuracy và 0.98 cho Macro F1), tiếp theo là Jaccard và LCS với hiệu suất tương đương. Điều này phản ánh rõ ưu thế của Levenshtein trong việc phát hiện các lỗi ký tự nhỏ lẻ nhờ dựa trên số phép biến đổi chuỗi tối thiểu. Nhìn chung, kết quả bảng cho thấy rằng việc lựa chọn thuật toán phù hợp nên phụ thuộc vào loại lỗi chủ đạo trong dữ liệu đầu vào rằng: Jaccard sẽ là lựa chọn tối ưu cho lỗi hoán vị từ, trong khi Levenshtein phát huy hiệu quả mạnh mẽ khi xử lý lỗi chính tả và ký tự.

4.2 Lựa chọn đề xuất sử dụng mô hình RAG đồ thị

Trong thế giới tự nhiên phong phú và không ngừng biến động, bộ não của các loài động vật có vú đã tiến hóa để sở hữu năng lực đặc biệt là lưu trữ khối lượng lớn thông tin

và liên tục tiếp thu tri thức mới mà không làm mất đi những gì đã học. Nổi bật nhất là bộ não con người — đỉnh cao của quá trình tiến hóa này, với khả năng không chỉ ghi nhớ, mà còn tích hợp, kết nối và cập nhật kiến thức một cách linh hoạt theo thời gian. Năng lực này chính là nền tảng cho khả năng lập luận, thích nghi và ra quyết định trong những tình huống phức tạp của đời sống.

Lấy cảm hứng từ cơ chế hoạt động của bộ não, các nhà nghiên cứu trong lĩnh vực trí tuệ nhân tạo đã nỗ lực phát triển LLMs với khả năng học hỏi và ghi nhớ hiệu quả hơn. Tuy đạt được nhiều tiến bộ, LLMs vẫn gặp khó khăn trong việc tiếp nhận và tích hợp tri thức mới sau giai đoạn tiền huấn luyện. Ngay cả khi được hỗ trợ bởi các phương pháp hiện đại như RAG, chúng vẫn xử lý thông tin một cách rời rạc. Hạn chế này càng rõ rệt trong các tác vụ yêu cầu tích hợp thông tin đa bước suy luận như phân tích khoa học, tóm tắt pháp lý hay chẩn đoán y tế.

Chính trong bối cảnh đó, HippoRAG được xây dựng bởi Gutiérrez et al. (2024) đã được giới thiệu tại hội nghị NeurIPS 2024[6], ra đời như một hướng tiếp cận mới mẻ, lấy cảm hứng từ thuyết *lập chỉ mục hồi hải mã*[7] trong nghiên cứu thần kinh học. Thuyết này cho rằng bộ não con người lưu trữ ký ức dài hạn bằng cách lập chỉ mục thông tin trong hồi hải mã[8], sau đó phối hợp với vỏ não mới[9] để tái dựng lại ký ức khi cần thiết (Teyler and DiScenna, 1986). Dựa trên lý thuyết này, HippoRAG mô phỏng cơ chế nhớ của con người bằng cách kết hợp LLMs với đồ thị tri thức và thuật toán PPR, tạo thành một hệ thống truy xuất hợp tác giữa "trí nhớ ngắn hạn" LLM và "trí nhớ dài hạn có tổ chức".

Tuy nhiên, phiên bản đầu tiên của HippoRAG — dù đã cho thấy hiệu quả nổi bật trong các tác vụ đòi hỏi khả năng liên tưởng và tạo lập nghĩa, vẫn chưa thực sự thuyết phục trong các tác vụ học liên tục và trí nhớ dài hạn. Để khắc phục điểm yếu đó và đạt được hiệu suất toàn diện hơn, đồng sáng lập Gutiérrez et al. (2025) phát triển HippoRAG-2. Hệ thống mới này hướng đến việc cân bằng và tối ưu hóa hiệu năng trên cả ba loại nhiệm vụ trí nhớ quan trọng: trí nhớ thực tế, trí nhớ tạo nghĩa và trí nhớ liên tưởng[10] — mở ra tiềm năng lớn cho các ứng dụng trong thế giới thực đòi hỏi khả năng suy luận phức tạp và tích hợp kiến thức sâu rộng.

4.2.1 Tổng quan về hệ thống HippoRAG-2

HippoRAG-2 là phiên bản cải tiến của hệ thống HippoRAG, được thiết kế nhằm mô phỏng ba thành phần cốt lõi trong cơ chế trí nhớ dài hạn của con người, dựa trên các kiến thức từ khoa học thần kinh. Bên cạnh đó, HippoRAG-2 được giới thiệu và nhấn mạnh là một khuôn[11] ghi nhớ hiệu quả cho các LLMs, được thiết kế nhằm tăng cường khả năng nhận diện và khai thác mối liên kết giữa các tri thức mới – tương tự như cách trí nhớ dài hạn của con người hoạt động. Mô hình này không chỉ nâng cao hiệu suất trong các tác vụ suy luận nhiều bước và tích hợp ngữ cảnh phức tạp, mà còn duy trì được hiệu

[6] NeurIPS (Conference on Neural Information Processing Systems) là một trong những hội nghị hàng đầu thế giới về trí tuệ nhân tạo và học máy, nơi công bố nhiều nghiên cứu tiên phong trong lĩnh vực này.
[7] hippocampal indexing theory
[8] hippocampus
[9] neocortex
[10] factual, semantic and associative memory
[11] framework

quả ở các tác vụ đơn giản. So với các phương pháp dựa trên đồ thị khác như GraphRAG (Edge et al., 2024), RAPTOR (Sarthi et al., 2024) hay LightRAG (Guo et al., 2024), HippoRAG-2 cho thấy ưu thế rõ rệt về chi phí và độ trễ khi thực thi trực tuyến, đồng thời tiết kiệm đáng kể tài nguyên cho giai đoạn lập chỉ mục ngoại tuyến.

Về tổng quát, có thể nói mô hình HippoRAG-2 được thiết kế với cấu trúc tổng quát tương tự với H-GraphRAG (chúng tôi đã đề cập ở Phần 3.1.2). Quy trình hoạt động của HippoRAG-2 được chia thành hai luồng hoạt động lần lượt là (i) lập chỉ mục ngoại tuyến và (ii) truy xuất trực tuyến. Trước khi trình bày chi tiết về từng luồng hoạt động, ở phần tiếp theo, chúng tôi sẽ trình bày về định dạng dữ liệu đầu vào tiêu chuẩn của hệ thống.

4.2.2 Chuẩn bị cho dữ liệu đầu vào

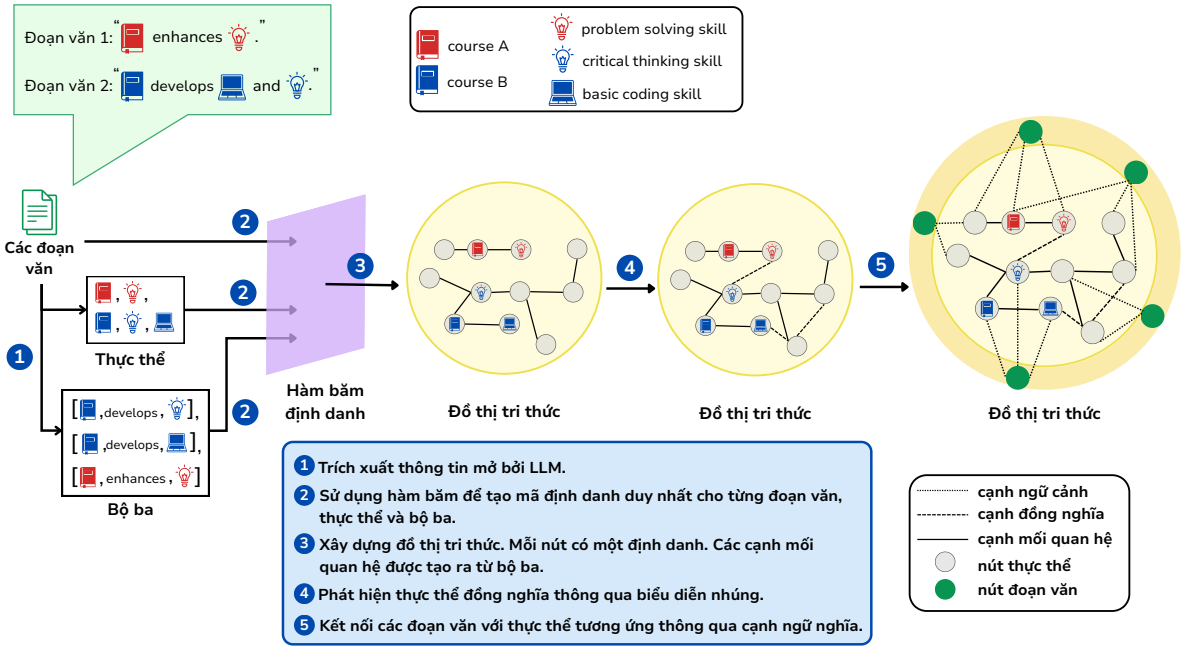
Các thành phần trong chuẩn đầu vào của hệ thống HippoRAG-2

```
[
    {
        "idx": 1,
        "title": "BAA00101",
        "text": "Course Marxist-Leninist Philosophy (ID: BAA00101)
        falls under the Compulsory category and ..."
    },
    {
        "idx": 2,
        "title": "BAA00102",
        "text": "In the odd semester, students can take Marxist-
        Leninist Political Economy (code: BAA00102). It is taught by
        Assigned lecturers at University of Science, ..."
    },
    {
        "idx": 3,
        "title": "BAA00103",
        "text": "Scientific Socialism (ID: BAA00103) is a Compulsory
        subject conducted during the even semester. The course is
        instructed by Lecturers at School of Political and ..."
    },
    {...}, {...}
]
```

Định dạng chuẩn đầu vào của hệ thống bao gồm ba trường thông tin chính, mỗi trường đảm nhận một chức năng cụ thể trong quy trình xử lý. Trường `id` là định danh duy nhất cho mỗi tài liệu, được đánh số từ 1 đến N (với N là tổng số học phần trong một ngành học). Trường `title` chính là tiêu đề tài liệu tương ứng, ở đây chúng tôi chọn sử dụng mã học phần làm tiêu đề cho tài liệu học phần. Đối với trường `text`, chúng tôi đã tạo ra 5 đoạn văn mẫu với các khoảng trống (được trình bày ở Phần Phụ lục A. Mô tả 5 đoạn văn mẫu). Sau đó, chúng tôi sẽ lắp các trường thông tin và dữ liệu được lấy từ 3 files kết quả từ quá trình xử lý dữ liệu đầu vào (ở Phần 4.1.3) là

`DS_output.json, MCS_output.json` và `AM_output.json` vào các khoảng trống ấy.

4.2.3 Luồng lập chỉ mục ngoại tuyến



Hình 4.4: Quy trình hoạt động của hệ thống HippoRAG-2 ở luồng lập chỉ mục ngoại tuyến.

Luồng hoạt động này được mô phỏng theo quá trình bộ nhớ của con người mã hóa và lưu trữ thông tin thành ký ức — nơi thông tin được xử lý, tổ chức và lưu trữ một cách có cấu trúc để phục vụ cho việc truy xuất sau này. Quá trình này tương ứng với việc chuyển đổi các tài liệu văn bản đầu vào thành một biểu diễn tri thức có cấu trúc dưới dạng đồ thị tri thức (Hình 4.4). Cụ thể các bước như sau:

Bước 1: LLM trích xuất thông tin mở từ các đoạn văn

Một LLM đã được huấn luyện theo các câu lệnh hướng dẫn chỉ thị - đóng vai trò như vỏ não mới nhân tạo, được sử dụng để thực hiện nhiệm vụ trích xuất thông tin mở (OpenIE). Cụ thể, LLM sẽ xử lý từng đoạn văn trong tập dữ liệu và chuyển đổi chúng thành các bộ ba, được gọi là *triples*. Mỗi bộ ba bao gồm ba thành phần: chủ ngữ, vị ngữ và đối tượng (tân ngữ), đại diện cho các thực thể và mối quan hệ giữa chúng. Không dừng lại ở đó, LLM sẽ tiếp tục trích xuất các thực thể từ chủ ngữ và đối tượng ở các bộ ba.

Bước 2: Mã hóa định danh

Sau khi hoàn thành bước 1, hệ thống sẽ bắt đầu mã hóa để tạo định danh duy nhất cho từng đoạn văn, thực thể và bộ ba thông qua một hàm băm (tác giả sử dụng hàm băm

mật mã học MD5[12]).

Bước 3: Xây dựng đồ thị tri thức

Hệ thống tiến hành xây dựng đồ thị tri thức với các nút thực thể. Mỗi nút sẽ mang một định danh và chúng sẽ được liên kết với nhau theo hai chiều[13] thông qua cạnh mối quan hệ từ các bộ ba.

Bước 4: Phát hiện từ đồng nghĩa

Song song với quá trình định danh để xây dựng đồ thị, hệ thống còn thực hiện biểu diễn các thực thể, bộ ba và đoạn văn trong không gian vector sử dụng một mô hình nhúng. Để xây dựng được các cạnh đồng nghĩa giữa các nút thực thể trong đồ thị, hệ thống cần tính toán độ tương đồng giữa các vector. Cụ thể, HippoRAG-2 sử dụng độ tương đồng cosine, được tính bằng tích vô hướng giữa các vector đã được chuẩn hóa L2 - tức đưa vector về độ lớn bằng 1 và vẫn giữ nguyên hướng của nó (chúng tôi cũng đã trình bày ở Phần 3.3.3).

Sau khi tính ra các điểm tương đồng, hệ thống sẽ lọc bớt những cặp có điểm thấp hơn một ngưỡng xác định (mặc định là 0.8) và chỉ giữ lại những cặp có độ tương đồng cao. Cuối cùng, những điểm tương đồng này sẽ được lưu làm trọng số cho các cặp nút, từ đó tạo cạnh đồng nghĩa giữa chúng trong đồ thị.

Bước 5: Tích hợp dày đặc và thưa thớt

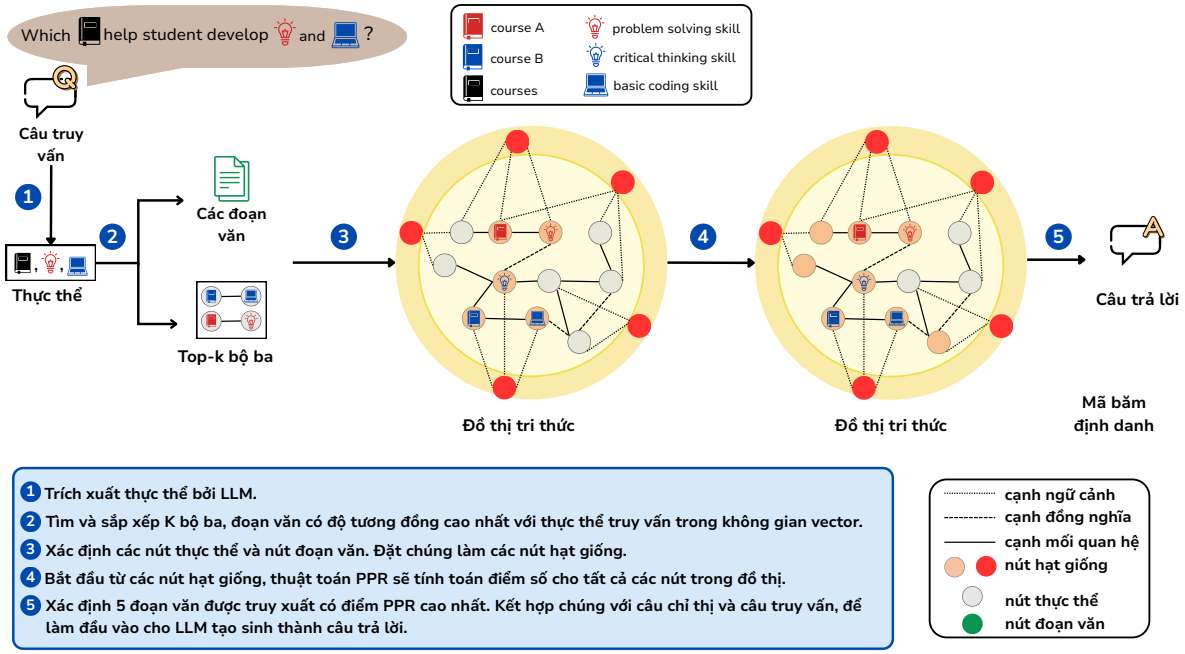
Đây là một cải tiến quan trọng của HippoRAG-2 so với HippoRAG ban đầu, vốn bị hạn chế bởi cách tiếp cận tập trung vào thực thể, từ đó gây mất nội dung về ngữ cảnh cho hệ thống. Lấy cảm hứng từ quá trình tích hợp mã hóa dày đặc và thưa thớt trong bộ não con người, HippoRAG-2 coi nút thực thể như một dạng mã hóa thưa thớt cho các khái niệm được trích xuất và tích hợp mã hóa dày đặc vào đồ thị tri thức bằng cách giới thiệu nút đoạn văn cùng với các cạnh ngữ nghĩa, tức là liên kết giữa nút đoạn văn và các nút thực thể tương ứng nằm trong chúng. Quá trình này giúp hệ thống tiến gần hơn đến hiệu quả của trí nhớ dài hạn của con người, bằng cách giải quyết các khoảng trống về khả năng hiểu ngữ nghĩa và tính liên kết.

4.2.4 Luồng truy xuất trực tuyến

Sau khi hoàn tất lập chỉ mục ngoại tuyến, hệ thống bước vào luồng truy xuất trực tuyến - mô phỏng dựa trên quá trình truy xuất ký ức trong não bộ con người. So với HippoRAG ban đầu, HippoRAG-2 đã cải tiến đáng kể khả năng kết nối truy vấn của người dùng với các khái niệm, mối liên hệ và ngữ cảnh từ các nút đoạn văn trong đồ thị tri thức (Hình

[12]Message Digest Algorithm 5 (MD5) là một hàm băm mật mã học tạo ra một chuỗi băm có độ dài cố định 128 bit (16 byte) từ một đầu vào có độ dài bất kỳ. Kết quả thường được biểu diễn dưới dạng 32 ký tự hệ thập lục phân.

[13]bidirectional edge connection



Hình 4.5: Quy trình hoạt động của hệ thống HippoRAG-2 ở luồng truy xuất trực tuyến.

4.5). Điều này đảm bảo cho hệ thống không những truy xuất được các đơn vị thông tin tương ứng, mà còn hiểu được ngữ nghĩa và bối cảnh tiềm ẩn giữa câu truy vấn và đồ thị tri thức. Luồng trực tuyến này sẽ lần lượt đi qua 5 bước chính như sau:

Bước 1: Trích xuất và nhúng thực thể từ câu truy vấn

Khi người dùng nhập một câu truy vấn Q, hệ thống sẽ sử dụng LLM để trích xuất tập hợp các thực thể Q_e từ truy vấn này, dựa trên các câu lệnh chỉ thị. Sau khi được trích xuất, các thực thể này sẽ được ánh xạ vào không gian vector để chuẩn bị cho các bước tiếp theo.

Bước 2: Truy xuất bộ ba liên quan và đoạn văn bằng truy xuất dày đặc

Trong không gian vector, hệ thống sẽ tính toán độ tương đồng cosine giữa các vector thực thể Q_e và vector biểu diễn của các bộ ba. Dựa vào các điểm tương đồng này, hệ thống sẽ lựa chọn top-k bộ ba có độ tương đồng cao nhất (mặc định $k = 200$). Đồng thời, hệ thống cũng thực hiện truy xuất dày đặc đoạn văn DPR[14] bằng cách tính toán độ tương đồng cosine giữa truy vấn và toàn bộ các đoạn văn trong đồ thị, nhằm xác định các đoạn văn có liên quan trực tiếp đến câu truy vấn.

Bước 3: Xác định các nút hạt giống

Các nút hạt giống đóng vai trò là điểm khởi đầu cho thuật toán PPR trong quá trình truy xuất trực tuyến. Các nút hạt giống được phân loại thành hai loại chính là nút thực

[14]Dense-Retrieval Passages (DPR)

thể và nút đoạn văn.

Với k bộ ba được chọn ở bước 2, hệ thống sẽ trích xuất các thực thể liên quan, sau đó ánh xạ chúng đến các nút thực thể (thông qua hàm băm MD5) trong đồ thị. Mỗi nút này sẽ được gán trọng số khởi tạo là độ tương đồng cosine được tính toán ở bước 2. Hệ thống sẽ lấy 5 thực thể có điểm cao nhất để làm nút hạt giống.

Với các nút đoạn văn, tất cả đều được đặt làm nút hạt giống. Lý giải cho điều này, các tác giả trình bày rằng dựa trên quan sát rằng việc kích hoạt một tập hợp rộng hơn các đoạn văn tiềm năng sẽ hiệu quả hơn cho việc khám phá các đoạn văn dọc theo chuỗi lý luận đa bước, thay vì chỉ tập trung vào các đoạn văn được xếp hạng cao nhất. Mặt khác, để tính điểm cho các nút đoạn văn này, hệ thống sử dụng điểm tương đồng từ giai đoạn DPR để xác định trọng số ban đầu. Các điểm DPR này sẽ được chuẩn hóa bằng cách nhân với một hệ số mặc định là 0.05, đảm bảo sự cân bằng về điểm số giữa các nút đoạn văn trong đồ thị.

