

**BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC NÔNG LÂM TP HCM
KHOA CÔNG NGHỆ THÔNG TIN**



LUẬN VĂN TỐT NGHIỆP
**Xây dựng chatbot tư vấn học vụ Nông
Lâm kết hợp giữa hệ thống GRAG (Graph
Retrieval-Augmented Generation) và RAG
(Retrieval-Augmented Generation)**

Ngành : Công nghệ thông tin
Niên khóa : 2021 - 2025
Lớp : DH21DTC
Sinh viên thực hiện : Cao Thành Nam - 21130448
Nguyễn Việt Pha - 21130467

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC NÔNG LÂM TP HCM
KHOA CÔNG NGHỆ THÔNG TIN



LUẬN VĂN TỐT NGHIỆP

**Xây dựng chatbot tư vấn học vụ Nông
Lâm kết hợp giữa hệ thống GRAG (Graph
Retrieval-Augmented Generation) và RAG
(Retrieval-Augmented Generation).**

Giảng viên hướng dẫn:

ThS. Nguyễn Đức Công Song

Sinh viên thực hiện:

Cao Thành Nam - 21130448

Nguyễn Việt Pha - 21130467

TP. HỒ CHÍ MINH, ngày 10 tháng 02 năm 2025

Xây dựng chatbot tư vấn học vụ Nông Lâm kết hợp giữa hệ thống GRAG (Graph Retrieval-Augmented Generation) và 2025

CÔNG TRÌNH HOÀN TẤT TẠI
TRƯỜNG ĐẠI HỌC NÔNG LÂM TP. HỒ CHÍ
MINH

Cán bộ hướng dẫn: **Th.S Nguyễn Đức Công Song**

Cán bộ phản biện: **TS Lê Phi Hùng**

Luận văn cử nhân được bảo vệ tại **HỘI ĐỒNG CHẤM LUẬN VĂN CỬ NHÂN**
TRƯỜNG ĐẠI HỌC NÔNG LÂM TP HCM ngày 10 tháng 02 năm 2025

Nhân xét của giáo viên hướng dẫn

Nhận xét của giáo viên phản biện

NHIỆM VỤ LUẬN VĂN CỬ NHÂN

Họ tên sinh viên: **Cao Thành Nam**

Phái: **Nam**

Ngày tháng năm sinh: 05/12/2003

Nơi sinh: Đồng Nai

Chuyên ngành: Công Nghệ Thông Tin

Điện thoại liên lạc: 0839060487

Email: 21130448@st.hcmuaf.edu.vn

Họ tên sinh viên: **Nguyễn Việt Pha**

Phái: **Nam**

Ngày tháng năm sinh: 10/04/2003

Nơi sinh: TPHCM

Chuyên ngành: Công Nghệ Thông Tin

Điện thoại liên lạc: 0982352578

Email: 21130467@st.hcmuaf.edu.vn

I. TÊN ĐỀ TÀI: Xây dựng chatbot tư vấn học vụ Nông Lâm kết hợp giữa hệ thống GRAG (Graph Retrieval-Augmented Generation) và RAG (Retrieval-Augmented Generation).

II. NHIỆM VỤ VÀ NỘI DUNG

- Nhiệm vụ: Cung cấp dịch vụ phục vụ cho việc hỏi đáp về học vụ Nông Lâm bằng ngôn ngữ tiếng Việt và xây dựng một trang web để sử dụng dịch vụ này.
- Nội dung:
 - + Nghiên cứu và kết hợp RAG và GRAG để truy xuất dữ liệu nội bộ.
 - + Xây dựng hệ thống Agenic GRAG để có thể tự động hóa và theo dõi qua trình lý luận.

III. NGÀY GIAO NHIỆM VỤ: 30/01/2025

IV. NGÀY HOÀN THÀNH NHIỆM VỤ: 26/08/2025

V. HỌ VÀ TÊN CÁN BỘ HƯỚNG DẪN: Th.S Nguyễn Đức Công Song

Ngày / /
CÁN BỘ HƯỚNG DẪN
(Ký và ghi rõ họ tên)

Ngày / /
CÁN BỘ PHẢN BIỆN
(Ký và ghi rõ họ tên)

Ngày / /
KHOA CNTT
(Ký và ghi rõ họ tên)

LỜI CẢM ƠN

Chúng em xin chân thành cảm ơn các thầy cô khoa Công nghệ thông tin trường Đại học Nông Lâm TP.Hồ Chí Minh, với những kiến thức quý báu và sự tận tâm, nhiệt huyết mà thầy cô đã truyền đạt cho chúng em trong suốt những năm đại học.

Chúng con xin gửi đến cha mẹ lời ghi ơn sâu sắc, những người đã sinh ra và dạy bảo chúng con trưởng thành như ngày hôm nay.