Bước 4: Duyệt đồ thị sử dụng thuật toán PPR

Sau khi xác định các nút hạt giống và trọng số tương ứng, hệ thống tiến hành sử dụng thuật toán PPR để "duyệt đồ thị". Như đã trình bày trong công thức (3.17), PPR sử dụng một vector phân phối $\mathbf{v}$, được xây dựng từ các trọng số của tất cả các nút có trong đồ thị. Hay nói một cách khác, giả sử có n nút thực thể và m nút đoạn văn. Gọi b là tổng trọng số của tất cả các nút trong đồ thị:

$$b = \sum_{i=1}^{n} \text{trọng số nút thực thể}_i + \sum_{j=1}^{m} \text{trọng số nút đoạn văn}_j.$$

Khi này, vector $\mathbf{v}$ sẽ là:

$$\mathbf{v} = [\frac{a_1}{b}, ..., \frac{a_N}{b}],$$

với:

- a là trọng số của nút,
- N là tổng số nút có trong đồ thị ($N = n + m$).

Từ đây, vector $\mathbf{v}$ sẽ được kết hợp với hệ số tắt dần $\alpha = 0.5$ để tính được vector $\mathbf{r} = (r_1, r_2, ..., r_N)$ với mỗi phần tử trong vector r đại diện cho điểm số PPR cho mỗi nút trong đồ thị.

Bước 5: Tạo sinh câu trả lời

Sau khi hoàn thành việc tính toán điểm số PPR, 5 nút đoạn văn với điểm số cao nhất, sẽ được ánh xạ qua hàm băm MD5 để biến đổi lại thành các nội dung tài liệu văn bản. Kết hợp chúng cùng với câu chỉ thị và câu truy vấn Q, làm dữ liệu đầu vào cho LLM để tạo sinh câu trả lời.

4.2.5 HippoRAG-2 không tích hợp đồ thị

Trong quá trình triển khai và đánh giá hiệu quả của hệ thống RAG trong bối cảnh dữ liệu giáo dục, chúng tôi lựa chọn hai mô hình: HippoRAG-2 và một phiên bản tùy chỉnh mang tên SimpleRAG. Cả hai đều được nhóm tác giả của HippoRAG-2 triển khai đồng nhất, nhằm đảm bảo tính công bằng trong các phép đo so sánh. Tác giả xây dựng SimpleRAG dựa trên HippoRAG-2, nhưng lược bỏ hoàn toàn giai đoạn trích xuất thực thể, bộ ba và xây dựng đồ thị tri thức. Hay nói một cách khác, mô hình SimpleRAG này dựa trên kiến trúc tổng thể của một mô hình RAG cơ bản được trình bày ở Phần (3.1.1). SimpleRAG đóng vai trò như một mô hình tiêu chuẩn để kiểm chứng vai trò của đồ thị tri thức trong một hệ thống RAG.

Ở giai đoạn truy xuất trực tuyến, SimpleRAG chỉ đơn thuần tạo vector biểu diễn cho câu truy vấn và tiến hành phép so sánh cosine giữa truy vấn và các đoạn văn được lưu trữ trong không gian vector. Toàn bộ quá trình truy xuất được rút gọn và không có mở rộng ngữ cảnh dựa trên quan hệ ngữ nghĩa.

Về mặt lưu trữ, HippoRAG duy trì đồng thời ba "kho" nhúng độc lập, tương ứng với đoạn văn, thực thể và bộ ba. Trong khi đó, SimpleRAG chỉ duy trì một kho duy nhất cho đoạn văn, không có khái niệm về thực thể hay bộ ba trong quá trình lập chỉ mục.

Thông qua việc đặt hai hệ thống trên cùng một nền tảng để đem ra đánh giá, chúng tôi hướng đến việc cung cấp góc nhìn toàn diện về vai trò của đồ thị tri thức trong các hệ thống RAG hiện đại, cũng như ảnh hưởng của chúng đối với chất lượng và độ chính xác của kết quả cuối cùng. Chúng tôi sẽ trình bày kết quả so sánh ở chương 5 tiếp theo.

Chương 5

Thực nghiệm và đánh giá

Trong phần này, độc giả sẽ nắm được các kết quả thực nghiệm so sánh giữa hai hệ thống: RAG cơ bản (Simple RAG) và RAG tích hợp đồ thị (HippoRAG-2, chúng tôi sẽ gọi là Graph-based RAG), lần lượt sử dụng hai mô hình ngôn ngữ lớn khác nhau là: GPT 4-o mini và Llama-3.1-8B. Bên cạnh đó, chương này sẽ mở đầu với phần trình bày về tập dữ liệu kiểm tra được thiết kế để đánh giá hai hệ thống trên.

5.1 Tập dữ liệu kiểm tra

Dựa trên phạm vi đề tài và các phương thức đánh giá đã được trình bày ở những chương trước, chúng tôi đã xây dựng và sử dụng ba bộ dữ liệu câu hỏi thực nghiệm cho ba ngành học khác nhau, bao gồm: DS, AM, và MCS. Mỗi ngành đều được xây dựng với ba bộ câu hỏi riêng biệt: closed_end, opened_end và multihop, với cấu trúc định dạng thống nhất theo JSON như sau:

Trường `id` là định danh duy nhất cho mỗi câu hỏi, đảm bảo khả năng theo dõi và đối chiếu. Trường `question` chứa nội dung câu hỏi truy vấn, `answer` là một danh sách chuỗi các đáp án/câu trả lời tham chiếu có thể chấp nhận được, hỗ trợ việc đánh giá tính chính xác của hệ thống. Trường `answerable` là một giá trị boolean thể hiện khả năng nội dung câu trả lời có nằm trong các tài liệu đầu vào hay không (True nếu có và False nếu không). Cuối cùng, trường `paragraphs` là một mảng các đoạn văn tham chiếu, trong đó mỗi đoạn gồm `title` là tiêu đề đoạn văn tham chiếu. Trường `is_supporting` là kiểu boolean cho biết đoạn đó có thực sự liên quan đến câu trả lời hay không (True nếu có và False nếu không). Ngoài ra, chúng tôi có tạo thêm trường `case_std_id` là kiểu số nguyên, nhằm xác định loại câu hỏi nằm ở tình huống nào đã được đặt ra trong Bảng 1.1.

Để đánh giá toàn diện khả năng truy xuất tri thức, lưu giữ thông tin chính xác, cũng như năng lực liên kết và lý giải của hệ thống RAG trong bối cảnh giáo dục, chúng tôi xây dựng ba bộ dữ liệu tương ứng với ba dạng thách thức đặc trưng trong lĩnh vực hỏi đáp tự động. Cụ thể, hai bộ dữ liệu đầu tiên là *closed_ end* và *opened_ end* đại diện cho nhóm các câu hỏi đơn giản, tập trung đánh giá khả năng ghi nhớ và truy xuất tri thức thực tế một cách chính xác từ một nguồn thông tin duy nhất (Wang et al., 2024, Mallen

et al., 2022); vì vậy chúng tôi sẽ áp dụng những tình huống nghiên cứu số 1,2 và 3. Trong khi đó, bộ *multihop* phản ánh mức độ phức tạp cao hơn khi yêu cầu hệ thống phải kết nối và tổng hợp nhiều mảnh thông tin nằm rải rác ở nhiều học phần khác nhau để hình thành câu trả lời cuối cùng (Yuan et al., 2024, Yang et al., 2018). Chúng tôi sẽ áp dụng những tính huống nghiên cứu còn lại từ số 4 đến 10 nhằm kiểm tra khả năng liên kết thông tin và năng lực lý giải của mô hình.

Các trường thông tin trong dữ liệu thực nghiệm

```
[
    {
    "id": "question_1",
    "question": "How many content topics are there in the Deep
    Learning for Data Science course?",
    "answer": [
      "14", "fourteen"
    ],
    "answerable": true,
    "paragraphs": [
      {
        "title": "MTH10622",
        "text": "Deep Learning for Data Science (ID: MTH10622) is
        a Elective subject conducted during the even semester...",
        "is_supporting": true,
        "idx": 0
      }
    ],
    "case_std_id": 3
  },
  {...}, {...}
]
```

Các bộ câu hỏi được xây dựng trải đều trên toàn bộ chương trình đào tạo của từng ngành, nhằm đảm bảo tính đại diện và khả năng khái quát trong đánh giá hệ thống. Bảng 5.1 cho thấy tổng số lượng câu hỏi cho từng tập kiểm tra ở từng ngành. Ngoài ra, chúng tôi cũng sẽ trình bày về quy trình soạn bộ dữ liệu kiểm tra cũng như chi tiết về số lượng câu hỏi cho mỗi tình huống nghiên cứu ở Phần Phụ lục B. Thông tin về bộ dữ liệu kiểm tra.

Ngành	Tổng số học phần	closed_end	opened_end	multihop
DS	75	40	40	30
MCS	90	50	50	30
AM	94	52	52	30

Bảng 5.1: Tổng số lượng câu hỏi cho từng tập kiểm tra theo từng ngành học

5.1.1 Bộ kiểm tra closed_end

Mỗi câu hỏi trong bộ này chỉ liên quan đến một đơn vị tri thức duy nhất (một học phần), do đó mô hình truy vấn chỉ cần tìm đúng một đoạn văn hỗ trợ là có thể đưa ra câu trả lời chính xác. Dạng câu hỏi thường xoay quanh các thông tin cụ thể như: điều kiện tiên quyết của một học phần, học phần thuộc chuyên ngành nào, hoặc một học phần có phải là bắt buộc hay không. Ví dụ như *"Is it true that the content of the Psychology course contains 5 parts?"*.

Đặc trưng của closed_end là yêu cầu câu trả lời phải ở dạng ngắn gọn, thường là yes/no (đúng/sai), một danh từ riêng hoặc một cụm danh từ được trích xuất trực tiếp từ văn bản truy xuất (số lượng từ tối đa trong câu trả lời tham chiếu là 3). Điều này giúp tập trung đánh giá khả năng tìm kiếm chính xác thông tin định danh và tránh gây nhiễu bởi các yếu tố ngôn ngữ học không cần thiết.

5.1.2 Bộ kiểm tra opened_end

Tương tự closed_end về cấu trúc truy xuất, bộ dữ liệu opened_end cũng chỉ yêu cầu tìm kiếm thông tin từ một học phần duy nhất. Tuy nhiên, điểm khác biệt nằm ở dạng câu trả lời là thay vì yêu cầu trả lời bằng từ đơn hoặc cụm danh từ, thì hệ thống phải tạo sinh ra một câu hoàn chỉnh, có đầy đủ chủ ngữ, vị ngữ và giữ được tính mạch lạc về ngữ pháp và ngữ nghĩa. Ví dụ về dạng câu hỏi cho bộ kiểm tra này là: *"What is the first content topic in Earth Science?"* hoặc *"How many topics are the in the content of Labwork on General Biology 2?"*.

Bộ dữ liệu này đặc biệt hữu ích trong việc đánh giá khả năng kết hợp giữa truy xuất và tạo sinh ngôn ngữ tự nhiên. Nó cũng phản ánh đúng nhu cầu thực tế trong môi trường giáo dục, nơi người học thường mong đợi một câu trả lời đầy đủ và có tính diễn giải cao từ hệ thống tư vấn hỏi-đáp học phần.

5.1.3 Bộ kiểm tra multihop

Bộ kiểm tra multihop được xây dựng nhằm mô phỏng các tình huống truy vấn phức tạp: các thông tin được truy xuất không nằm gọn trong một học phần duy nhất, mà được phân tán ở nhiều đơn vị tri thức khác nhau (Yang et al., 2018). Do đó, hệ thống phải thực hiện lý luận nhiều bước – tức là kết nối, tổng hợp và suy luận từ nhiều đoạn văn để đưa ra câu trả lời cuối cùng.

Các câu hỏi trong multihop thường mang tính chất phân tích hoặc lập luận, ví dụ như: *"What is the category and offering semester for Introduction to Data Science, Data Mining, and Introduction to Machine Learning?"* hoặc *"For Compulsory courses available in the odd semester with no prerequisites, what common academic institution teaches Marxist-Leninist Philosophy, Marxist-Leninist Political Economy, and English 1?"*. Để trả lời đúng, hệ thống phải lần theo chuỗi điều kiện tiên quyết, nhận diện mối quan hệ giữa các học phần, từ đó tổng hợp thông tin từ hai hoặc nhiều đoạn văn (tối đa là 5) có liên quan. Vì vậy, trường `paragraphs` trong bộ multihop thường chứa từ 2 đến 5 đoạn văn tham chiếu, sẽ được đánh chỉ mục `idx` riêng biệt. Dạng câu trả lời tham chiếu trong

bộ này là câu hoàn chỉnh, với các danh từ và cụm danh từ được lấy chính xác từ nội dung của các đoạn văn tham chiếu.

5.2 Mô tả thực nghiệm và Đánh giá kết quả

Trong phần này, chúng tôi sẽ chia ra thành 2 nhóm thực nghiệm sử dụng hai LLM khác nhau lần lượt là: (i) sử dụng mô hình ngôn ngữ GPT-4o mini; (ii) sử dụng mô hình ngôn ngữ Llama-3.1-8B. Chúng tôi sẽ đánh giá và so sánh trên hai phương diện là: truy xuất và khả năng tạo sinh câu trả lời. Ngoài ra, vì sự giới hạn thời gian cũng như tài chính, chúng tôi chỉ sử dụng Llama-3.1-8B cùng với mô hình tiêu chuẩn Contriever (mà không sử dụng các mô hình nhúng hiện đại khác) trong luận văn này. Các kết quả từ nhóm (ii) này vẫn sẽ cung cấp những góc nhìn và đánh giá khách quan hơn cho cả hai mô hình Simple RAG và Graph-based RAG.

5.2.1 Kỹ thuật ra chỉ thị

Như đã đề cập ở Phần 3.5, kỹ thuật ra chỉ thị giữ vai trò then chốt trong một hệ thống RAG. Trong phần này, chúng tôi thực hiện sẽ trình bày cú pháp của các chỉ thị được dùng trong dự án cho ba nhiệm vụ ở cả hai mô hình ngôn ngữ là GPT-4o mini và Llama-3.1-8B:

1. Trích xuất thực thể định danh sử dụng few-shot learning.
2. Xây dựng mối quan hệ giữa các thực thể định danh đã được trích xuất sử dụng zero-shot learning.
3. Tạo sinh câu trả lời ngôn ngữ tự nhiên dựa trên các tài liệu và thực thể định danh được xác định có liên quan đến câu hỏi sử dụng zero-shot learning.
4. Dùng AI để đánh giá khả năng tạo sinh ngôn ngữ.

Chỉ thị của HippoRAG để trích xuất thực thể cho quá trình lập chỉ mục

Your task is to extract named entities from the given paragraph. Respond with a JSON list of entities. For example, the input text is:
"Elementary Financial Mathematics (ID: MTH10201) is a Compulsory subject conducted during the odd semester. The course is instructed by Phan Thi Phuong, and it It requires prior completion of Calculus 1A, Calculus 2A. Upon completion, students will achieve: Equip students with the basic knowledge of finance and financial mathematics for discrete non-random models. The course includes: Including the theory of interest rates, money chains, forms of borrowing, appraisal of investment projects, valuation of bonds and stocks".
The output should be a JSON of:
{"named_entities": ["Elementary Financial Mathematics", "MTH10201", "Compulsory subject", "odd semester", "Phan Thi Phuong", "Calculus 1A", "Calculus 2A", "basic knowledge of finance and financial mathematics for discrete non-random models"] }

Chỉ thị của HippoRAG để xây dựng mối quan hệ giữa các thực thể định danh

Your task is to construct an RDF (Resource Description Framework) graph from the given passages and named entity lists. Respond with a JSON list of triples, with each triple representing a relationship in the RDF graph.
Pay attention to the following requirements:

- Each triple should contain at least one, but preferably two, of the named entities in the list for each passage.
- Clearly resolve pronouns to their specific names to maintain clarity.

Chỉ thị của HippoRAG để tạo sinh câu trả lời ngôn ngữ tự nhiên

As an advanced reading comprehension assistant, your task is to analyze text passages and corresponding questions meticulously.
Your response start after "Thought: ", where you will methodically break down the reasoning process, illustrating how you arrive at conclusions.
Conclude with "Answer: " to present a concise, definitive response, devoid of additional elaborations, and keep the same writing style as the found source text if the question is a opened-end question else you can answer just Yes or No.

Chỉ thị dùng AI để đánh giá khả năng tạo sinh ngôn ngữ

You are a judge who evaluates the quality of answers to questions. You should provide a score in the scale of 1 to 10 (without explanation), where 1 is the predicted answer is completely irrelevant to the ground truth answer or the question, and 10 is the predicted answer is exactly the same as the ground truth answer in the semantic meaning.

5.2.2 Sử dụng mô hình ngôn ngữ GPT-4o mini

Kết quả đánh giá hiệu suất truy xuất thông tin

Để đánh giá hiệu suất truy xuất thông tin một cách toàn diện, chúng tôi sử dụng hệ số Recall với ba mức top-k: $k \in \{1, 2, 5\}$ (đã được giới thiệu ở Phần 3.6). Bên cạnh đó, chúng tôi cũng muốn lưu ý rằng chỉ số $R@1$ đánh giá khả năng hệ thống truy xuất đúng tài liệu liên quan nhất ở vị trí đầu tiên. Với các truy vấn đơn, chỉ số này phản ánh trực tiếp độ chính xác của mô hình. Tuy nhiên, trên tập Multihop – nơi mỗi truy vấn yêu cầu tổng hợp từ 2 đến 5 tài liệu – điểm số $R@1$ thường thấp (nhiều trường hợp dưới 50), do hệ thống chỉ được xét duy nhất một kết quả đầu tiên để so sánh với toàn bộ tập tài liệu đúng. Điều này khiến nhiều câu truy vấn dù có tài liệu liên quan nằm ở vị trí thứ hai hoặc ba cũng không được tính điểm. Vì vậy, các chỉ số như $R@2$ hoặc $R@5$ được dùng để phản ánh toàn diện hiệu quả truy xuất trong các bài toán đa bước.

Ngoài ra, Bảng 5.2, 5.3 và 5.4, sẽ là các kết quả truy xuất tài liệu ở từng ngành lần lượt là AM, DS và MCS sử dụng GPT-4o mini ở giai đoạn trích xuất thực thể và bộ ba và các mô hình nhúng khác nhau.

Bảng 5.2: So sánh hiệu suất truy xuất thông tin dùng hệ số Recall@top-k (R@k) cho bộ dữ liệu ngành **AM** sử dụng GPT-4o mini ở giai đoạn trích xuất thực thể và bộ ba. Kết quả của phương pháp vượt trội hơn được **in đậm**.

	Closed-end QA			Opened-end QA			Multi-hop QA			Average		
	R@1	**R@2**	**R@5**	**R@1**	**R@2**	**R@5**	**R@1**	**R@2**	**R@5**	**R@1**	**R@2**	**R@5**
Simple RAG												
BM25	36.5	38.5	48.1	44.2	55.8	63.5	26.7	40.0	60.0	35.8	44.7	57.2
Contriever	42.3	46.2	59.6	38.5	48.1	55.8	27.2	41.1	60.0	36.0	45.1	58.5
GritLM-7B	**59.6**	59.6	**63.5**	65.4	**69.2**	**69.2**	33.3	66.1	**78.9**	52.8	65.0	**70.5**
NV-Embed-v2	**59.6**	**63.5**	**63.5**	**67.3**	**69.2**	**69.2**	**37.8**	**71.1**	**78.9**	**54.9**	**67.9**	**70.5**
text-embedding-3-small	51.9	61.5	**63.5**	63.5	**69.2**	**69.2**	31.1	61.7	77.8	48.8	64.1	70.2
Graph-based RAG												
Contriever	53.8	59.6	**63.5**	61.5	63.5	65.4	33.9	53.9	73.3	49.8	59.0	67.4
GritLM-7B	55.8	**63.5**	**63.5**	63.5	65.4	**69.2**	32.8	60.0	80.0	50.7	62.9	**70.9**
NV-Embed-v2	53.8	59.6	**63.5**	**67.3**	**67.3**	**69.2**	**34.4**	**65.0**	**80.0**	**51.9**	**64.0**	**70.9**
text-embedding-3-small	**57.7**	61.5	61.5	65.4	65.4	**69.2**	29.4	61.7	78.9	50.8	62.9	69.9

Kết quả trong Bảng 5.2 cho thấy **NV-Embed-v2** là phương pháp đạt hiệu suất truy vấn cao nhất trong tổng thể, với điểm trung bình $R@1 = 54.9$, vượt trội so với phương pháp truyền thống BM25 ($R@1 = 35.8$) trong tập hỏi-đáp của ngành **AM**. Kết quả này thể hiện rõ ưu thế của các mô hình nhúng hiện đại so với phương pháp dựa trên tần suất từ khóa. Trong ba bộ kiểm tra, Multi-hop cho kết quả tốt nhất trên hầu hết các mô hình, đặc biệt là ở $R@5$. Điều này cho thấy khả năng truy xuất trong bối cảnh yêu cầu suy luận đa bước thì cả Simple RAG và Graph-based RAG đều có kết quả xấp xỉ 80. Việc tích hợp RAG với đồ thị giúp cải thiện đáng kể hiệu suất đối với một số mô hình, điển hình là Contriever với mức tăng rõ rệt từ $R@1 = 36.0$ lên 49.8. Tuy nhiên, đối với các mô hình nhúng mạnh mẽ như GritLM-7B và NV-Embed-v2, sự cải thiện là tương đối nhỏ, cho thấy rằng bản thân các mô hình này đã có khả năng truy xuất ngữ nghĩa tốt mà không cần quá nhiều hỗ trợ từ cấu trúc đồ thị.

Ngược lại, kết quả trong Bảng 5.3 cho thấy hiệu suất truy vấn ở ngành **DS** nhìn chung cao hơn đáng kể so với ngành **AM** (Bảng 5.2), phản ánh mức độ nhất quán và dễ truy xuất hơn trong bộ dữ liệu này. Trong các phương pháp, **NV-Embed-v2** tiếp tục đạt hiệu suất cao nhất ở mức $R@1 = 71.4$, ngang bằng với text-embedding-3-small, và cao hơn đáng kể so với phương pháp truyền thống BM25 ($R@1 = 44.4$). Đối với từng loại câu hỏi, tập Closed-end đạt kết quả gần như tuyệt đối ở $R@5$ (nhiều mô hình đạt 92.5), cho thấy khả năng truy xuất rất chính xác đối với loại câu hỏi có phạm vi thông tin rõ ràng. Trong khi đó, hai tập kiểm tra còn lại là Opened-end và Multi-hop đều cho kết quả $R@5$ thấp hơn xấp xỉ 2.5 so với kết quả $R@5$ ở ngành **AM**. Việc áp dụng Graph-based RAG tiếp tục đem lại hiệu quả đáng kể cho những mô hình yếu hơn, đặc biệt là Contriever, với mức tăng $R@1$ từ 39.7 lên 51.4. Đối với các mô hình nhúng mạnh như GritLM-7B, NV-Embed-v2, và text-embedding-3-small, mức cải thiện tuy không lớn nhưng vẫn giúp đạt đến hiệu suất truy xuất gần tối đa, với $R@5 = 78.5$–78.8 và $R@1$ dao động từ 70.5–71.4. Tóm lại, đối với bộ câu hỏi cho ngành **DS**, chúng tôi nhận thấy việc lựa chọn mô hình nhúng chất lượng cao kết hợp với Simple RAG giúp tối ưu hóa chi

Bảng 5.3: So sánh hiệu suất truy xuất thông tin dùng hệ số Recall@top-k (R@k) cho bộ dữ liệu ngành **DS** sử dụng GPT-4o mini ở giai đoạn trích xuất thực thể và bộ ba. Kết quả của phương pháp vượt trội hơn được **in đậm**.

	Closed-end QA			Opened-end QA			Multi-hop QA			Average		
	R@1	**R@2**	**R@5**	**R@1**	**R@2**	**R@5**	**R@1**	**R@2**	**R@5**	**R@1**	**R@2**	**R@5**
Simple RAG												
BM25	67.5	75.0	77.5	42.5	50.0	57.5	23.3	43.3	53.3	44.4	56.1	62.8
Contriever	55.0	62.5	75.0	42.5	52.5	62.5	21.7	35.3	60.3	39.7	50.1	65.9
GritLM-7B	**85.0**	87.5	**92.5**	57.5	**67.5**	**67.5**	**30.6**	57.2	73.3	57.7	70.7	77.8
NV-Embed-v2	82.5	**90.0**	**92.5**	60.0	65.0	**67.5**	28.3	55.0	**75.6**	56.9	70.0	**78.5**
text-embedding-3-small	82.5	87.5	**92.5**	**65.0**	**67.5**	**67.5**	28.3	**57.8**	**75.6**	**58.6**	**70.9**	**78.5**
Graph-based RAG												
Contriever	70.0	85.0	**92.5**	62.5	**67.5**	**67.5**	26.7	47.8	69.4	53.1	66.8	76.5
GritLM-7B	**80.0**	**87.5**	**92.5**	**65.0**	**67.5**	**67.5**	**32.8**	60.0	74.4	**59.3**	71.7	78.1
NV-Embed-v2	77.5	**87.5**	**92.5**	60.0	62.5	**67.5**	**32.8**	57.8	**75.6**	56.8	69.3	**78.5**
text-embedding-3-small	77.5	**87.5**	**92.5**	57.5	65.0	**67.5**	29.4	**61.7**	**75.6**	54.8	**71.4**	**78.5**

phí và tài nguyên trong hệ thống.

Kết quả trong Bảng 5.4 cho thấy hiệu suất truy vấn ở ngành **MCS** đạt mức cao nhất trong cả ba ngành được khảo sát, với các mô hình nhúng hiện đại đều đạt giá trị $R@5$ tiệm cận hoặc bằng 100.0 ở các nhiệm vụ đơn lẻ. Trong đó, G**ritLM-7B** nổi bật với hiệu suất trung bình $R@1 = 71.1$, $R@2 = 81.4$, và $R@5 = 886$. trong Simple RAG, khẳng định khả năng truy xuất vượt trội và ổn định của mô hình này trong bộ dữ liệu **MCS**. Bên cạnh đó, cả NV-Embed-v2 và text-embedding-3-small cũng đạt hiệu suất rất cao với $R@5 = 87.5$–86.9, cho thấy mức độ bền vững và hiệu quả của các mô hình nhúng trong việc xử lý các câu hỏi đóng, mở và suy luận nhiều bước. Đối với từng loại nhiệm vụ, Opened-end tiếp tục duy trì hiệu suất tốt nhất (với $R@1$ lên đến 96.0).Tuy vậy, so với hai ngành trước (**AM, DS**), hiệu suất ở Multi-hop trong **MCS** có xu hướng cao hơn rõ rệt, đặc biệt khi sử dụng GritLM-7B ($R@5 = 89.7$) và NV-Embed-v2 ($R@5 = 87.5$). Điều này cho thấy dữ liệu thuộc ngành **MCS** có tính nhất quán và gợi ý ngữ nghĩa mạnh hơn, hỗ trợ tốt hơn cho các mô hình suy luận. Tóm lại, việc áp dụng Graph-based RAG không mang lại cải thiện đáng kể đối với Simple RAG cùng với các mô hình nhúng mạnh.

Kết quả đánh giá hiệu suất tạo sinh ngôn ngữ

Dựa trên kết quả truy xuất thông tin, việc đánh giá bước tạo sinh ngôn ngữ chỉ được thực hiện trên các phương pháp có hiệu suất cao nhất là hoàn toàn hợp lý. Cụ thể, chỉ **GritLM-7B** và **NV-Embed-v2** được chọn để đánh giá tạo sinh do chúng thống trị về hiệu suất truy vấn với R@1 trung bình từ 48.8-71.1. Việc loại bỏ BM25, Contriever, và text-embedding-3-small khỏi giai đoạn tạo sinh là cần thiết vì hiệu suất truy vấn thấp của chúng (R@1 = 35.8-49.8) sẽ tạo ra "hiệu ứng cổ chai[1]". Bên cạnh đó, chúng tôi sẽ sử

[1]Bottleneck effect: Trong ngữ cảnh này, chúng tôi muốn ám chỉ hiện tượng khi chất lượng truy xuất kém (do lựa chọn phương pháp truy xuất không hiệu quả) trở thành điểm nghẽn, làm hạn chế nghiêm trọng chất lượng đầu ra của mô hình tạo sinh, bất kể năng lực ngôn ngữ của mô hình có cao đến đâu.

Bảng 5.4: So sánh hiệu suất truy xuất thông tin dùng hệ số Recall@top-k (R@k) cho bộ dữ liệu ngành **MCS** sử dụng GPT-4o mini ở giai đoạn trích xuất thực thể và bộ ba. Kết quả của phương pháp vượt trội hơn được **in đậm**.

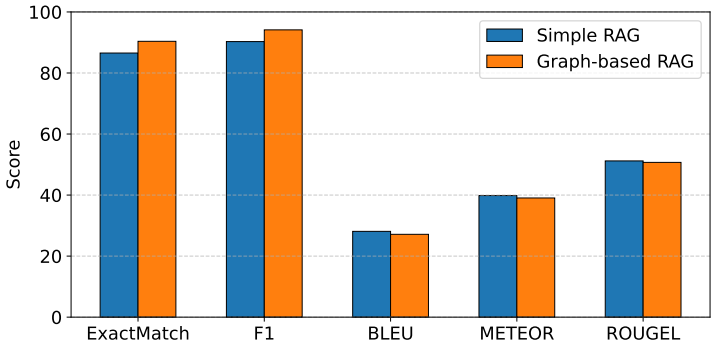
	Closed-end QA			Opened-end QA			Multi-hop QA			Average		
	R@1	**R@2**	**R@5**	**R@1**	**R@2**	**R@5**	**R@1**	**R@2**	**R@5**	**R@1**	**R@2**	**R@5**
Simple RAG												
BM25	40.0	60.0	**80.0**	78.0	84.0	84.0	20.0	30.0	63.3	46.0	58.0	75.8
Contriever	38.0	46.0	68.0	80.0	88.0	92.0	17.8	37.8	62.2	45.3	57.3	74.1
GritLM-7B	**80.0**	**80.0**	**80.0**	**96.0**	**96.0**	**96.0**	**37.2**	**68.1**	**89.7**	**71.1**	**81.4**	**88.6**
NV-Embed-v2	78.0	**80.0**	**80.0**	94.0	**96.0**	**96.0**	35.6	64.2	86.4	69.2	80.1	87.5
text-embedding-3-small	72.0	78.0	**80.0**	94.0	**96.0**	**96.0**	28.1	47.8	84.7	64.7	73.9	86.9
Graph-based RAG												
Contriever	**74.0**	76.0	**80.0**	80.0	92.0	94.0	30.3	54.2	80.0	61.4	74.1	84.7
GritLM-7B	70.0	**80.0**	**80.0**	88.0	**96.0**	**96.0**	**33.3**	**63.6**	**86.4**	63.8	**79.9**	**87.5**
NV-Embed-v2	**74.0**	**80.0**	**80.0**	**92.0**	94.0	**96.0**	30.6	59.2	85.3	**65.5**	77.7	87.1
text-embedding-3-small	66.0	76.0	**80.0**	90.0	**96.0**	**96.0**	31.9	54.4	85.6	62.6	75.5	87.2

dụng toàn bộ câu hỏi trong cả hai tập Opened-end và Closed-end theo từng ngành để trực quan kết quả ở Hình 5.1, 5.2 và 5.3. Tập dữ liệu Multihop sẽ được trực quan ở Hình 5.4.

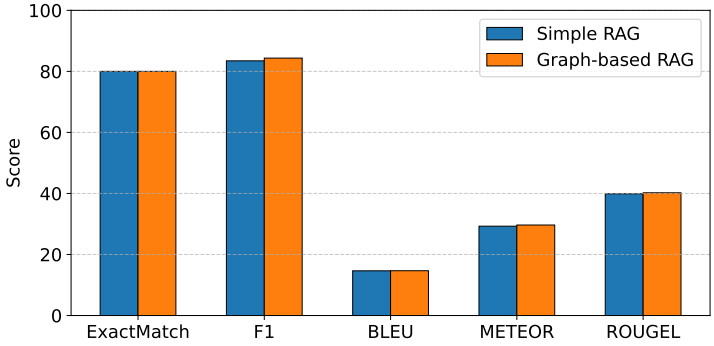
Theo mô tả trong phần hệ số đánh giá và tập dữ liệu kiểm tra, hiệu suất tạo sinh ngôn ngữ được đo lường thông qua các chỉ số khác nhau tùy theo loại nhiệm vụ: nhiệm vụ đóng sử dụng ExactMatch và điểm F1, trong khi nhiệm vụ tham chiếu áp dụng ba chỉ số BLEU@4, METEOR, và ROUGE-L. Tương tự với đánh giá truy vấn thông tin, kết quả tạo sinh cũng được phân tích theo từng ngành để cung cấp góc nhìn chi tiết về hiệu suất chuyên biệt. Điều quan trọng cần nhấn mạnh là chất lượng tạo sinh phụ thuộc trực tiếp vào hiệu suất truy xuất thông tin, do đó kết quả đánh giá mô-đun tạo sinh chủ yếu phản ánh hiệu suất tổng thể cuối cùng của toàn bộ hệ thống RAG.

Biểu đồ cột ở Hình 5.1 cho thấy sự khác biệt đáng kể giữa hai nhóm chỉ số đánh giá và hiệu quả của hai mô hình RAG trong bối cảnh ngành AM. Ở nhóm chỉ số đánh giá trên tập kiểm tra nhiệm vụ đóng, cả ExactMatch và F1 đều đạt kết quả cao (trên 80 điểm) ở cả hai phương pháp, trong đó Graph-based RAG nhỉnh hơn nhẹ, phản ánh khả năng tạo sinh ngôn ngữ chính xác khi hệ thống được cung cấp đầu vào ngắn gọn, trực tiếp. Ngược lại, với nhóm chỉ số đánh giá dựa trên tham chiếu gồm BLEU, METEOR và ROUGE-L, cả hai phương pháp đều cho điểm số khá thấp (dưới 50), đặc biệt BLEU chỉ quanh mức 30, cho thấy việc mô hình hóa ngôn ngữ trong các tình huống phức tạp hơn như câu hỏi mở vẫn còn nhiều thách thức. Điểm đáng chú ý là sự khác biệt giữa Simple RAG và Graph-based RAG không thực sự rõ rệt ở tất cả các chỉ số, dù Graph-based RAG có cải thiện nhẹ ở ExactMatch và F1. Điều này cho thấy rằng việc bổ sung đồ thị, cũng như các phương pháp trích xuất thực thể trong hệ thống RAG không mang lại hiệu quả vượt trội đáng kể cho mô hình ngôn ngữ tạo sinh câu trả lời chính xác, so với sử

Nói cách khác, nếu thông tin truy xuất không đủ liên quan hoặc chính xác, thì mô hình tạo sinh cũng khó có thể tạo ra câu trả lời đúng hoặc hữu ích.



Hình 5.1: Kết quả đánh giá mô-đun tạo sinh ngôn ngữ với mô hình nhúng **NV-Embed-v2** trên hai tập dữ liệu kiểm tra Closed-end và Opened-end của **AM**. Hai hệ số đầu tiên ExactMatch và F1 dùng để đánh giá tập kiểm tra nhiệm vụ đóng, trong khi đó ba hệ số còn lại gồm BLEU, METEOR, và ROUGEL dùng để đánh giá dựa trên tham chiếu.

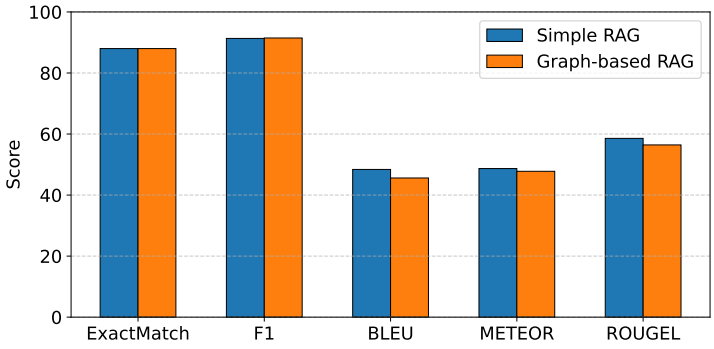


Hình 5.2: Kết quả đánh giá mô-đun tạo sinh ngôn ngữ với mô hình nhúng **GritLM-7B** trên hai tập dữ liệu kiểm tra Closed-end và Opened-end của **DS**. Hai hệ số đầu tiên ExactMatch và F1 dùng để đánh giá tập kiểm tra nhiệm vụ đóng, trong khi đó ba hệ số còn lại gồm BLEU, METEOR, và ROUGEL dùng để đánh giá dựa trên tham chiếu.

dụng một mô hình Simple RAG cùng với mô hình nhúng mạnh mẽ trong mô-đun truy xuất.

Hình 5.2 minh họa kết quả đánh giá mô-đun tạo sinh ngôn ngữ cho ngành **DS** khi sử dụng mô hình **GritLM-7B**, phương pháp có hiệu suất truy vấn tốt nhất trong Bảng 5.3. Tương tự **AM**, hai chỉ số đo lường trên tập kiểm tra nhiệm vụ đóng là ExactMatch và F1 tiếp tục đạt kết quả cao (khoảng 80–85 điểm), cho thấy khả năng tạo sinh chính xác và phù hợp khi hệ thống được cung cấp thông tin rõ ràng. Trong khi đó, ba chỉ số đánh giá dựa trên tham chiếu (BLEU, METEOR, ROUGE-L) vẫn duy trì ở mức thấp, đặc biệt BLEU chỉ đạt xấp xỉ 30 điểm, phản ánh những khó khăn trong việc tạo ra văn bản có tính linh hoạt ngôn ngữ và tương đồng ngữ nghĩa với câu trả lời tham chiếu. Sự khác biệt giữa Simple RAG và Graph-based RAG ở biểu đồ này gần như không đáng kể trên toàn bộ các chỉ số.

Một quan sát tương tự về hiệu suất giữa hai mô hình RAG cũng được thể hiện ở tập dữ liệu ngành **MCS**. Bảng 5.3 minh họa kết quả đánh giá mô-đun tạo sinh ngôn ngữ cho ngành **MCS** khi sử dụng mô hình **GritLM-7B**, phương pháp có hiệu suất truy vấn cao nhất theo bảng 5.4. Kết quả cho thấy cả hai mô hình Simple RAG và Graph-based



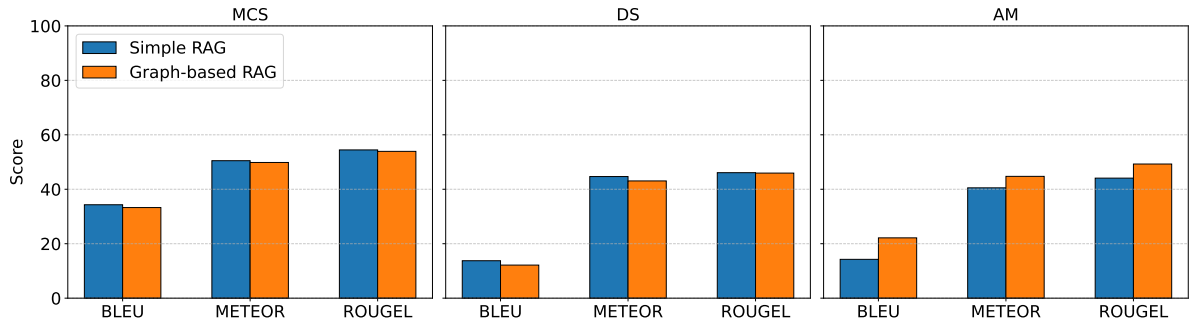
Hình 5.3: Kết quả đánh giá mô-đun tạo sinh ngôn ngữ với mô hình nhúng **GritLM-7B** trên hai tập dữ liệu kiểm tra Closed-end và Opened-end của **MCS**. Hai hệ số đầu tiên ExactMatch và F1 dùng để đánh giá tập kiểm tra nhiệm vụ đóng, trong khi đó ba hệ số còn lại gồm BLEU, METEOR, và ROUGEL dùng để đánh giá dựa trên tham chiếu.

RAG đều đạt điểm số cao ở hai chỉ số ExactMatch và F1 (xấp xỉ 90 điểm), phản ánh độ chính xác cao khi tạo sinh câu trả lời khớp với nhãn mục tiêu trong các câu hỏi có phạm vi đóng. Không giống hai ngành trước, hệ thống tại đây cũng thể hiện hiệu suất tương đối tốt ở nhóm chỉ số đánh giá dựa trên tham chiếu: cụ thể BLEU, METEOR, và ROUGE-L đều đạt mức điểm trên 45, cho thấy khả năng tạo sinh ngữ nghĩa phong phú và bám sát các biểu hiện tham chiếu trong văn bản. Một lần nữa, ở cả **AM, DS, MCS** thì việc bổ sung đồ thị và các phương pháp trích xuất thực thể, bộ ba không mang lại cải thiện thực sự đáng kể vượt trội.