Đặc biệt, chúng em xin chân thành cảm ơn Thầy Nguyễn Đức Công Song đã tận tình hướng dẫn, chỉ bảo và giúp đỡ chúng em trong suốt quá trình thực hiện đề tài nghiên cứu này. Cảm ơn vì thầy đã truyền tải cho chúng em không chỉ về kiến thức chuyên ngành mà còn cả về cách sống, cách ứng xử, cách suy nghĩ giúp chúng em trưởng thành hơn.

Cũng xin cảm ơn tất cả những bạn bè đã chia sẻ kiến thức và tận tình giúp đỡ chúng em hoàn thành đề tài này.

Trong quá trình thực hiện đề tài nghiên cứu, mặc dù chúng em đã có những cố gắng nỗ lực thực hiện nhưng chúng em chắc không thể tránh được những sai sót nhất định. Kính mong sự thông cảm và tận tình chỉ bảo của quý Thầy Cô. Xin chân thành cảm ơn mọi người.

TP. HCM, ngày 05 tháng 09 năm 2025
NHÓM THỰC HIỆN LUẬN VĂN TỐT NGHIỆP

DANH MỤC CÁC HÌNH

Hình 2.1 Kiến trúc RAG truyền thống.....	11
Hình 2.2 Kết hợp kết quả từ nhiều truy vấn.....	18
Hình 2.3 Quá trình chunking theo kích thước cố định.....	29
Hình 2.4 Quá trình chunking theo ngũ nghĩa.....	31
Hình 2.5 Quá trình chunking dựa trên LLM.....	32
Hình 2.6 Câu cypher lấy ra đồ thị “quá trình hình thành và phát triển”	37
Hình 2.7 Quy trình tạo ra nhúng của mô hình 1024 chiều.....	40
Hình 2.8 Quy trình tạo ra nhúng của mô hình 768 chiều.....	42
Hình 2.9 Quy trình tạo ra nhúng của mô hình 512 chiều.....	43
Hình 2.10 Quy trình tạo ra nhúng của mô hình late interaction	46
Hình 3.1 Kiến trúc chính của hệ thống	48
Hình 3.2 Quy trình thực hiện	50
Hình 3.3 Quy trình tiền xử lý dữ liệu.....	53
Hình 4.1 Các bước thực hiện hệ thống	55
Hình 4.2 Biểu đồ so sánh độ chính xác của các mô hình.....	58
Hình 4.3 Thống kê số lượng câu hỏi đúng và sai của các mô hình	59
Hình 4.4 Biểu đồ độ chính xác của mô hình rerank.....	60
Hình 4.5 Thống kê số lượng câu hỏi đúng và sai của các mô hình	60
Hình 4.6 Tổ chức dữ liệu đồ thị của phần 1	66
Hình 4.7 Tổ chức dữ liệu đồ thị của phần 2 và 3	67
Hình 4.8 Biểu đồ histogram thống kê phân phối độ tương đồng.....	79
Hình 4.9 Biểu đồ violin thống kê phân phối chi tiết độ tương đồng	80
Hình 4.10 Kiến trúc ứng dụng	82
Hình 4.12 Lược đồ Usecase cho chức năng quản lý tài liệu.....	89
Hình 4.13 Lược đồ Usecase cho chức năng chat	94

DANH MỤC CÁC BẢNG

Bảng 2.1 So sánh giữa các phương pháp chunking	34
Bảng 2.2 So sánh giữa RAG và GRAG	38
Bảng 4.1 So sánh độ tương đồng trung bình của ba kiến trúc	78
Bảng 4.2 Bảng thống kê Accuracy, Halluc và Missing	81
Bảng 4.3 Mô tả Usecase cho chức năng đăng ký tài khoản.....	84
Bảng 4.4 Mô tả Usecase cho chức năng đăng nhập.....	85
Bảng 4.5 Mô tả Usecase cho chức năng đăng xuất.....	86
Bảng 4.6 Mô tả Usecase cho chức năng quên mật khẩu.....	87
Bảng 4.7 Mô tả Usecase cho chức năng quản lý hồ sơ.....	88
Bảng 4.8 Mô tả Usecase cho chức năng thống kê	89
Bảng 4.9 Mô tả Usecase cho chức năng upload tài liệu	90
Bảng 4.10 Mô tả Usecase cho chức năng trích xuất văn bản	91
Bảng 4.11 Mô tả Usecase cho chức năng lưu trữ trên S3	91
Bảng 4.12 Mô tả Usecase cho chức năng tạo chunks văn bản	92
Bảng 4.13 Mô tả Usecase cho chức năng tạo thực thể và quan hệ	94
Bảng 4.14 Mô tả Usecase cho chức năng tạo cuộc hội thoại mới	95
Bảng 4.15 Mô tả Usecase cho chức năng tạo chat với tài liệu	96
Bảng 4.16 Mô tả Usecase cho chức năng gửi tin nhắn	97
Bảng 4.17 Mô tả Usecase cho chức năng xem lịch sử chat	98
Bảng 4.18 Mô tả Usecase cho chức năng xóa cuộc hội thoại.....	99
Bảng 4.19 Mô tả Usecase cho chức năng lưu cuộc hội thoại	100

TÓM TẮT

RAG ra đời như một cứu tinh cho nhiệm vụ sinh văn bản đối với các dữ liệu mà LLMs chưa được huấn luyện, bằng cách truy xuất vào nguồn dữ liệu bên ngoài để lấy thêm ngữ cảnh cho câu trả lời và LLMs sẽ sử dụng ngữ cảnh đó và đưa ra câu trả lời cuối cùng, lợi ích của việc này nhằm giảm bớt chi phí huấn luyện và ảo tưởng trong quá trình sinh văn bản. Tuy nhiên, RAG vẫn còn gặp nhiều hạn chế trong nhiệm vụ QFS, ngoài ra còn khó khăn trong việc chia các chunk làm sao để một đoạn vẫn giữ được nguyên vẹn thông tin mà không bị ngắt giữa chừng.

Để giải quyết các tồn đọng của RAG, một nghiên cứu về GRAG đã ra đời, bằng cách tận dụng khả năng của LLMs để trích xuất các thực thể và các mối quan hệ trong một đoạn, từ đó xây dựng được một KG mà LLMs có thể truy xuất để đưa ra câu trả lời một cách tổng quan hơn. Điều này đã chứng minh hiệu suất cải thiện đáng kể so với phương pháp RAG cơ bản về cả tính toàn diện và tính đa dạng của các câu trả lời được tạo ra.

Agent AI đang là SoTa trong NLP, có khả năng tự động hóa trong việc lý luận, quan sát và hành động giống như con người. Ngoài kết hợp RAG và GRAG, chúng em còn sử dụng LLM để thực hiện ba việc trên nhằm đưa ra câu trả lời tốt nhất.

ABSTRACT

RAG emerged as a savior for text generation tasks with data that LLMs have not been trained on, by retrieving external data sources to provide additional context for the answers. The LLMs then use this context to produce the final answer. The benefit of this approach is to reduce training costs and mitigate hallucinations during text generation. However, RAG still faces several limitations in the QFS task, and there are challenges in chunking data in a way that preserves the integrity of the information without cutting it off midway.

To address the shortcomings of RAG, research into GRAG has been developed, leveraging the ability of LLMs to extract entities and relationships within a passage, thereby building a KG that LLMs can access to provide more comprehensive answers. This has shown a significant improvement in performance compared to the basic RAG method, in terms of both the comprehensiveness and diversity of the generated answers.

Agent AI is currently SoTA in NLP, capable of automating reasoning, observation, and action in a human-like manner. In addition to combining RAG and GRAG, we also utilize LLMs to perform these three tasks to provide the best possible answers.

DANH SÁCH CHỮ VIẾT TẮT

Từ viết tắt	Từ tiếng anh	Từ tiếng việt
AI	Artificial Intelligence	Trí tuệ nhân tạo
API	Application Programming Interface	Giao diện lập trình ứng dụng, cho phép các ứng dụng giao tiếp với nhau và thường có giới hạn về số lượng yêu cầu
BE	Backend	Cách hoạt động của hệ thống
CLS	Classifier token (trong BERT)	Token đặc biệt đại diện cho toàn bộ câu, được thêm vào đầu câu khi tạo embedding
DBMS	Database Management System	Hệ thống quản lý cơ sở dữ liệu
FE	Frontend	Cách tương tác và trải nghiệm với hệ thống
GRAG	Graph Retrieval-Augmented Generation	Sinh truy xuất tăng cường đồ thị
HNSW	Hierarchical Navigable Small World	Thuật toán tìm kiếm vector tương tự hiệu quả được sử dụng trong Qdrant cho Vector Index, có cấu trúc đa lớp

HTTP	Hypertext Transfer Protocol	Giao thức truyền tải dữ liệu trên web, ví dụ "HTTP 500" là mã lỗi máy chủ
IR	Information Retrieval	Truy xuất thông tin
JSON	JavaScript Object Notation	Định dạng dữ liệu dùng để chứa thông tin meta hoặc làm định dạng trả về dữ liệu trích xuất
KG	Knowledge Graph	Đồ thị tri thức
LLM	Large Language Model	Mô hình ngôn ngữ lớn
LLMs	Large Language Models	Các mô hình ngôn ngữ lớn
LSTM	Long Short-Term Memory	Kiến trúc mạng nơ-ron tương tự RNN, cũng có hạn chế đã được Transformer khắc phục
NLG	Natural Language Generation	Sinh ngôn ngữ tự nhiên
NLP	Natural Language Processing	Xử lý ngôn ngữ tự nhiên
QA	Question-Answering	Hỏi và trả lời
QFS	Query-Focused Summarization	Tóm tắt tập trung vào truy vấn