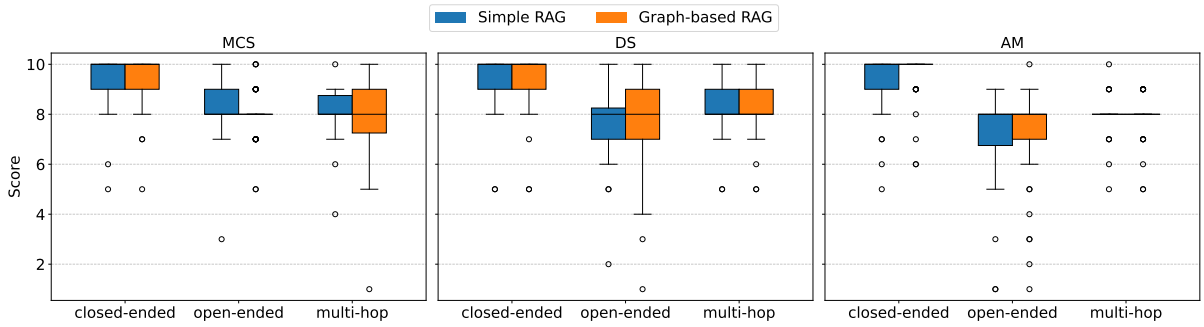
Đối với việc đánh giá suy luận đa bước, chúng tôi sử dụng các hệ số đánh giá dựa trên tham chiếu cho tập kiểm tra này. Kết quả ở Hình 5.4 cho thấy một sự tương quan đồng điệu giữa tập kiểm tra đa bước và đánh giá dựa trên tham chiếu. Ở đó, hệ số BLEU của tập dữ liệu ngành **MCS** cho kết quả vượt trội hơn hẳn so với hai ngành còn lại (dưới 20). Kết quả của hai mô hình RAG khá cạnh tranh với nhau, tuy nhiên phương pháp Simple RAG có một chút ưu thế hơn trên cả hai tập **MCS** và **DS**, ngoại trừ tập dữ liệu **AM** ở đó Graph-based RAG lại chiếm ưu thế hơn, nhưng cũng không đáng kể.

Bên cạnh việc đánh giá dựa trên các hệ số đo lường định lượng, chúng tôi cũng sử dụng một phương pháp đánh giá dựa trên AI. Như đã đề cập ở Phần 3.6.4, ưu điểm của phương pháp đánh giá này là có thể chấp nhận sự khác biệt giữa câu trả lời từ hệ thống và câu trả lời được mong đợi, và chỉ tập trung vào đánh giá sự phù hợp về mặt ngữ nghĩa. chúng tôi chia kết quả đánh giá dựa trên AI thành hai phần: (i) đánh giá theo tập dữ liệu ngành và (ii) đánh giá theo tình huống câu hỏi. Kết quả đánh giá dựa trên thang điểm 10, điểm 1 nghĩa là câu trả lời hoàn toàn không phù hợp với câu hỏi hoặc hoàn toàn không liên quan tới câu trả lời được mong đợi. Ngược lại điểm đánh giá càng cao nghĩa là câu trả lời từ mô-đun tạo sinh ngôn ngữ rất phù hợp với câu hỏi và cả câu trả lời được mong đợi, có thể chấp nhận khác biệt về cách diễn giải.

Hình 5.5 trình bày kết quả đánh giá chất lượng câu trả lời đầu ra từ mô-đun tạo sinh ngôn ngữ bằng cách sử dụng GPT-4 làm người phản biện, với thang điểm 1–10 phản ánh mức độ phù hợp ngữ nghĩa giữa câu trả lời sinh ra và câu trả lời tham chiếu. Kết quả được phân tích riêng theo ba ngành học và ba loại tập đánh giá: closed-end, opened-end và multi-hop. Nhìn chung, tất cả các phương pháp đều đạt điểm trung bình cao, dao



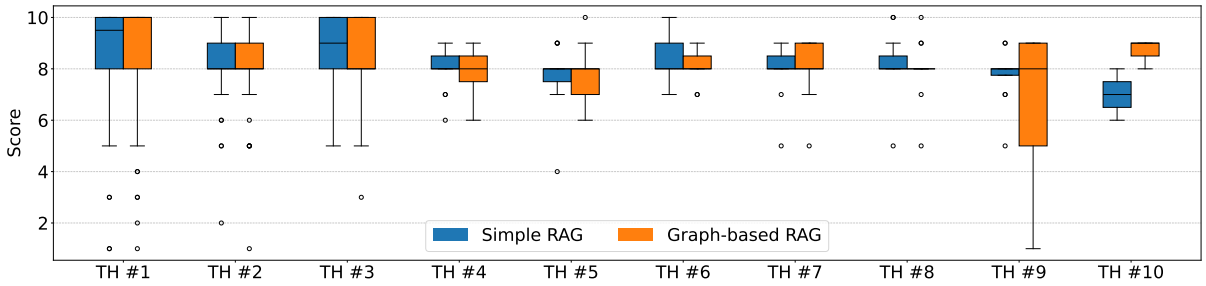
Hình 5.4: Kết quả đánh giá mô-đun tạo sinh ngôn ngữ dựa trên tập kiểm tra Multi-hop.



Hình 5.5: Kết quả sử dụng GPT-4 để đánh giá sự phù hợp về ngữ nghĩa giữa câu trả lời từ mô-đun tạo sinh ngôn ngữ và câu trả lời được mong đợi trong 2 tập dữ liệu kiểm tra: (i) nhiệm vụ đóng và (ii) dựa trên tham chiếu.

động từ 8 đến gần 10, phản ánh khả năng tạo sinh ngôn ngữ chất lượng khá tốt của hệ thống trong hầu hết các trường hợp. Không quá ngạc nhiên khi tập đánh giá closed-end đạt điểm cao nhất trên cả ba ngành (đặc biệt rõ ở MCS và AM), cho thấy hệ thống có thể sinh ra câu trả lời chính xác và phù hợp với câu tham chiếu. Trong khi đó, ở hai loại nhiệm vụ còn lại là opened-end và multi-hop thì điểm số có xu hướng thấp hơn, phản ánh độ phức tạp diễn đạt ngữ nghĩa cao hơn. Ngoài ra, sự khác biệt giữa Simple RAG và Graph-based RAG trong đánh giá ngữ nghĩa, độ chính xác trong câu trả lời dự đoán so với tham chiếu thì không quá rõ ràng. Mặc dù có một vài trường hợp Graph-based RAG cho điểm trung bình nhỉnh hơn, song sự khác biệt không ổn định và chưa đủ lớn để khẳng định về hiệu quả vượt trội. Điều này cho thấy rằng việc bổ sung cấu trúc đồ thị vào quá trình truy vấn chưa thực sự giúp cải thiện đáng kể chất lượng ngôn ngữ ở bước tạo sinh khi đánh giá dưới góc nhìn của mô hình ngôn ngữ lớn như GPT-4.

Về khía cạnh đánh giá theo tình huống câu hỏi, hình 5.6 minh họa kết quả đánh giá ngữ nghĩa bằng GPT-4 theo từng tình huống nghiên ứu, nhằm kiểm tra khả năng sinh ngôn ngữ phù hợp với ngữ cảnh cụ thể. Nhìn tổng thể, phần lớn các tình huống (số 1, 2, 3, 5, 6, 7, và 8) đều cho điểm số trung bình tương đối cao, dao động quanh ngưỡng 8 điểm trở lên, bất kể là sử dụng Simple RAG hay Graph-based RAG. Điều này cho thấy rằng hệ thống có thể tạo ra các câu trả lời nhất quán về ngữ nghĩa trong các tình huống phổ biến, với mức độ khác biệt giữa hai phương pháp là không đáng kể. Tuy nhiên, một số tình huống cụ thể lại thể hiện sự khác biệt rõ rệt giữa hai phương pháp. Ở trường hợp số 10, Graph-based RAG tỏ ra vượt trội, khi điểm đánh giá tập trung ở mức 9, cao hơn đáng kể so với Simple RAG vốn chỉ dao động quanh mức 6–8 điểm. Kết quả này cho thấy



Hình 5.6: Kết quả sử dụng GPT-4 để đánh giá sự phù hợp về ngữ nghĩa giữa câu trả lời từ mô-đun sinh ngôn ngữ và câu trả lời được mong đợi trong tập dữ liệu kiểm tra dựa trên tình huống.

rằng việc tận dụng cấu trúc đồ thị và phương pháp trích xuất thực thể, bộ ba có thể giúp hệ thống tổ chức ngữ cảnh tốt hơn cho các câu hỏi truy vấn phức tạp hoặc yêu cầu mở rộng liên kết giữa các khái niệm. Ngược lại, trường hợp số 9 lại cho thấy Graph-based RAG tạo ra kết quả thiếu ổn định, với điểm đánh giá trải rộng từ dưới 6 đến 9 điểm, trong khi Simple RAG cho phân phối điểm hẹp hơn và ổn định quanh mức 8 điểm. Điều này cho thấy rằng trong một số trường hợp, việc sử dụng đồ thị có thể dẫn đến sự nhiễu thông tin truy xuất hoặc mất trọng tâm, làm giảm độ nhất quán cho câu trả lời tạo sinh ngôn ngữ.

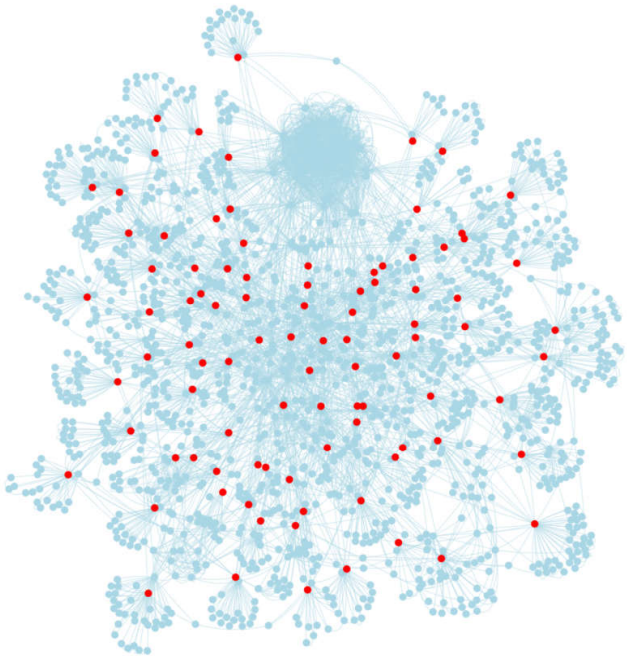
Phân tích đồ thị

Để có cái nhìn tổng quát hơn về độ phức tạp của một đồ thị tri thức được sử dụng trong một tập dữ liệu theo ngành. Chúng tôi đã có thông kê thông qua Bảng 5.5, sử dụng mô hình nhúng text-embedding-3-small. Dựa vào bảng này, dễ dàng nhận thấy tổng số nút đoạn văn tương ứng với tổng số học phần ở mỗi ngành. Điều này có nghĩa là hệ thống không chia nhỏ các tài liệu học phần thành các đoạn nhỏ hơn. Chính xác là như vậy, vốn dĩ tổng số token lớn nhất trong một học phần ở mỗi ngành là không quá lớn (**DS: 211 token,MCS: 229 token, AM: 215 token**), và có thể vừa cửa sổ ngữ cảnh đầu vào cho LLM để thực hiện cho mỗi đoạn văn tác vụ trích xuất thực thể là 230 token và trích xuất bộ ba là 230 token. Bên cạnh đó, Hình 5.7, Hình 5.8 và Hình 5.9 sẽ là các đồ thị tri thức của lần lượt ba ngành.

Bảng 5.5: Thông tin đồ thị tri thức theo từng tập dữ liệu ngành.

	AM	**MCS**	**DS**
Tổng số nút đoạn văn	94	90	75
Tổng số nút	1624	1621	1414
Tổng số bộ ba mối liên hệ	1904	1934	1642
Tổng số bộ ba cùng nghĩa	5065	4764	3175
Tổng số cạnh	9006	8773	6593

Trong đồ thị ngành AM (Hình 5.7), các nút học phần màu đỏ có xu hướng tập trung mạnh vào khu vực trung tâm, cho thấy mức độ liên kết cao giữa các học phần và các



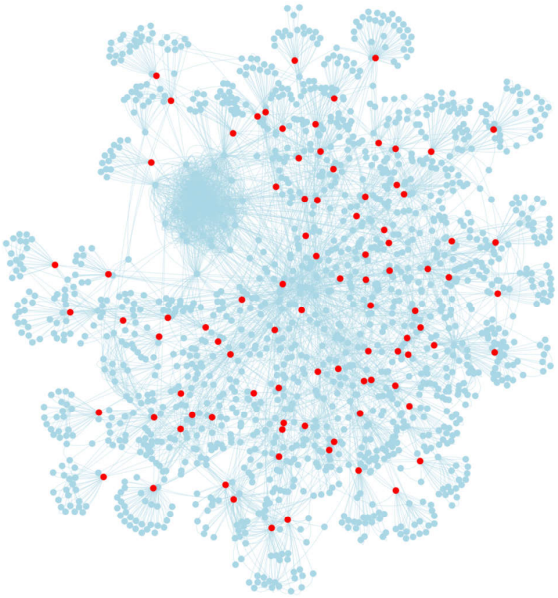
Hình 5.7: Đồ thị tri thức ngành AM, trong đó: các nút đỏ là nút đoạn văn, nút xanh là nút thực thể.

thực thể tri thức liên quan. Mạng lưới này dường như có cấu trúc "tập trung ở phần lõi", phản ánh đặc điểm của các môn học cơ bản trong ngành AM, vốn thường xuyên chia sẻ nền tảng kiến thức với nhau (ví dụ như đại số, giải tích, phương trình vi phân, v..v).

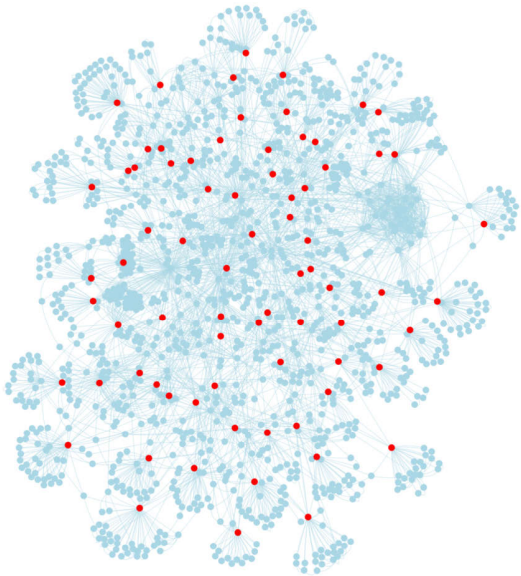
Ngược lại, đồ thị ngành MCS và DS (Hình 5.8, và Hình 5.9) thì lại thể hiện sự phân bố đều hơn giữa các học phần, với các cụm học phần lan tỏa ra nhiều nhánh nhỏ. Điều này phần nào phản ánh tính chất rộng và đa lĩnh vực của 2 ngành này, nơi mà các học phần có thể tập trung vào các hướng chuyên biệt như thuật toán, mạng máy tính, trí tuệ nhân tạo, hay hệ thống phần mềm, mỗi hướng đều có tập thực thể tri thức đặc thù. Ngoài ra, thêm một lý do nữa cho sự phân tán ở ngành DS là số lượng học phần ở ngành này lại ít hơn 2 học phần còn lại khoảng 15 học phần.

Mặc dù đồ thị tri thức của ngành AM thể hiện cấu trúc trung tâm tập trung và tính gắn kết cao giữa các học phần, song kết quả thực nghiệm lại cho thấy hiệu suất truy xuất thông tin – đo bằng chỉ số $R@5$ trung bình ở Bảng 5.2 – lại thấp hơn đáng kể so với 2 ngành còn lại. Hiện tượng này phản ánh một điểm quan trọng rằng cấu trúc đồ thị gắn kết không đồng nghĩa với hiệu quả truy xuất cao. Chúng tôi có thể lý giải điều này từ một số khía cạnh:

Với đồ thị AM, sự tập trung vào các thực thể tri thức lõi khiến cho không gian truy vấn bị giới hạn trong một vùng tri thức chặt chẽ, làm giảm độ đa dạng khi truy xuất. Khi mô hình gặp một truy vấn có tính phân biệt cao hoặc yêu cầu kiến thức ngữ cảnh cụ thể, khả năng mở rộng sang các thực thể ngoại biên trở nên hạn chế – dẫn đến việc mô hình có thể lặp lại hoặc đưa ra các ngữ cảnh không đủ đặc trưng trong top-5 kết quả đầu tiên. Ngược lại, đồ thị ngành MCS và DS tuy phân tán hơn, nhưng lại tạo điều kiện cho các mô hình khai thác tri thức theo nhiều hướng khác nhau. Mỗi học phần có thể



Hình 5.8: Đồ thị tri thức ngành MCS, trong đó: các nút đỏ là nút đoạn văn, nút xanh là nút thực thể.



Hình 5.9: Đồ thị tri thức ngành DS, trong đó: các nút đỏ là nút đoạn văn, nút xanh là nút thực thể.

liên kết đến một cụm tri thức chuyên biệt, giúp tăng khả năng tìm ra các kết quả phù hợp hơn với nội dung truy vấn. Điều này cho phép *Recall*@5 đạt giá trị cao hơn, vì mô hình có thể phát hiện ra nhiều phương án trả lời có liên quan nằm rải rác trong đồ thị.

Thứ hai, yếu tố nội dung và phong cách viết mô tả học phần cũng có thể ảnh hưởng đến hiệu quả truy xuất. Các học phần ngành MCS và DS có xu hướng sử dụng nhiều cụm từ chuyên ngành và khái niệm kỹ thuật rõ ràng hơn, giúp hệ thống trích xuất thông tin và sinh ngôn ngữ hiệu quả hơn. Trong khi đó, nội dung học phần ngành AM thường mang tính trừu tượng, dùng ngôn ngữ toán học hoặc diễn đạt khái quát hơn, có thể gây khó khăn cho mô hình trong quá trình ánh xạ ngữ nghĩa.

Phân tích các mẫu kiểm tra thất bại

Trong Phần này, chúng tôi phân tích các trường hợp hệ thống HippoRAG-2 đưa ra kết quả không chính xác trên tập kiểm tra thuộc cả ba ngành. Sau quá trình đánh giá và đối chiếu giữa câu trả lời của hệ thống và các đáp án tham chiếu trong bộ dữ liệu kiểm thử, chúng tôi nhận thấy các mẫu thất bại chủ yếu rơi vào hai nhóm nguyên nhân: (i) câu trả lời tham chiếu trong tập kiểm tra chứa sai sót hoặc không chính xác; (ii) trả lời theo dạng yes/no trong khi câu hỏi yêu cầu nội dung chi tiết; (iii) câu trả lời từ hệ thống bị thiếu thông tin cần thiết hoặc không truy xuất được nội dung phù hợp. Việc phân tích những trường hợp thất bại này giúp làm rõ hơn những sai lệch trong các kết quả định lượng được trình bày ở hai Phần trước là 5.2.2 và 5.2.2.

- **Trường hợp (i):** Trong câu hỏi thứ 23 của ngành AM ở tập closed-end là *How many skills are enhanced in English 2?*, hệ thống trả lời chính xác là 4. Tuy nhiên, câu trả lời từ tham chiếu lại là English 2. Bên cạnh đó, chúng tôi nhận thất trường hợp (i) chỉ xuất hiện ở tập kiểm tra closed-end: tập DS, MCS và AM đều có 2 câu sai ở trường hợp này. Đây được xem là một hạn chế trong quá trình kiểm duyệt dữ liệu tham chiếu của chúng tôi, đòi hỏi sự rà soát chặt chẽ hơn trong các nghiên cứu tiếp theo.

- **Trường hợp (ii):** Chẳng hạn như trong câu hỏi thứ 50 của ngành MCS ở tập closed-end là *Is Introduction to Machine Learning a compulsory or electivet course?*. Đây không phải là câu hỏi dạng yes/no, nhưng hệ thống lại trả lời là yes. Vì vậy nên lỗi chỉ cũng chỉ xuất hiện ở tập kiểm tra closed-end: tập DS và MCS đều có 2 câu sai, riêng AM thì chỉ có 1 câu sai. Điều này cho thấy sự "thiên vị" khi xử lý các câu hỏi dạng yes/no. Cụ thể, hệ thống thường đưa ra câu trả lời 'yes' ngay cả khi câu hỏi yêu cầu một câu trả lời lựa chọn hoặc diễn giải, điều này phần nào phản ánh sự thiếu linh hoạt trong khả năng nhận diện và xử lý dạng câu hỏi của LLM.

- **Trường hợp (iii):** Ở câu hỏi thứ 6 của ngành DS ở tập opened-end có câu hỏi *What are some of the basic legal concepts and terms covered in the general objective of General Law course?* thì hệ thống không truy xuất được nội dung phù hợp. Những lỗi này xuất hiện ở trong tập kiểm tra opened-end và multihop.

Bảng 5.6 cho thấy số lượng lỗi thuộc trường hợp (iii), tức hệ thống không truy xuất được nội dung phù hợp để trả lời. Lỗi này có xu hướng xuất hiện nhiều ở cả hai tập kiểm

tra Opened-end và Multihop, nhưng đặc biệt cao trong tập Multihop. Điều này cho thấy hệ thống gặp khó khăn rõ rệt khi xử lý các câu hỏi dạng Multihop..

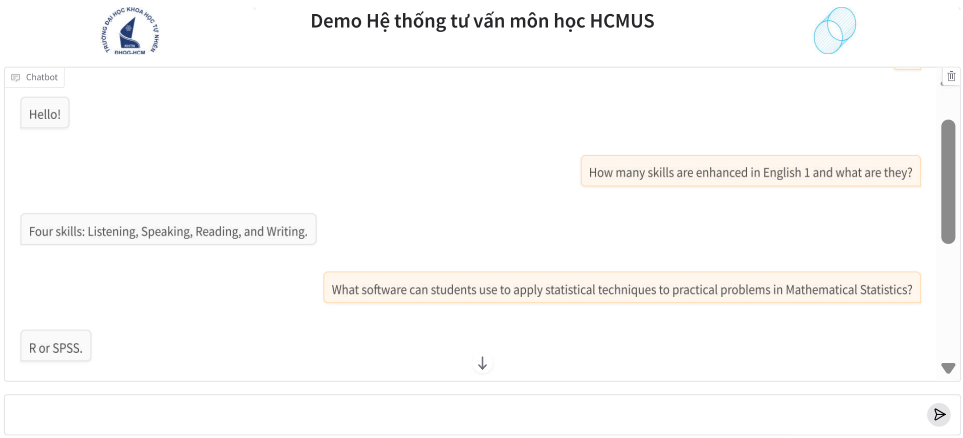
Bảng 5.6: Bảng thống kê số lượng trường hợp kiểm tra sai trường hợp (iii) ở bộ kiểm tra Opened-end và Multihop theo ngành tương ứng.

Ngành	Opened-end	Multihop
AM	3	5
MCS	1	3
DS	1	5

Chi phí thực nghiệm

Chúng tôi sẽ trình bày về chi phí thực nghiệm cho toàn bộ công đoạn thực nghiệm khi sử dụng hai mô hình RAG cùng với GPT-4o mini ở Phần Phụ lục C. Chi phí thực nghiệm cho Phần 5.2.2.

Demo thực nghiệm



Hình 5.10: Cửa sổ demo.

5.2.3 Sử dụng mô hình ngôn ngữ Llama-3.1-8B

Ở phần thực nghiệm này, chúng tôi sẽ đánh giá và so sánh hiệu suất truy xuất và tạo sinh ngôn ngữ của cả hai mô hình Simple RAG và Graph-based RAG, sử dụng mô hình nhúng có phần đơn giản hơn là Contriever cùng với mô hình ngôn ngữ Llama-3.1-8B của AI@Meta (2024).

Kết quả hiệu suất truy xuất thông tin

Bảng 5.7: So sánh hiệu suất truy vấn thông tin giữa Simple RAG và Graph-based RAG theo Recall@top-k (R@k) sử dụng Llama-3.1-8B trong trích xuất thực thể, bộ ba và mô hình Contriever cho ba ngành **AM, DS và MCS**. Kết quả vượt trội sẽ được **in đậm**.

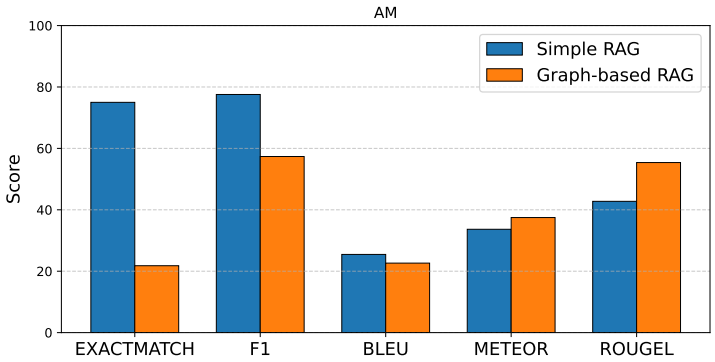
Mô hình	Closed-end QA			Opened-end QA			Multi-hop QA			Average		
	R@1	R@2	R@5	R@1	R@2	R@5	R@1	R@2	R@5	R@1	R@2	R@5
AM												
Simple RAG	42.3	46.2	59.6	38.5	48.1	55.8	27.2	41.1	60.0	36.0	45.1	58.5
Graph-based	**75.0**	**82.7**	**94.2**	**80.8**	**90.4**	**98.1**	**38.9**	**73.9**	**90.6**	**64.9**	**82.3**	**94.3**
DS												
Simple RAG	55.0	62.5	75.0	42.5	52.5	62.5	21.7	35.3	60.3	39.7	50.1	65.9
Graph-based RAG	**67.5**	**80.0**	**92.5**	**70.0**	**92.5**	**97.5**	**36.1**	**65.6**	**86.1**	**57.9**	**79.4**	**92.1**
MCS												
Simple RAG	38.0	46.0	68.0	80.0	88.0	92.0	17.8	37.8	62.2	45.3	57.3	74.1
Graph-based RAG	**64.0**	**82.0**	**96.0**	**86.0**	**92.0**	**100.0**	**31.9**	**62.0**	**90.0**	**64.7**	**78.7**	**95.3**

Bảng 5.7 thể hiện sự vượt trội rõ rệt của mô hình Graph-based RAG so với Simple RAG trên cả ba ngành **AM**, **DS** và **MCS**, cũng như ở tất cả các loại câu hỏi và 3 mức độ truy xuất (R@1, R@2 và R@5) khi kết hợp sử dụng Llama-3.1-8B. Đặc biệt, các chỉ số trung bình R@5 ở cả ba ngành khi sử dụng Graph-based RAG luôn cao hơn 90, vượt trội hơn cả điểm trung bình R@5 cao nhất của Graph-based RAG khi sử dụng các mô hình nhúng hiện đại: ở Bảng 5.2 (70.9), Bảng 5.3 (78.5) và Bảng 5.4 (87.5). Điều này chứng minh rằng sự kết hợp ăn ý giữa LLama-3.1-8B và Contriever trong Graph-based RAG mang lại hiệu quả cao trong truy xuất và rất phù hợp với cả bộ dữ liệu ở cả 3 ngành. Ngược lại với Simple RAG kết quả trung bình R@5 cho thấy khi sử dụng Contriever không mang lại sự vượt trội: ở AM là 58.5 (trong khi ở Bảng 5.2 là 70.5), DS là 65.9 (trong khi ở ở Bảng 5.3 là 78.5) và MCS là 74.1 (trong khi ở Bảng 5.4 là 88.6).

Kết quả đánh giá hiệu suất tạo sinh ngôn ngữ

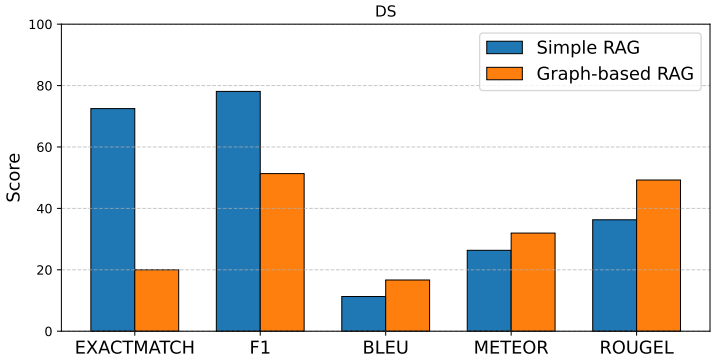
Tiếp theo, chúng tôi so sánh hiệu suất sinh ngôn ngữ giữa Simple RAG và Graph-based RAG với tập dữ liệu AM, DS và MCS và tập câu hỏi nhiệm vụ đóng (Hình 5.11, Hình 5.12 và Hình 5.13) và câu hỏi mở (Hình 5.14). Với tập câu hỏi nhiệm vụ đóng, Simple RAG cho hiệu suất Exact Match và F1 là vượt trội hơn nhiều so với RAG dựa trên đồ thị. Tuy nhiên khi xét về khả năng ngữ nghĩa và cấu trúc câu từ tạo sinh và tham chiếu ở cả hai hệ thống không quá chênh lệch nhiều và đạt hiệu suất ở mức khá. Từ đó, có thể thấy việc LLM sinh câu trả lời với RAG dựa trên đồ thị không giống hoàn toàn với tham chiếu tuy nhiên xét về mặt ý nghĩa cũng như so sánh trên từng tổ hợp cụm từ thì câu trả lời của RAG dựa trên đồ thị so với Simple RAG thì không chênh lệch quá nhiều. Với tập dữ liệu câu hỏi mở, chênh lệch giữa 2 hệ thống là không quá rõ ràng. Với tập dữ liệu DS giá trị BLEU ở cả hai mô hình đều có điểm thấp nhất so với các ngành khác, chỉ khoảng 13–14, điểm số METEOR và ROUGEL cho thấy hiệu suất khá và cũng tương

đương nhau.Với hai mô hình có thể thấy điểm chung là BLEU thấp có thể cho thấy chất lượng câu trả lời của cả hai mô hình thiếu sự trùng khớp chính xác về từ ngữ. Tuy nhiên, với METEOR và ROUGEL, kết quả khá cho thấy mô hình có thể vẫn truyền tải ý nghĩa tốt dù khác cách diễn đạt. Còn với hai ngành học còn lại nhìn chung khá ổn và không chênh lệch quá lớn giữa hai hệ thống

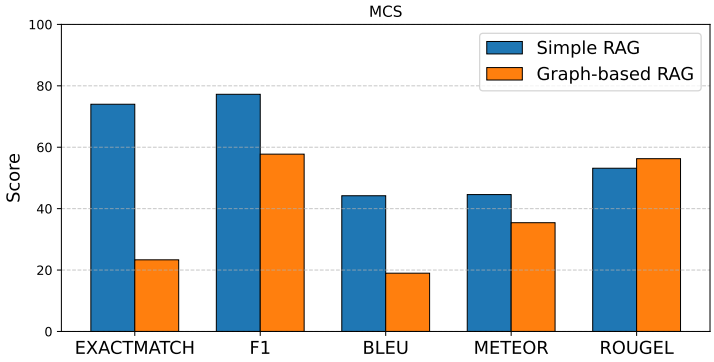


Hình 5.11: Kết quả đánh giá mô-đun tạo sinh ngôn ngữ với mô hình nhúng Contriever và Llama-3.1-8B trên hai tập dữ liệu kiểm tra là closed-end và opened-end của ngành **AM**. Hai hệ số đầu tiên ExactMatch và F1 dùng để đánh giá tập kiểm tra nhiệm vụ đóng, trong khi đó ba hệ số còn lại gồm BLEU, METEOR, và ROUGEL dùng để đánh giá dựa trên tham chiếu.

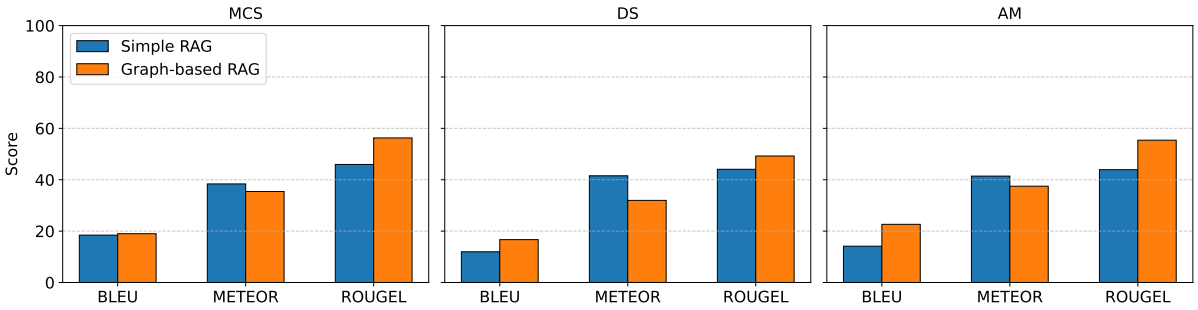
Tương tự với F1-Score, thì BLEU với mục tiêu tìm trùng khớp dựa trên tỷ lệ tổ hợp các cụm n-gram là 4 từ trùng khớp giữa câu trả lời sinh ra từ mô hình ngôn ngữ và tham chiếu chính xác. Với ba bộ dữ liệu câu hỏi, có thể thấy điều đặc biệt là tập Closed-end có chỉ số cao hơn Opened-end ở tập dữ liệu, điều đặc biệt này xảy ra khi mô hình ngôn ngữ Llama-3.1-8B thường xuyên trả về các trả lời ngắn và lấy đúng các từ trọng tâm ở trong các đoạn văn mà không có thêm diễn giải dẫn đến hiện tượng như vậy. Còn với tập dữ liệu Multihop cho cả 3 tập dữ liệu luôn đạt kết quả khá tệ khi chỉ khoảng 10% đến 15% càng thể hiện việc không liên kết được các đoạn văn và dùng các thông tin để đưa ra câu trả lời hợp lý.



Hình 5.12: Kết quả đánh giá mô-đun tạo sinh ngôn ngữ với mô hình nhúng Contriever và Llama-3.1-8B trên hai tập dữ liệu kiểm tra closed-end và opened-end của ngành **DS**. Hai hệ số đầu tiên ExactMatch và F1 dùng để đánh giá tập kiểm tra nhiệm vụ đóng, trong khi đó ba hệ số còn lại gồm BLEU, METEOR, và ROUGEL dùng để đánh giá dựa trên tham chiếu.

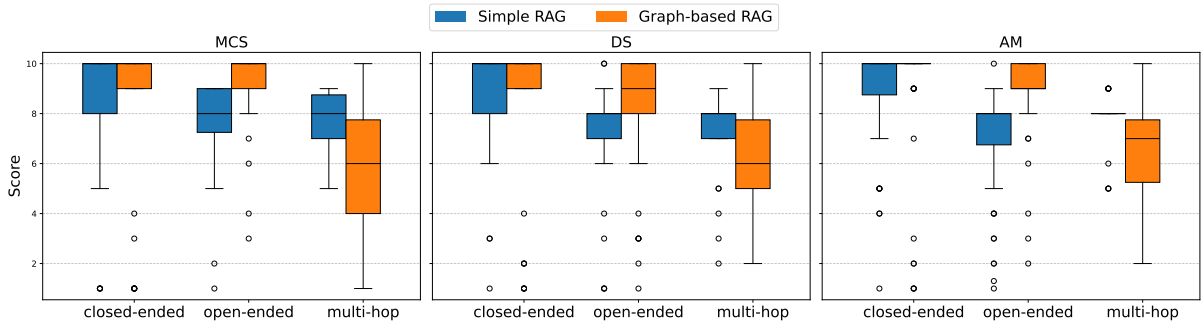


Hình 5.13: Kết quả đánh giá mô-đun tạo sinh ngôn ngữ với mô hình nhúng Contriever và Llama-3.1-8B trên hai tập dữ liệu kiểm tra closed-end và opened-end của ngành **MCS**. Hai hệ số đầu tiên ExactMatch và F1 dùng để đánh giá tập kiểm tra nhiệm vụ đóng, trong khi đó ba hệ số còn lại gồm BLEU, METEOR, và ROUGEL dùng để đánh giá dựa trên tham chiếu.

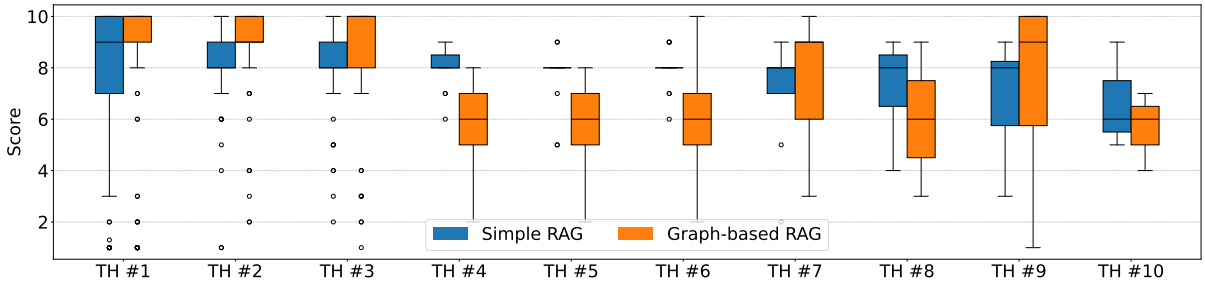


Hình 5.14: Kết quả đánh giá mô–đun tạo sinh ngôn ngữ dựa trên tập kiểm tra multi-hop.

Hình 5.15 cho thấy được khả năng chấp nhận được của các câu trả lời từ hệ thống so với tham chiếu và được đánh giá bởi mô hình ngôn ngữ lớn khác là GPT-4o mini với tập dữ liệu câu hỏi Closed-end và Opened-end là cao khi dao động từ 9 đến 10 và ít các điểm ngoại lai thấp khẳng định khả năng trả lời các câu hỏi của hệ thống khi sử dụng Llama-3.1-8B là chấp nhận được, tuy nhiên với tập dữ liệu DS và bộ dữ liệu câu hỏi Opened-end chỉ nằm ở mức khá khi có nhiều mức điểm thấp, điều này do khả năng truy xuất thông tin của tập dữ liệu DS thấp hơn so với 2 tập dữ liệu còn lại dẫn đến câu trả lời được sinh ra thiếu thông tin, không kết nối được các đoạn văn bản với nhau. Cuối cùng là tập dữ liệu câu hỏi Multihop cho thấy điểm đánh giá từ GPT-4 thấp với bộ dữ liệu AM là tốt nhất khi không có điểm 0. Điều này chỉ ra rằng dù hệ thống có khả năng truy xuất thông tin đa bước nhưng mô hình ngôn ngữ vẫn gặp khó khăn trong việc tổng hợp, xâu chuỗi và diễn đạt chính xác câu trả lời suy luận. Biểu hiện rõ nhất là ở ngành DS và MCS, nơi điểm số thấp chiếm tỷ lệ đáng kể, cho thấy mô hình hoặc không truy xuất được đủ thông tin liên quan, hoặc không thể tái cấu trúc chúng một cách hợp lý trong câu trả lời cuối cùng.



Hình 5.15: Kết quả sử dụng GPT-4 để đánh giá sự phù hợp về ngữ nghĩa giữa câu trả lời từ mô–đun sinh ngôn ngữ và câu trả lời được mong đợi trong 2 tập dữ liệu kiểm tra: (i) nhiệm vụ đóng và (ii) dựa trên tham chiếu.



Hình 5.16: Kết quả sử dụng GPT-4 để đánh giá sự phù hợp về ngữ nghĩa giữa câu trả lời từ mô–đun sinh ngôn ngữ và câu trả lời được mong đợi trong tập dữ liệu kiểm tra dựa trên tình huống.

Hình 5.16 so sánh khả năng sinh ngôn ngữ dựa trên câu hỏi và ngữ cảnh trả lời của Simple RAG và Graph-based RAG. Nhìn chung tổng thể, hệ thống RAG dựa trên đồ thị cho khoảng điểm đánh giá rộng hơn nhiều và có những điểm đánh giá dưới 5 là nhiều hơn so với Simple RAG. Với các câu hỏi thuộc dạng trường hợp 1, 2, 3 và 8, 9, 10 cho thấy chênh lệch giữa giá trị trung bình và mức điểm đánh giả phổ biến của Simple RAG và Graph-based RAG là không chênh lệch quá nhiều khi cả hai đều đạt hiệu suất là khá cao. Tuy nhiên, với các trường hợp 4, 5, 6 cho thấy sự khác biệt rõ ràng nhất giữa Simple RAG và Graph-based RAG khi mức điểm của RAG dựa trên đồ thị cho khoảng điểm trải dài từ 1 đến 10 và mức điểm phổ biến là thấp hơn nhiều so với Simple RAG. Từ đó, có thể thấy Graph-based RAG không quá vượt trội so với Simple RAG, với các câu trả lời các thông tin đơn giản không cần liên kết các đoạn thông tin thì điểm số là tốt hơn nhưng khi cần liên kết các đoạn văn thông tin hoặc đưa ra thông tin suy luận thì RAG dựa trên đồ thị cho các kết quả tệ hơn Simple RAG.

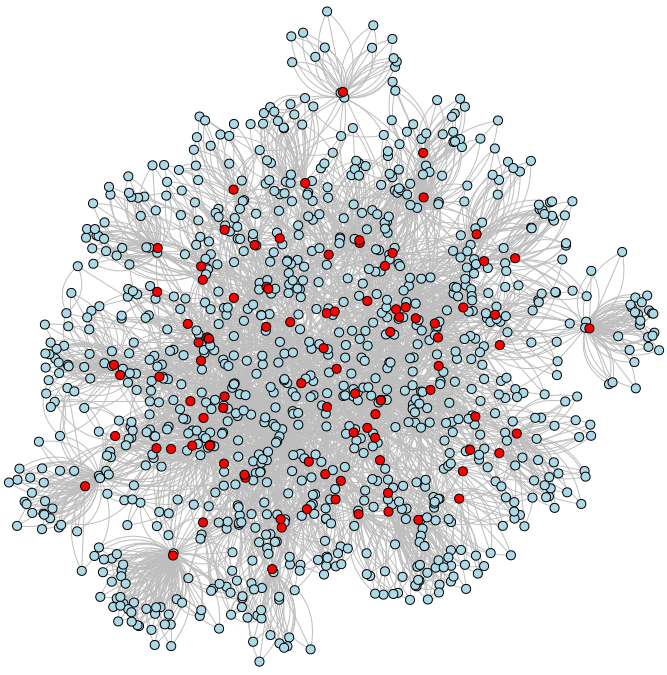
Phân tích đồ thị

Chúng tôi triển khai việc đánh giá chất lượng của giai đoạn ngoại tuyến cho hệ thống RAG có đồ thị này bằng việc phân tích cấu trúc đồ thị tri thức của 3 tập dữ liệu.

Bảng 5.8: Thông tin đồ thị tri thức theo từng tập dữ liệu ngành, sử dụng Llama-3.1-8B để trích xuất thực thể, bộ ba và mô hình nhúng Contriever.

Thống kê	**AM**	**MCS**	**DS**
Tổng số nút đoạn văn	94	90	75
Tổng số nút	976	1016	822
Tổng số bộ ba mối liên hệ	1236	1280	1002
Tổng số bộ ba cùng nghĩa	1889	2047	1522
Tổng số cạnh	4448	4702	3603

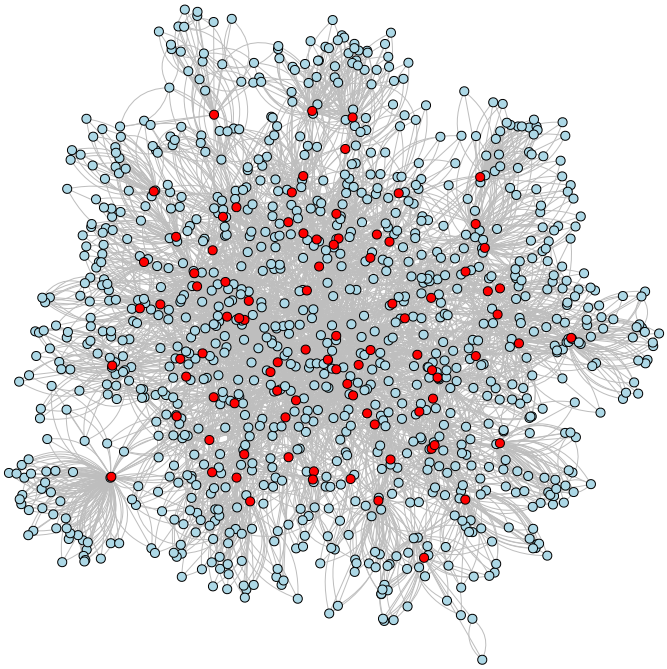
So sánh (i) Bảng 5.5 (sử dụng GPT-4o mini cùng với text-embedding-3-small) với ii) Bảng 5.8 (sử dụng Llama-3.1-8B và Contriever), ta thấy (i) tạo ra số lượng nút, bộ ba quan hệ và cạnh trong đồ thị tri thức cao hơn đáng kể trên cả ba ngành AM, DS và MCS. Ví dụ, tổng số cạnh trong (i) (AM: 9006) gần gấp đôi so với (ii) (AM: 4448). Mặc dù, điều này cho thấy GPT-4o mini kết hợp với embedding text-3 có khả năng trích xuất nhiều thực thể và quan hệ phong phú hơn, góp phần tạo ra đồ thị tri thức chi tiết và liên kết chặt chẽ hơn. Nhưng, kết quả của Graph-based RAG ở (i) lại không vượt trội hơn (ii). Từ đó, chúng tôi nhận thấy rằng sự phức tạp trong đồ thị không thật sự hỗ trợ và ảnh hưởng đáng kể trong giai đoạn truy xuất.



Hình 5.17: Đồ thị tri thức ngành AM, với nút đỏ là nút đoạn văn và nút xanh là nút thực thể.

Với đồ thị ngành AM (Hình 5.17) có thể thấy các nút học phần màu đỏ có xu hướng tập trung vào khu vực trung tâm của đồ thị, kết hợp với đó là các nút thực thể của đồ thị cho thấy khả năng liên kết lớn cho thấy mối quan hệ chặt chẽ giữa nội dung các môn

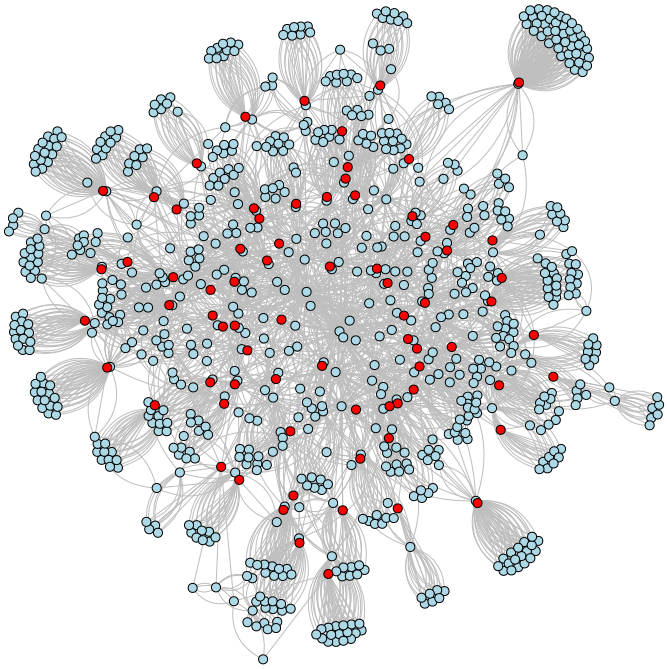
học. Điều này có thể giải thích với ngành Toán Ứng dụng khi các môn học sử dụng các nền tảng của Toán học như đại số, đại sô tuyến tính, giải tích, vi tích phân, xác suất thống kê, ... Ngoài ra, với đồ thị tri thức cũng càng thể hiện rõ khả năng trích xuất thông tin bộ ba tri thức của Llama-3.1-8B yếu hơn nhiều so với đồ thị trích xuất bằng GPT-4o mini (hình 5.7) khi mức độ liên kết giữa các nút đoạn văn và nút thực thể cũng như các nút thực thể là tốt hơn khá nhiều so với Llama-3.1-8B.



Hình 5.18: Đồ thị tri thức ngành MCS với nút đỏ là nút đoạn văn và nút xanh là nút thực thể.

Đồ thị tri thức ngành MCS (hình 5.18) cho thấy các nút thực thể có phần phức tạp như đồ thị tri thức của AM (hình 5.17). Ngoài ra, các nút màu đỏ là các nút đoạn văn bản và các nút thực thể màu xanh cho thấy có độ phân tán thay vì tạo thành từng nhóm và các nút liên kết chặt chẽ như AM (hình 5.17). Một điều có giải thích và làm rõ hơn khi chương trình đào tạo ngành Toán Tin có nội dung các môn học thuộc về hướng Toán học như các môn giai đoạn đại cương của ngành Toán Ứng dụng tập trung vào phát triển toán học như đại số, giải tích và xác suất thống kê để phục vụ việc sử dụng các công cụ và phương pháp toán học trong lĩnh vực tin học và có hướng tập trung vào mảng tin học, kĩ thuật và công nghệ như các môn lập trình Web, hệ điều hành, phần mềm tính toán và kết hợp tin học và thống kê tập trung khoa học dữ liệu như ML, AI và xử lý dữ liệu lớn. Tuy nhiên, khi sử dụng mô hình ngôn ngữ có lượng tham số là nhỏ thì mức độ phức tạp về số lượng thực thể cùng bộ ba là ít hơn nhiều dẫn tới khả năng mở rộng không tốt như Hình 5.8.

Với đồ thị tri thức ngành DS, có thể thấy các nút đoạn văn và các nút thực thể là phân tán đều nhất trong cả 3 ngành là AM, MCS và DS, tồn tại một số cụm nút thực thể là độc lập, chúng chỉ liên kết với đoạn văn bản chứa chúng nhưng không có thấy được các mối liên kết đồng nghĩa. Lý do giải thích cho việc này là do sự phân bổ ở nhiều việc đào tạo ở nhiều lĩnh vực, bao gồm các môn Toán học như: Giải tích, Đại số tuyến tính, Xác



Hình 5.19: Đồ thị tri thức ngành DS với nút đỏ là nút đoạn văn và nút xanh là nút thực thể.

suất - Thống kê, và Toán rời rạc; cùng với các môn Tin học như: Nhập môn phần mềm tính toán và Nhập môn lập trình máy tính và vào giai đoạn chuyên ngành chia thành các môn học hỗ trợ cho 2 định hướng là khoa học và công nghệ. Ngoài ra, độ phức tạp của đồ thị tri thức chưa đủ tốt được như hình 5.9.

Phân tích mẫu kiểm tra thất bại

Ở phần này, chúng tôi sẽ phân tích một số nguyên nhân khiến hệ thống gặp vấn đề khi hoạt động. Ở đây, chúng tôi tập trung phân tích vào các câu hỏi khiến Recall@5 nhỏ hơn 1 và một số câu mà hệ thống đưa ra câu trả lời không chính xác.

Đầu tiên là với câu hỏi "What is the prerequisite for English 1?" thì hệ thống trả về kết quả là "English 1" trong khi câu trả lời đúng là None.

Bảng 5.9: Hai ví dụ cho việc hệ thống đưa ra trả lời sai.

Câu truy vấn	What is the prerequisite for English 1?
Câu trả lời từ hệ thống	English 1
Câu trả lời chính xác	none.
Ttitle của đoạn văn chứa thông tin	BAA00011.
Title của đoạn văn được truy xuất	1.BAA00012. 2.BAA00013. 3.MTH10119 4.MTH00010. **5.BAA00011**
Bộ ba liên quan đến truy vấn (Top-5)	("english 1", "is the prerequisite for", "the course with id baa00012") ("teaching mathematics in english 1", "prerequisites are", "none") ("baa00013", "prerequisite is", "english 2") ("calculus 1a", "has no prerequisites for", "this course") ("teaching mathematics in english 1", "is", "elective course") Chỉ có chủ ngữ của duy nhất một bộ ba xuất hiện trong đoạn văn chứa thông tin
Bộ ba sau khi sắp xếp	("english 1", "is the prerequisite for", "the course with id baa00012")

Với câu hỏi như trên 5.9 có thể thấy hệ thống có khả năng đưa ra các bộ ba liên quan đến truy vấn và sau khi sắp xếp đều chứa thực thể liên quan đến môn học English 1, từ đó truy xuất được các đoạn văn có chứa môn học English 1 trùng khớp với đoạn văn chứa thông tin. Tuy nhiên, vì khả năng hạn chế của mô hình ngôn ngữ với đoạn văn dài nên hệ thống đưa về thông tin cuối cùng là sai lệch so với tham chiếu. Chúng tôi đã tổng hợp số trường hợp sai này ở Bảng 5.10.

Bảng 5.10: Số lượng mẫu thất bại (Recall@5 < 100) theo từng loại nhiệm vụ truy vấn thông tin cho các ngành AM, DS và MCS.

Ngành	**Closed-end QA**	**Opened-end QA**	**Multi-hop QA**
AM	3	1	6
DS	3	1	10
MCS	2	0	7

Chi phí thực nghiệm

Ở phần này, chúng tôi tập trung vào chi phí cho các mô hình nhúng như Contriever và mô hình ngôn ngữ Llama-3.1-8B và đưa ra nhận định với chi phí tiêu hao

Chi phí tài nguyên phần cứng

Mô hình nhúng Contriever cho tiêu hao phần cứng là nhẹ nhất khi chiếm dụng bộ nhớ máy tính là khoảng 23.08MB, ngoài ra đây là một mô hình với mã nguồn mở và hỗ trợ với nhiều thư viện khác nhau nên khá dễ dàng để triển khai.

Với mô hình ngôn ngữ Llama-3.1-8B, chúng tôi triển khai chúng qua Ollama API với nền tảng Docker, để sử dụng mượt mà và ổn định chúng tôi đề nghị nên sử dụng GPU để có thể hiệu quả nhất. Chúng tôi sử dụng GPU là GTX 3060 với 12GB VRAM cùng với mô đun ThreadPoolExecutor của Python giúp tận dụng xử lý song song các tác vụ, giúp trích xuất các thông tin thực thể mở nhanh hơn.

Vì chúng tôi sử dụng một mô hình ngôn ngữ lớn là GPT-4o mini để đánh giá câu trả lời từ hệ thống, vì vậy sẽ tốn một lượng chi phí nhỏ để chạy hệ thống đánh giá. Trong đó, với phần 5.2.3 này chúng tôi đã dùng 1,405 yêu cầu tới GPT-4o mini và tổng 267,255 tokens do yêu cầu gửi và GPT-4o mini trả về, lượng chi phí tiêu tốn chỉ là $0.04.

Chương 6

Tổng kết

6.1 Tóm tắt những đóng góp nổi bật

Luận văn này khởi đầu từ một nhu cầu thực tiễn: làm thế nào để giúp sinh viên tiếp cận thông tin học phần một cách hiệu quả hơn, nhanh chóng hơn, thông qua các hệ thống hỏi-đáp thông minh. Từ đó, chúng tôi đã xây dựng một hệ thống gợi ý học phần theo dạng câu hỏi, dựa trên hai hướng tiếp cận là RAG cơ bản và RAG kết hợp đồ thị tri thức – một hướng tiếp cận đang nhận được sự quan tâm mạnh mẽ trong lĩnh vực truy xuất tăng cường sinh văn bản. Và dựa trên các kết quả thực nghiệm, chúng tôi đưa ra nhận xét rằng hệ thống RAG cơ bản (SimpleRAG) phù hợp hơn so với RAG tích hợp đồ thị (HippoRAG-2/Graph-based RAG) khi sử dụng một mô hình nhúng hiện đại và một LLM phù hợp, thì sẽ cả về mặt chi phí, thời gian thực thi và hiệu quả đầu ra.

1. Bước đầu tiên trong hành trình này là xây dựng đường ống xử lý dữ liệu đầu vào, bao gồm hàng chục tài liệu mô tả học phần từ ba ngành học đặc thù DS, MCS và AM. Nhằm chuẩn hóa và rút trích thông tin một cách hiệu quả, chúng tôi đã áp dụng và so sánh bốn thuật toán đo độ tương đồng giữa hai chuỗi – bao gồm khoảng cách Levenshtein, LCS, hệ số Jaccard, và Jaro-Winkler. Mỗi thuật toán mang một đặc điểm riêng trong việc xử lý các biến thể ngữ nghĩa và lỗi chính tả, và kết quả thực nghiệm đã giúp chúng tôi lựa chọn giải pháp phù hợp cho từng bước xử lý cụ thể.

2. Tiếp đó, chúng tôi tập trung nghiên cứu hai kiến trúc chính là RAG cơ bản và RAG sử dụng đồ thị. Trong quá trình này, các mô hình nhúng đóng vai trò then chốt – từ các mô hình mã nguồn mở như GritLM-7B, Nvidia NV-Embed-v2 đến mô hình API của OpenAI. Cùng với đó là khả năng của hệ thống hỏi đáp với nhiều mô hình nhúng khác nhau với mô hình ngôn ngữ lớn từ khác nhau. Đặc biệt, ở hướng RAG đồ thị, chúng tôi đã tích lũy thêm một kiến thức mới về thuật toán duyệt đồ thị PPR để khai thác tri thức liên kết giữa các học phần, từ đó tăng cường ngữ cảnh truy xuất một cách có hệ thống hơn.

3. Cuối cùng, chúng tôi tiến hành so sánh tổng quan giữa hai hướng tiếp cận – từ hiệu suất truy xuất, chất lượng sinh câu trả lời, cho đến tính nhất quán trong trả lời các câu hỏi đa bước. Quá trình đánh giá được thực hiện thông qua một tập kiểm

tra gồm nhiều dạng câu hỏi (Closed-end, Opened-end và Multihop), với các dữ liệu đầu vào đã được xử lý và chuẩn hóa từ bước đầu. Những phân tích định tính và định lượng trong chương 5 đã cho thấy rằng mặc dù RAG đồ thị có ưu thế trong các tình huống yêu cầu suy luận dựa trên liên kết học phần, song RAG cơ bản – nhờ sự tiến bộ vượt bậc của các mô hình nhúng hiện đại, cũng đạt kết quả rất cạnh tranh với thời gian xử lý ngắn hơn và kiến trúc đơn giản hơn.

6.2 Những hạn chế trong nghiên cứu và đề xuất giải pháp

Trong quá trình triển khai và đánh giá hệ thống, chúng tôi nhận thấy một số hạn chế nhất định, bao gồm:

Hạn chế về độ tin cậy của tập kiểm thử: Một trong những yếu tố ảnh hưởng đến kết quả đánh giá của hệ thống là chất lượng và độ tin cậy của tập kiểm thử. Trong luận văn này, chúng tôi đã tự xây dựng tập câu hỏi-đáp dựa trên dữ liệu học phần thuộc ba ngành DS, MCS và AM. Các câu hỏi được soạn bởi sinh viên và sau đó được chúng tôi rà soát, chỉnh lý. Tuy nhiên, vẫn có khả năng tồn tại sai sót trong khâu biên soạn hoặc kiểm duyệt đáp án, đặc biệt là ở các câu hỏi có phạm vi rộng, đòi hỏi kiến thức tổng hợp. Bên cạnh đó, một hạn chế tinh tế hơn nhưng cũng rất quan trọng là cách chúng tôi xác định câu trả lời tham chiếu. Ở một số trường hợp, các câu trả lời mẫu được trình bày khá cứng nhắc hoặc "chuẩn hóa" theo một cách diễn đạt duy nhất, trong khi thực tế, cùng một nội dung có thể được diễn đạt theo nhiều cách khác nhau mà vẫn giữ nguyên thông tin cốt lõi. Việc thiếu sự linh hoạt này có thể khiến hệ thống bị đánh giá thấp một cách không công bằng, đặc biệt khi sử dụng các chỉ số định lượng như BLEU, METEOR hay ROUGE-L – vốn dựa vào mức độ trùng khớp chuỗi từ hoặc cụm từ giữa câu trả lời sinh ra và đáp án tham chiếu.

Hạn chế trong cơ chế lưu trữ và mở rộng quy mô hệ thống: Cả hai mô hình Simple RAG và Graph-based RAG đều được triển khai dưới hình thức tự lưu trữ[1], với toàn bộ dữ liệu truy xuất và kết quả lưu trữ tại máy chủ nội bộ. Ngoài ra, khi hiện này xuất hiện các cơ sở dữ liệu cho các mô hình nhúng hoặc cho các dữ liệu đồ thị tri thức giúp đỡ rất nhiều khi tăng thêm không gian lưu trữ khi số lượng dữ liệu tăng dần là một hạn chế với cơ chế hiện tại. Điều này tuy giúp kiểm soát dữ liệu tốt hơn, nhưng lại đặt ra nhiều vấn đề khi hệ thống cần mở rộng quy mô hoặc tích hợp với các hệ thống đào tạo thực tế tại trường đại học. Hiện nay, các hệ thống RAG mới ra đời cũng chủ yếu được thiết kế theo kiến trúc đóng, thiếu các giải pháp lưu trữ linh hoạt hoặc quản lý truy vấn mang tính cộng tác.

Chưa đánh giá được trải nghiệm thực tế của người dùng cuối: Mặc dù mục tiêu của luận văn là xây dựng một hệ thống tư vấn học phần dạng hỏi-đáp phục vụ cho sinh viên, nhưng cho đến thời điểm hiện tại, hệ thống mới chỉ dừng lại ở mức thử nghiệm nội bộ. Chúng tôi chưa triển khai hệ thống ở môi trường thực tế, cũng như chưa thu thập được phản hồi trực tiếp từ sinh viên – nhóm người dùng cuối có vai trò then chốt trong việc đánh giá hiệu quả sử dụng, mức độ dễ hiểu của câu trả lời và khả năng hỗ trợ học

[1] self-hosted

tập thực sự của hệ thống.

Giới hạn về phạm vi ứng dụng trong lĩnh vực khai phá dữ liệu giáo dục: Một hạn chế khác của luận văn nằm ở phạm vi triển khai ứng dụng: hệ thống hiện tại mới chỉ tập trung vào nhiệm vụ hỏi-đáp thông tin các học phần với đầu vào là dữ liệu mô tả học phần và đầu ra là các câu trả lời ngắn gọn phục vụ nhu cầu tra cứu của sinh viên. Điều này khiến hệ thống chủ yếu hoạt động như một công cụ tìm kiếm thông minh theo ngữ nghĩa, chưa đi sâu vào các khía cạnh phức tạp hơn của lĩnh vực khai phá dữ liệu giáo dục EDM[2], chẳng hạn như cá nhân hóa gợi ý học phần theo hành vi học tập, phân tích quá trình tiến bộ của sinh viên, hoặc dự đoán kết quả học tập dựa trên dữ liệu lịch sử.

6.3 Đề xuất hướng nghiên cứu tiếp theo

Trong quá trình triển khai và thử nghiệm hệ thống tư vấn học phần dựa trên mô hình RAG, chúng tôi nhận thấy rằng tiềm năng phát triển của hệ thống không chỉ nằm ở việc tối ưu hóa mô hình truy xuất và sinh văn bản, mà còn ở khả năng mở rộng phạm vi dữ liệu và tích hợp sâu hơn vào thực tế sử dụng. Dưới đây là một số hướng phát triển quan trọng nhằm nâng cao hiệu quả, độ chính xác và giá trị ứng dụng của hệ thống trong tương lai.

Mở rộng phạm vi dữ liệu, ngôn ngữ và ngữ cảnh truy vấn: Trong phiên bản hiện tại, hệ thống chủ yếu khai thác nội dung từ mô tả học phần hoàn toàn bằng Tiếng Anh. Tuy nhiên, trên thực tế, sinh viên cần nhiều hơn thế. Không những về mặt ngôn ngữ Tiếng Việt, các sinh viên có như câu muốn đặt các câu hỏi về học phần có thể liên quan đến nội dung giảng dạy cụ thể, phương pháp đánh giá, mức độ khó, hoặc thậm chí là cảm nhận của sinh viên các khóa trước. Do đó, việc mở rộng dữ liệu từ mô tả học phần sang các tài liệu học thuật như đề cương chi tiết, giáo trình, kế hoạch giảng dạy, đánh giá sinh viên và kết quả học tập trước đó sẽ giúp hệ thống đưa ra câu trả lời đầy đủ và phù hợp với bối cảnh thực tế hơn.

Hướng ứng dụng và triển khai thực tế: Như đã đề cập ở phần hạn chế, chúng tôi nhận thấy một trong những định hướng khả thi và giàu tiềm năng nhất là triển khai hệ thống dưới dạng chatbot tích hợp trực tiếp vào cổng thông tin sinh viên. Thay vì phải dò tìm thông tin từ file PDF hay hỏi qua email, sinh viên chỉ cần nhập câu hỏi vào giao diện chat và nhận được câu trả lời gần như ngay lập tức. Việc này không chỉ cải thiện trải nghiệm người dùng, mà còn giúp giảm tải khối lượng công việc của bộ phận cố vấn học tập trong trường. Bên cạnh đó, hệ thống có thể phát huy hiệu quả hơn nữa nếu được tích hợp thêm dữ liệu cá nhân hóa, chẳng hạn như kết quả học tập, lịch sử đăng ký môn học, mục tiêu nghề nghiệp hoặc thời gian tốt nghiệp dự kiến của từng sinh viên. Khi đó, hệ thống không chỉ trả lời các câu hỏi thông thường, mà còn có thể đưa ra gợi ý môn học phù hợp nhất cho từng cá nhân, như: *"Bạn nên học môn X vào học kỳ sau để đảm bảo điều kiện tốt nghiệp đúng hạn"*, hay *"Môn Y phù hợp với định hướng phân tích dữ liệu mà bạn đang theo đuổi"*.

[2]Educational Data Mining

6.4 Kết luận

Từ một hệ thống hỏi-đáp dựa trên mô hình RAG cơ bản, cùng với sự hỗ trợ và tham gia từ các mô hình nhúng hiện và LLM hiện đại, chúng tôi tin rằng hướng phát triển trong tương lai có thể biến hệ thống này thành một trợ lý học thuật thông minh, cá nhân hóa, linh hoạt và tối ưu chi phí và tài nguyên. Từ đó, hệ thống có thể đồng hành song song cùng sinh viên xuyên suốt hành trình đại học. Bằng việc mở rộng phạm vi dữ liệu, đa dạng hóa ngôn ngữ và tích hợp thực tế sử dụng, hệ thống có thể trở thành một phần không thể thiếu trong hệ sinh thái giáo dục đại học hiện đại – nơi mà thông tin học thuật cần được truy cập nhanh chóng, chính xác và theo nhu cầu cá nhân.

Tài liệu tham khảo

Aggarwal, C. C. (2015). *Data Mining: The Textbook*. Springer, Cham, Switzerland.

AI@Meta (2024). Llama 3 Model Card. https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md. Accessed: 2025-06-12.

Alon, U., Xu, F., He, J., Sengupta, S., Roth, D., and Neubig, G. (2022). Neuro-symbolic language modeling with automaton-augmented retrieval. In *International Conference on Machine Learning*, pages 468–485. PMLR.

Asai, A., Wu, Z., Wang, Y., Sil, A., and Hajishirzi, H. (2024). Self-rag: Learning to retrieve, generate, and critique through self-reflection. In *The Twelfth International Conference on Learning Representations (ICLR)*.

Banerjee, S. and Lavie, A. (2005). METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan. Association for Computational Linguistics.

Borgeaud, S., Mensch, A., Hoffmann, J., Cai, T., Rutherford, E., Millican, K., Van Den Driessche, G., Lespiau, J.-B., Damoc, B., Clark, A., et al. (2022). Improving language models by retrieving from trillions of tokens. In *International Conference on Machine Learning*, pages 2206–2240. PMLR.

Brin, S. and Page, L. (1998). The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Cheng, X., Luo, D., Chen, X., Liu, L., Zhao, D., and Yan, R. (2023). Lift yourself up: Retrieval-augmented text generation with self-memory. In *Thirty-seventh Conference on Neural Information Processing Systems (NeurIPS)*.

Cherifi, C., Labatut, V., and Santucci, J.-F. (2011). On flexible web services composition networks. In *Digital Information and Communication Technology and Its Applications (DICTAP 2011), Proceedings, Part I*, Lecture Notes in Computer Science, pages 45–59, Dijon, France. Springer Berlin Heidelberg.

Christen, P. (2012). The data matching process. In *Data Matching*, pages 23–35. Springer Berlin Heidelberg.

Da, C., Wang, P., and Yao, C. (2022). Levenshtein ocr. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 322–338, Cham. Springer Nature Switzerland.

Denkowski, M. and Lavie, A. (2010). Extending the METEOR machine translation evaluation metric to the phrase level. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*, pages 250–253, Los Angeles, California. Association for Computational Linguistics.

Diana, N. E. and Ulfa, I. H. (2019). Measuring performance of n-gram and jaccard-similarity metrics in document plagiarism application. In *Journal of Physics: Conference Series*, volume 1196, page 012069. IOP Publishing.

Dong, Y., Wang, S., Zheng, H., Chen, J., Zhang, Z., and Wang, C. (2024). Advanced rag models with graph structures: Optimizing complex knowledge reasoning and text generation. In *2024 5th International Symposium on Computer Engineering and Intelligent Communications (ISCEIC)*, pages 626–630. IEEE.

Edge, D., Trinh, H., Cheng, N., Bradley, J., Chao, A., Mody, A., Truitt, S., Metropolitansky, D., Ness, R. O., and Larson, J. (2024). From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*.

Elastic (2024). Elasticsearch learned sparse encoder (elser). Accessed: 2025-07-13.

Esteban, A., Zafra, A., and Romero, C. (2020). Helping university students to choose elective courses by using a hybrid multi-criteria recommendation system with genetic optimization. *Knowledge-Based Systems*, 194:105385.

Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., Dai, Y., Sun, J., Wang, H., and Wang, H. (2023). Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2(1).

Glass, M., Rossiello, G., Chowdhury, M. F. M., Naik, A., Cai, P., and Gliozzo, A. (2022). Re2g: Retrieve, rerank, generate. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2701–2715.

Gong, E. J., Bang, C. S., Lee, J. J., Park, J., Kim, E., Kim, S., Kimm, M., and Choi, S. H. (2024). The potential clinical utility of the customized large language model in gastroenterology: A pilot study. *Bioengineering*, 12:1.

Grandini, M., Bagli, E., and Visani, G. (2020). Metrics for multi-class classification: an overview. *arXiv preprint arXiv:2008.05756*.

Gu, J., Wang, C., and Zhao, J. (2019). Levenshtein transformer. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc. NeurIPS 2019.

Guo, Z., Xia, L., Yu, Y., Ao, T., and Huang, C. (2024). Lightrag: Simple and fast retrieval-augmented generation. .

Gutiérrez, B. J., Shu, Y., Gu, Y., Yasunaga, M., and Su, Y. (2024). Hipporag: Neurobiologically inspired long-term memory for large language models. In *Advances in Neural Information Processing Systems 37 (NeurIPS 2024)*.

Gutiérrez, B. J., Shu, Y., Qi, W., Zhou, S., and Su, Y. (2025). From rag to memory: Non-parametric continual learning for large language models. *arXiv preprint arXiv:2502.14802*.

Guu, K., Lee, K., Tung, Z., Pasupat, P., and Chang, M.-W. (2020). Retrieval augmented language model pre-training. In *International Conference on Machine Learning*, pages 3929–3938. PMLR.

Han, Z. F., Lin, J., Gurung, A., Thomas, D. R., Chen, E., Borchers, C., Gupta, S., and Koedinger, K. R. (2024). Improving assessment of tutoring practices using retrieval-augmented generation. *arXiv preprint arXiv:2402.14594*.

Haveliwala, T. H. (2002). Topic-sensitive pagerank. In *Proceedings of the 11th International Conference on World Wide Web (WWW '02)*, pages 517–526, Honolulu, Hawaii, USA. ACM.

Hinton, G., Vinyals, O., and Dean, J. (2015). Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Huang, W., Lapata, M., Vougiouklis, P., Papasarantopoulos, N., and Pan, J. (2023). Retrieval augmented generation with rich answer encoding. In *Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1012–1025.

Ivison, H., Wang, Y., Pyatkin, V., Lambert, N., Peters, M., Dasigi, P., Jang, J., Wadden, D., Smith, N. A., Beltagy, I., et al. (2023). Camels in a changing climate: Enhancing lm adaptation with tulu 2. *arXiv preprint arXiv:2311.10702*.

Izacard, G., Caron, M., Hosseini, L., Riedel, S., Bojanowski, P., Joulin, A., and Grave, E. (2021). Unsupervised dense information retrieval with contrastive learning. *arXiv preprint arXiv:2112.09118*.

Izacard, G. and Grave, E. (2021). Leveraging passage retrieval with generative models for open domain question answering. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 874–880.

Izacard, G., Lewis, P., Lomeli, M., Hosseini, L., Petroni, F., Schick, T., Dwivedi-Yu, J., Joulin, A., Riedel, S., and Grave, E. (2023). Atlas: Few-shot learning with retrieval augmented language models. *Journal of Machine Learning Research*, 24(251):1–43.

Jaccard, P. (1901). Étude comparative de la distribution florale dans une portion des alpes et du jura. *Bulletin de la Société Vaudoise des Sciences Naturelles*, 37:547–579.

Jaro, M. A. (1989). Advances in record-linkage methodology as applied to matching the 1985 census of tampa, florida. *Journal of the American Statistical Association*, 84:414–420.

Jiang, Z., Xu, F., Gao, L., Sun, Z., Liu, Q., Dwivedi-Yu, J., Yang, Y., Callan, J., and Neubig, G. (2023). Active retrieval augmented generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7969–7992. Association for Computational Linguistics.

Karpukhin, V., Oguz, B., Min, S., Lewis, P. S., Wu, L., Edunov, S., Chen, D., and 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)*, volume 1, pages 6769–6781.

Kwiatkowski, T., Palomaki, J., Redfield, O., Collins, M., Parikh, A., Alberti, C., Epstein, D., Polosukhin, I., Devlin, J., Lee, K., et al. (2019). Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

Lee, C., Roy, R., Xu, M., Raiman, J., Shoeybi, M., Catanzaro, B., and Ping, W. (2024). Nv-embed: Improved techniques for training llms as generalist embedding models. *arXiv preprint arXiv:2405.17428*.

Levenshtein, V. I. (1965). Binary codes capable of correcting deletions, insertions, and reversals. In *Doklady Akademii Nauk*, volume 163, pages 845–848. Russian Academy of Sciences.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Kulikov, I., Ghazvininejad, M., Luo, M., Yih, W.-t., et al. (2020). Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474.

Li, M., Kilicoglu, H., Xu, H., and Zhang, R. (2025a). Biomedrag: A retrieval augmented large language model for biomedicine. *Journal of Biomedical Informatics*, 162:104769.

Li, Z., Li, C., Zhang, M., Mei, Q., and Bendersky, M. (2024). Retrieval augmented generation or long-context llms? a comprehensive study and hybrid approach. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 881–893.

Li, Z., Wang, Z., Wang, W., Hung, K., Xie, H., and Wang, F. L. (2025b). Retrieval-augmented generation for educational application: A systematic survey. *Computers and Education: Artificial Intelligence*, page 100417.

Lin, C.-Y. (2004). ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Liu, S., Wright, A. P., McCoy, A. B., Huang, S. S., Steitz, B., and Wright, A. (2025). Detecting emergencies in patient portal messages using large language models and

knowledge graph-based retrieval-augmented generation. *Journal of the American Medical Informatics Association*, 32(6):1032–1039.

Lo, K., Wang, L. L., Neumann, M., Kinney, R., and Weld, D. S. (2019). S2orc: The semantic scholar open research corpus. *arXiv preprint arXiv:1911.02782*.

Logan, R., Fleischmann, Z., Annis, S., et al. (2022). 3GOLD: optimized levenshtein distance for clustering third-generation sequencing data. *BMC Bioinformatics*, 23(1):95.

Mallen, A., Asai, A., Zhong, V., Das, R., Khashabi, D., and Hajishirzi, H. (2022). When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. *arXiv preprint arXiv:2212.10511*.

Matsumoto, N., Moran, J., Choi, H., Hernandez, M. E., Venkatesan, M., Wang, P., and Moore, J. H. (2024). Kragen: a knowledge graph-enhanced rag framework for biomedical problem solving using large language models. *Bioinformatics*, 40(6):btae353.

Meta AI (2023). Llama 2: Open foundation and chat models. Accessed: 2025-07-17.

Mistral AI (2023). Announcing mistral 7b. Accessed: 2025-07-17.

Muennighoff, N., Hongjin, S., Wang, L., Yang, N., Wei, F., Yu, T., Singh, A., and Kiela, D. (2024). Generative representational instruction tuning. In *ICLR 2024 Workshop: How Far Are We From AGI*.

Muennighoff, N., Tazi, N., Magne, L., and Reimers, N. (2022). Mteb: Massive text embedding benchmark. *arXiv preprint arXiv:2210.07316*.

Neumann, A. T., Yin, Y., Sowe, S., Decker, S., and Jarke, M. (2025). An llm-driven chatbot in higher education for databases and information systems. *IEEE Transactions on Education*, 68:103–116.

Nomic AI (2024). Gritlm-7b - hugging face. https://huggingface.co/nomic-ai/GritLM-7B. Accessed: 2025-06-22.

NVIDIA (2024). Nv-embed-v2 - hugging face. https://huggingface.co/nvidia/NV-Embed-v2. Accessed: 2025-06-22.

OpenAI (2024). New embedding models and api updates. https://openai.com/blog/new-embedding-models-and-api-updates. Accessed: 2025-06-19.

OpenAI (2025). Openai platform – api pricing. https://platform.openai.com/docs/pricing. Accessed: 2025-06-22.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Podani, J. (2021). The wonder of the jaccard coefficient: from alpine floras to bipartite networks. *Flora Mediterranea*, 31:105–123.

Procko, T. T. and Ochoa, O. (2024). Graph retrieval-augmented generation for large language models: A survey. In *2024 Conference on AI, Science, Engineering, and Technology (AIxSET)*, pages 166–169. IEEE.

Ram, O., Levine, Y., Dalmedigos, I., Muhlgay, D., Shashua, A., Leyton-Brown, K., and Shoham, Y. (2023). In-context retrieval-augmented language models. *Transactions of the Association for Computational Linguistics*, 11:1316–1331.

Robertson, S. and Zaragoza, H. (2009). The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389.

Salton, G. and Yu, C. T. (1974). On the construction of effective vocabularies for information retrieval. In *Operator Algebras, Unitary Representations, Enveloping Algebras, and Invariant Theory*, pages 48–60. Birkhäuser, Boston, MA.

Santhanam, K., Khattab, O., Saad-Falcon, J., Potts, C., and Zaharia, M. (2022). Colbertv2: Effective and efficient retrieval via lightweight late interaction. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3715–3734.

Sarthi, P., Abdullah, S., Tuli, A., Khanna, S., Goldie, A., and Manning, C. D. (2024). Raptor: Recursive abstractive processing for tree-organized retrieval. In *The Twelfth International Conference on Learning Representations*.

Sawarkar, K., Mangal, A., and Solanki, S. R. (2024). Blended rag: Improving rag (retriever-augmented generation) accuracy with semantic search and hybrid query-based retrievers. In *2024 IEEE 7th International Conference on Multimedia Information Processing and Retrieval (MIPR)*, pages 155–161. IEEE.

Shareef, S. H. M., Sulaiman, R. A., and Basari, A. S. H. (2023). The hybrid jaro-winkler and manhattan distance using dissimilarity measure for test case prioritization approach. *International Journal of Advanced Computer Science & Applications*, 14(11).

Taneja, K., Maiti, P., Kakar, S., Guruprasad, P., Rao, S., and Goel, A. K. (2024). Jill watson: A virtual teaching assistant powered by chatgpt. In *Artificial Intelligence in Education*, pages 324–337. Springer Nature, Switzerland.

Teyler, T. J. and DiScenna, P. (1986). The hippocampal memory indexing theory. *Behavioral neuroscience*, 100(2):147.

Vast.ai (2025). Vast.ai – gpu rental marketplace. https://vast.ai. Accessed: 2025-06-18.

Wang, L., Yang, N., Huang, X., Yang, L., Majumder, R., and Wei, F. (2023). Improving text embeddings with large language models. *arXiv preprint arXiv:2401.00368*.

Wang, Y., Ren, R., Li, J., Zhao, W. X., Liu, J., and Wen, J.-R. (2024). Rear: A relevance-aware retrieval-augmented framework for open-domain question answering. *arXiv preprint arXiv:2402.17497*.

Winkler, W. E. (1990). String comparator metrics and enhanced decision rules in the fellegi-sunter model of record linkage. In *Proceedings of the Section on Survey Research Methods*, pages 354–359. American Statistical Association.

Wu, M., Luo, S., Xu, H., and Sun, W. (2025). Optimizing emergency decision-making for natural disaster using light retrieval augmented generation. In *2025 8th International Conference on Advanced Algorithms and Control Engineering (ICAACE)*, pages 2237–2241. IEEE.

Xu, C., Zhao, D., Wang, B., and Xing, H. (2024a). Enhancing retrieval-augmented lms with a two-stage consistency learning compressor. In *International Conference on Intelligent Computing*, pages 511–522. Springer.

Xu, F., Shi, W., and Choi, E. (2024b). Recomp: Improving retrieval-augmented lms with context compression and selective augmentation. In *The Twelfth International Conference on Learning Representations (ICLR)*.

Xu, Z., Cruz, M. J., Guevara, M., Wang, T., Deshpande, M., Wang, X., and Li, Z. (2024c). Retrieval-augmented generation with knowledge graphs for customer service question answering. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2905–2909.

Yancey, W. E. (2005). Evaluating string comparator performance for record linkage. Technical Report RR2005/05, US Bureau of the Census.

Yang, Z., Qi, P., Zhang, S., Bengio, Y., Cohen, W. W., Salakhutdinov, R., and Manning, C. D. (2018). Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*.

Yu, Y., Ping, W., Liu, Z., Wang, B., You, J., Zhang, C., Shoeybi, M., and Catanzaro, B. (2024). Rankrag: Unifying context ranking with retrieval-augmented generation in llms. *Advances in Neural Information Processing Systems*, 37:121156–121184.

Yuan, T., Ning, X., Zhou, D., Yang, Z., Li, S., Zhuang, M., Tan, Z., Yao, Z., Lin, D., Li, B., et al. (2024). Lv-eval: A balanced long-context benchmark with 5 length levels up to 256k. *arXiv preprint arXiv:2402.05136*.

Phụ lục

A. Mô tả 5 đoạn văn mẫu

5 đoạn văn mẫu

1. The course with ID {**course_id**} is titled {**course_name**}. It is offered in the {**semester**} semester and taught by {**teacher_name**}. This is a {**course_type**} course with {**required_prerequisites**} as its prerequisite. Students will gain the following outcomes: {**learning_outcomes**}. The course content covers: {**content**}.

2. {**course_name**} (ID: {**course_id**}) is a {**course_type**} subject conducted during the {**semester**} semester. The course is instructed by {**teacher_name**}, and it {**required_prerequisites**}. Upon completion, students will achieve: {**learning_outcomes**}. The course includes: {**content**}.

3. In the {**semester**} semester, students can take {**course_name**} (code: {**course_id**}). It is taught by {**teacher_name**}, categorized as a {**course_type**} course. {**required_prerequisites**}. Learning outcomes include: {**learning_outcomes**}. Topics covered in this course are: {**content**}.

4. {**course_name**} is a {**course_type**} course (ID: {**course_id**}) available in the {**semester**} semester. It is led by {**teacher_name**}. Prerequisites: {**required_prerequisites**}. The course aims to deliver the following outcomes: {**learning_outcomes**}. Main content: {**content**}.

5. Course {**course_name**} (ID: {**course_id**}) falls under the {**course_type**} category and is scheduled for the {**semester}** semester. The instructor is {**teacher_name**}. Note: {**required_prerequisites**}. Learning outcomes expected: {**learning_outcomes**}. The course discusses: {**content**}.

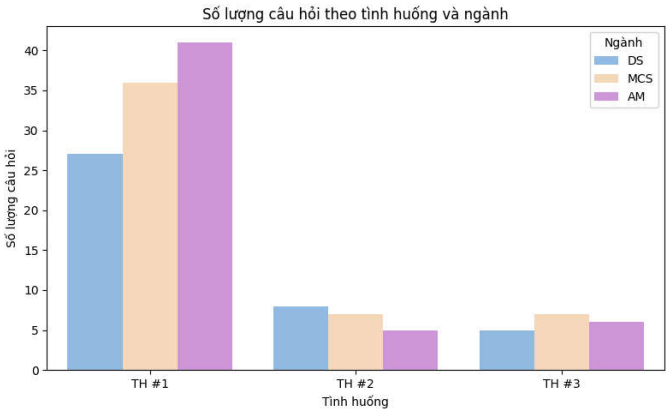
Hình 6.1: Thông tin về 5 đoạn văn mẫu được sử dụng trong phần soạn dữ liệu đầu vào cho hệ thống HippoRAG-2.

B. Thông tin về bộ dữ liệu kiểm tra

Về phần soạn ba bộ kiểm tra hỏi đáp ở Phần 5.1, đều được soạn thủ công. Cụ thể, chúng tôi đã nhờ sự trợ giúp của 30 bạn sinh viên khóa 2021, trong đó có 10 bạn thuộc ngành Khoa học dữ liệu, 10 bạn thuộc ngành Toán Ứng dụng và 10 bạn còn lại thuộc ngành Toán Tin. Các bạn sinh viên này sẽ được chia làm ba nhóm tương tự với ngành các bạn học. Mỗi nhóm sẽ được đọc về nội dung của các học phần đã qua xử lý, trong ngành tương ứng và mỗi bạn sẽ đặt ít nhất là 10 đến 20 câu hỏi xuyên suốt tất cả các "nhóm" học phần (từ đại cương đến cơ sở ngành đến chuyên ngành) cho từng bộ kiểm tra, dựa trên các tình huống nghiên cứu mà chúng tôi đã đề ra, kèm với câu trả lời mà các bạn muốn được nhận phản hồi lại dựa trên thông tin các bạn đã đọc cũng như những trải nghiệm hiểu biết của các bạn. Yêu cầu cho câu trả lời là các danh từ hay cụm từ phải giống với dữ liệu gốc. Tất cả các câu hỏi và trả lời đều sẽ phải viết bằng tiếng anh.

Sau khi đã thu thập các bộ câu hỏi-đáp của các nhóm, chúng tôi sẽ tiến hành sắp xếp cũng như loại bỏ những câu hỏi bị trùng lặp nhau. Đồng thời chúng tôi cũng rà soát lại từng câu hỏi và câu trả lời so với dữ liệu gốc để đảm bảo độ chính xác. Sau khi qua tất cả các bước sàng lọc và tinh chỉnh thì chúng tôi cũng có ba bộ kiểm tra hoàn chỉnh. Tuy nhiên, vì sự chênh lệnh về tổng số học phần trong một ngành học, chúng tôi quyết định sẽ tách $\frac{1}{2}$ số học phần trong mỗi ngành - một nửa là để soạn nội dung cho tập closed_end, một nửa còn lại để soạn nội dung cho tập opened_end như bảng 5.1. Lý giải cho điều này, chúng tôi muốn kiểm tra một cách toàn vẹn hệ thống RAG trên toàn bộ nội dung của tất cả các học phần của một ngành học cụ thể. Về số lượng câu hỏi cho tập multihop, bởi lẽ do "độ khó" của bộ này nằm ở việc phải truy vấn từ ít nhất là 2 cho tới 5 học phần, vì vậy số lượng câu hỏi-đáp của các bạn sinh viên cũng bị trùng khá nhiều, vì vậy chúng tôi chọn lọc ra mỗi bộ sẽ có 30 câu hỏi, sẽ đồng đều về từng tình huống nghiên cứu. Tuy nhiên, vẫn đảm bảo các câu hỏi-đáp sẽ dàn trải khắp các nhóm học phần từ đại cương - cơ sở ngành và chuyên ngành.

Bộ kiểm tra closed_end

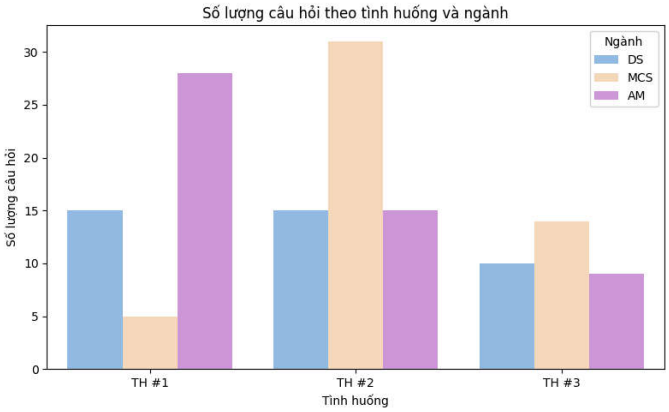


Hình 6.2: Số lượng câu hỏi cho từng tình huống trong tập closed_end.

Hình 6.2 minh họa sự chênh lệch rõ rệt về số lượng câu hỏi được đặt ra cho tình huống 1 so với hai tình huống còn lại, và xu hướng này xuất hiện đồng đều ở cả ba nhóm ngành. Sự khác biệt này phần nào phản ánh nhu cầu phổ biến của sinh viên trong việc tìm hiểu tổng quan các thông tin cơ bản xoay quanh học phần — đặc biệt là các học phần thuộc nhóm đại cương và cơ sở ngành — nhằm phục vụ cho việc đăng ký học. Tuy nhiên, đáng chú ý là vẫn có khoảng một phần ba số câu hỏi tập trung vào tình huống 2 và 3 ở tất cả các nhóm ngành. Điều này cho thấy, bên cạnh mong muốn nắm bắt thông tin cơ bản, sinh viên cũng quan tâm đến các khía cạnh chuyên sâu hơn như định hướng đầu ra hay nội dung cụ thể của học phần.

Bộ kiểm tra opened_end

Như đã đề cập ở trên, tập opened_end bao gồm các câu hỏi liên quan đến các học phần từ mức cơ sở ngành đến chuyên ngành. Dựa vào hình 6.3 có thể quan sát được sự khác biệt đáng kể trong hành vi đặt câu hỏi giữa các nhóm ngành, tùy theo từng tình huống



Hình 6.3: Số lượng câu hỏi cho từng tình huống trong tập opened_end.

nghiên cứu. Trong tình huống thứ nhất, khi sinh viên được yêu cầu tìm kiếm thông tin tổng quan về học phần, nhóm ngành AM thể hiện sự chủ động nổi bật với số lượng câu hỏi cao nhất. Kết quả này phản ánh nhu cầu cao của sinh viên ngành AM trong việc tiếp cận và nắm bắt thông tin cơ bản về học phần – có thể là nhằm hỗ trợ quá trình lựa chọn hoặc lên kế hoạch học tập cá nhân. Ngược lại, nhóm MCS có số lượng câu hỏi thấp nhất, cho thấy mức độ quan tâm đến thông tin khái quát của học phần ở nhóm này là tương đối hạn chế. Sang tình huống thứ hai, liên quan đến việc truy vấn các mục tiêu đầu ra của học phần, sinh viên nhóm MCS lại nổi bật với số lượng câu hỏi vượt trội. Điều này cho thấy họ có xu hướng chủ động hơn trong việc tìm hiểu định hướng kiến thức và kỹ năng mà học phần mang lại, từ đó phục vụ cho mục tiêu học tập dài hạn hoặc định hướng nghề nghiệp. Trong khi đó, hai nhóm còn lại (DS và AM) duy trì số lượng câu hỏi ở mức trung bình, phản ánh nhu cầu vừa phải đối với loại thông tin này. Ở tình huống thứ ba, khi truy vấn tập trung vào nội dung chi tiết của học phần theo từng chủ đề, cả ba nhóm đều ghi nhận sự sụt giảm về số lượng câu hỏi. Kết quả này gợi ý rằng nhu cầu tìm hiểu sâu về nội dung cụ thể chưa thực sự phổ biến ở các nhóm sinh viên, ít nhất là trong giai đoạn tìm kiếm ban đầu. Tuy vậy, nhóm MCS vẫn giữ mức độ quan tâm cao hơn tương đối so với hai nhóm còn lại, trong khi nhóm DS và AM có xu hướng giảm nhẹ về mức độ tương tác.

Bộ kiểm tra multihop

Bộ multihop là thách thức lớn nhất trong ba bộ dữ liệu, không chỉ đòi hỏi mô hình phải có năng lực truy xuất chính xác, mà còn phải có khả năng diễn giải mối quan hệ logic giữa các đơn vị tri thức trong toàn bộ chương trình đào tạo. Như đã đề cập ở trên, số lượng câu hỏi cho bộ multihop sẽ được lựa chọn giống nhau về tổng số câu hỏi cho từng tình huống nghiên cứu. Đối với tình huống số 4, 5, 6, 7, 8 thì mỗi tình huống sẽ có 5 câu; tình huống 9 sẽ là 4 câu và tình huống 10 sẽ là 1 câu. Lý giải cho tình huống số 10, bởi vì tập kiểm tra closed_end đã có các câu hỏi đánh giá khá tổng quát về thông tin các nút của tên các giảng viên (thông qua tình huống số 1) nên chúng tôi muốn tập trung hơn về các tình huống từ số 4 đến số 9.

C. Chi phí thực nghiệm cho Phần 5.2.2

Trong phần này, chúng tôi sẽ tập trung phân tích chi phí tài nguyên phần cứng khi tự vận hành[3] ba mô hình nhúng phổ biến gồm GritLM-7B, Nvidia NV-Embed-v2, và OpenAI text-embedding-3-small (trong đó mô hình của OpenAI được truy cập thông qua API trả phí). Mục tiêu là đưa ra một cái nhìn thực tế về mức độ đầu tư cần thiết cho từng lựa chọn, từ đó hỗ trợ các nhà phát triển hoặc tổ chức đưa ra quyết định phù hợp với nhu cầu và điều kiện hạ tầng cụ thể.

Chi phí tài nguyên phần cứng

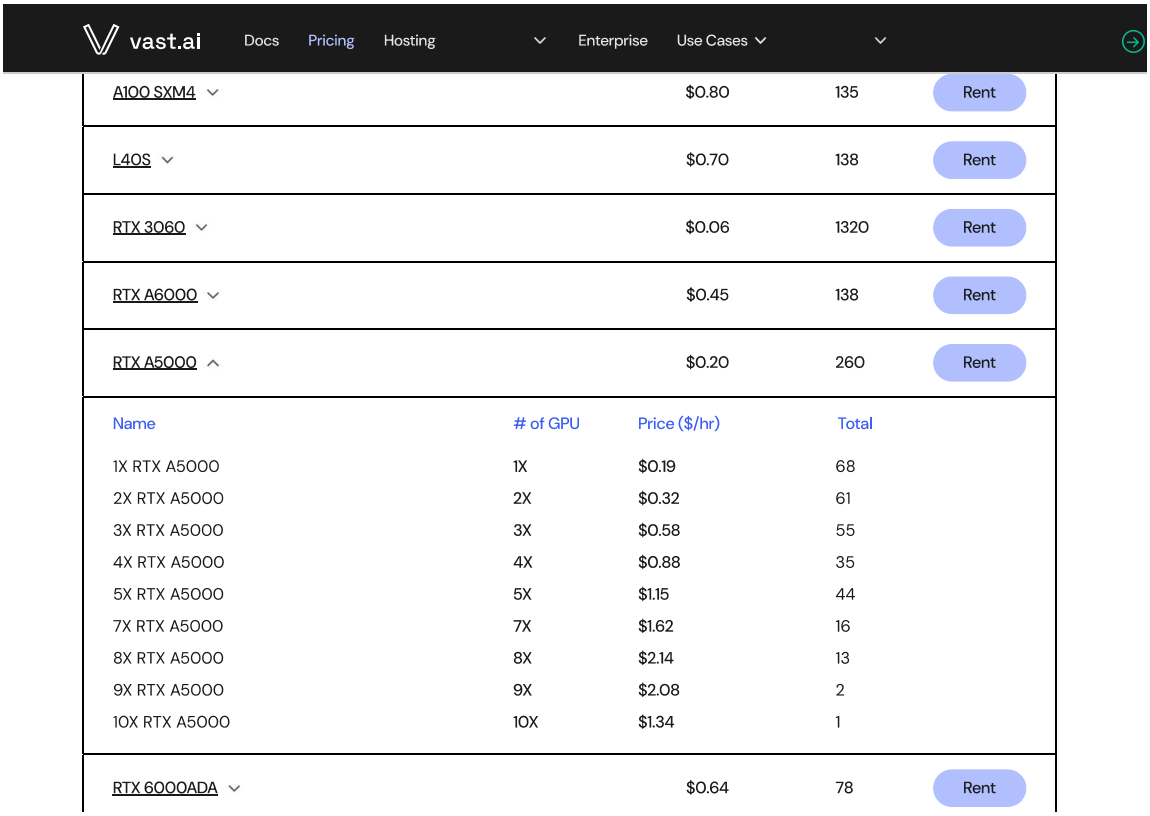
Trong quá trình triển khai thực nghiệm nhằm đánh giá hiệu năng giữa Simple RAG và Graph-based RAG, một thách thức hiện hữu không chỉ đến từ độ chính xác truy xuất mà còn đến từ chi phí tài nguyên phần cứng khi vận hành các mô hình nhúng quy mô lớn. Để đảm bảo tính công bằng và đồng nhất trong các thí nghiệm, chúng tôi lựa chọn triển khai self-host ba mô hình nhúng phổ biến gồm Contriever, GritLM-7B, và NV-Embed-v2 – đại diện cho ba thế hệ mô hình khác nhau về quy mô và độ phức tạp.

Trong số đó, Contriever tỏ ra nhẹ nhàng nhất khi chỉ chiếm dụng khoảng 438MB bộ nhớ, cho phép triển khai nhanh chóng trên các máy trạm phổ thông. Tuy nhiên, khi tiến đến hai mô hình còn lại, chúng tôi sớm nhận ra một rào cản đáng kể về phần cứng. Cụ thể, GritLM-7B, một mô hình ngôn ngữ mã nguồn mở do IBM phát triển, yêu cầu bộ nhớ VRAM khoảng 15GB (Nomic AI, 2024), trong khi NV-Embed-v2 của NVIDIA cũng có mức tiêu thụ tương tự (NVIDIA, 2024), lên đến xấp xỉ 15GB. Đây đều là các mô hình nhúng mạnh mẽ, nhưng đi kèm theo đó là yêu cầu cao về kiến trúc phần cứng nếu muốn vận hành một cách mượt mà và ổn định.

Để đáp ứng yêu cầu này, chúng tôi khuyến nghị sử dụng card đồ họa NVIDIA RTX A5000 – một lựa chọn phù hợp cho các tác vụ AI cường độ cao. RTX A5000 được trang bị 24GB bộ nhớ GDDR6 với ECC, 8,192 nhân CUDA và băng thông bộ nhớ lên đến 768 GB/s, đảm bảo khả năng xử lý mô hình lớn mà không bị nghẽn tài nguyên. Tuy nhiên, mức đầu tư cho phần cứng này là không nhỏ – giá thị trường hiện dao động từ 3,721 đến 4,000 USD cho mỗi chiếc, chưa bao gồm chi phí máy chủ, tản nhiệt, hay vận hành dài hạn.

Đứng trước bài toán tối ưu chi phí, chúng tôi đã thử nghiệm một hướng tiếp cận khác: thuê máy chủ GPU từ nền tảng vast.ai (Vast.ai, 2025). Đây là một nền tảng cho phép người dùng thuê GPU từ các nhà cung cấp phân tán trên toàn thế giới với chi phí linh hoạt, phù hợp cho các dự án nghiên cứu hoặc nhu cầu tính toán ngắn hạn. Trong thực nghiệm, chúng tôi lựa chọn gói cấu hình 1×GPU RTX A5000, với mức giá trung bình 0.19 USD/giờ (Hình 6.4). Quá trình thực nghiệm được tiến hành trên cả ba bộ dữ liệu đầu vào xuyên suốt trong 24 giờ, từ đó tổng chi phí thuê GPU là 4.56 USD. Với mức chi phí này, chúng tôi có thể triển khai toàn bộ quy trình thử nghiệm của hệ thống mà không cần đầu tư hàng nghìn đô la cho phần cứng chuyên dụng, đồng thời vẫn đảm bảo khả năng chạy mượt các mô hình nhúng nặng như GritLM và NV-Embed-v2. Điều này mở ra một hướng đi thực tiễn cho các nhóm nghiên cứu, tổ chức giáo dục hoặc startup

[3] self-host



Hình 6.4: Bảng giá thuê máy chủ GPU thông qua dịch vụ vast.ai.

trong việc triển khai mô hình RAG ở quy mô vừa và nhỏ mà không bị giới hạn bởi chi phí phần cứng ban đầu.

Chi phí truy xuất API từ dịch vụ OpenAI

Bên cạnh phương án tự triển khai mô hình nhúng và mô hình ngôn ngữ trong nội bộ, chúng tôi cũng thử nghiệm một lựa chọn khác là sử dụng dịch vụ API do OpenAI cung cấp. Cách tiếp cận này mang lại nhiều ưu điểm rõ rệt về mặt triển khai, đặc biệt là trong việc rút ngắn thời gian khởi tạo hệ thống, giảm thiểu rủi ro về hạ tầng và giúp tập trung tối đa vào thiết kế logic của hệ tư vấn học phần. Trong kịch bản thử nghiệm này, chúng tôi sử dụng hai mô hình từ OpenAI: text-embedding-3-small để mã hóa văn bản và gpt-4o-mini để sinh câu trả lời từ ngữ cảnh được truy xuất.

Với mô hình nhúng text-embedding-3-small, tổng số token đầu vào mà hệ thống đã xử lý là 4,059,210 tokens. Theo biểu phí chính thức được công bố trên trang web OpenAI tại thời điểm tháng 6 năm 2025 (Hình 6.6), mức giá dành cho mô hình này là 0.0002 USD cho mỗi 1,000 tokens, tương đương 0.02 USD trên mỗi 1 triệu tokens. Như vậy, tổng chi phí cho giai đoạn nhúng văn bản là khoảng 0.081 USD, một con số rất thấp nếu so sánh với chi phí triển khai phần cứng tương ứng cho các mô hình nhúng lớn hơn (OpenAI, 2025).

Đối với giai đoạn tạo câu trả lời, chúng tôi sử dụng mô hình gpt-4o-mini, một phiên bản ngôn ngữ nhỏ gọn nhưng vẫn đảm bảo độ chính xác và khả năng hiểu ngữ cảnh. Trong quá trình thực nghiệm, tổng số token đầu vào được gửi đến API là 37,827,250

OpenAI Platform

Pricing

Latest models

New: Save on synchronous requests with flex processing.

Text tokens — Price per 1M tokens · Batch API price

Model	Input	Cached input	Output
gpt-4.1 ↳ gpt-4.1-2025-04-14	$2.00	$0.50	$8.00
gpt-4.1-mini ↳ gpt-4.1-mini-2025-04-14	$0.40	$0.10	$1.60
gpt-4.1-nano ↳ gpt-4.1-nano-2025-04-14	$0.10	$0.025	$0.40
gpt-4.5-preview ↳ gpt-4.5-preview-2025-02-27	$75.00	$37.50	$150.00
gpt-4o ↳ gpt-4o-2024-08-06	$2.50	$1.25	$10.00
gpt-4o-audio-preview ↳ gpt-4o-audio-preview-2024-12-17	$2.50	-	$10.00
gpt-4o-realtime-preview ↳ gpt-4o-realtime-preview-2024-12-17	$5.00	$2.50	$20.00
gpt-4o-mini ↳ gpt-4o-mini-2024-07-18	$0.15	$0.075	$0.60
gpt-4o-mini-audio-preview ↳ gpt-4o-mini-audio-preview-2024-12-17	$0.15	-	$0.60

Hình 6.5: Chi phí sử dụng API từ OpenAI cho mô hình ngôn ngữ gpt-4o-mini.

Embeddings

Price per 1M tokens · Batch API price

Model	Cost
text-embedding-3-small	$0.02
text-embedding-3-large	$0.13
text-embedding-ada-002	$0.10

Hình 6.6: Chi phí sử dụng API từ OpenAI cho mô hình nhúng text-embedding-3-small.

tokens, và số token đầu ra do mô hình sinh ra là 6,924,323 tokens. Theo mức giá hiện hành, chi phí cho token đầu vào là 0.15 USD trên mỗi 1 triệu tokens, trong khi token đầu ra được tính ở mức 0.60 USD trên mỗi 1 triệu tokens (Hình 6.5). Từ đó, tổng chi phí sử dụng gpt-4o-mini là 9.829 USD.

D. Đường link tới repo của hệ thống HippoRAG-2

https://github.com/OSU-NLP-Group/HippoRAG/tree/main