RAG	Retrieval-Augmented Generation	Sinh truy xuất tăng cường
RNN	Recurrent Neural Network	Kiến trúc mạng nơ-ron đã cũ, có hạn chế mà Transformer đã vượt qua
SEP	Separator token	Token đặc biệt đánh dấu kết thúc câu, được thêm vào cuối câu khi tạo embedding
SKB	Semi-structured Knowledge Bases	Cơ sở kiến thức bán cấu trúc
SOTA	State-Of-The-Art	Tiên tiến nhất
TF-IDF	Term Frequency-Inverse Document Frequency	Thuật toán biểu diễn văn bản dưới dạng vector số, mã hóa ý nghĩa ngữ nghĩa để xử lý ngôn ngữ hiệu quả
token	token	Đơn vị cơ bản sau khi tách văn bản để xử lý
vector	vector	Dạng biểu diễn số của dữ liệu

MỤC LỤC

LỜI CẢM ƠN	VI
DANH MỤC CÁC HÌNH	VII
DANH MỤC CÁC BẢNG	VIII
TÓM TẮT	IX
DANH SÁCH CHỮ VIẾT TẮT	XI
MỤC LỤC	XIV
CHƯƠNG 1. MỞ ĐẦU	1
1.1 Lý do chọn đề tài	1
1.2 Các công trình nghiên cứu liên quan	2
1.3 Mục tiêu của đề tài	3
1.4 Đối tượng và phạm vi nghiên cứu	3
1.5 Kết quả cần đạt	4
CHƯƠNG 2. CƠ SỞ LÝ THUYẾT	5
2.1 Tổng quan về chatbot	5
2.1.1. Giới thiệu	5
2.1.2. Lịch sử hình thành	5
2.1.3. Phân loại chatbot	5
2.1.4. Học máy trong chatbot	6
2.1.5. Nhúng	6
2.2 Các chỉ số đánh giá mô hình	7
2.2.1. Hit@K	7
2.2.2. Recall@K	7
2.2.3. MRR	8

2.2.4. Cosine similarity	8
2.2.5. Hallucination	9
2.2.6. Accuracy.....	10
2.2.7. Missing	10
2.3 RAG (Retrieval-Augmented Generation).....	10
2.3.1. Các thành phần chính	11
2.3.2. Các phương pháp tối ưu RAG.....	13
2.3.3. Hạn chế của RAG truyền thống	34
2.4 GRAG(Graph Retrieval-Augmented Generation).....	35
2.4.1. Ưu điểm cốt lõi của GRAG.....	35
2.4.2. Neo4j	36
2.5 So sánh RAG và GRAG	38
2.6 Các mô hình nhúng được sử dụng	38
CHƯƠNG 3. PHƯƠNG PHÁP LUẬN	46
3.1 Phát biểu bài toán	46
3.2 Phương pháp giải quyết bài toán	47
3.3 Quy trình tương tác của hệ thống	49
3.4 Quy trình thực hiện.....	50
3.4.1. Chuẩn bị các mô hình gốc	50
3.4.2. Chuẩn bị tập dữ liệu	52
3.4.3. Tiền xử lý dữ liệu	52
3.4.4. Tạo ngân hàng dữ liệu	54
3.4.5. So sánh và đánh giá với kiến trúc khác	55
CHƯƠNG 4. TRIỂN KHAI THỰC HIỆN	55
4.1 Tổng quan hệ thống	55

4.2 Xây dựng nguồn kiến thức	55
4.2.1. Trích xuất dữ liệu cho lưu trữ vector	55
4.2.2. Trích xuất dữ liệu cho lưu trữ đồ thị	55
4.3 Xây dựng cơ sở dữ liệu vector Qdrant.....	55
4.3.1. Thiết kế mô hình dữ liệu cho Qdrant	56
4.3.2. Các bước thiết kế mô hình dữ liệu	56
4.3.3. Tạo embedding từ văn bản	58
4.3.4. Phát triển module tìm kiếm RAG.....	61
4.4 Xây dựng cơ sở dữ liệu đồ thị Neo4j.....	64
4.4.1. Thiết kế mô hình dữ liệu cho Neo4j.....	65
4.4.2. Phát triển module tìm kiếm GRAG.....	69
4.5 Xây dựng module phê bình(Critical Module)	72
4.6 Đánh giá tập dữ liệu ZaloAI	74
4.7 Đánh giá tập dữ liệu StaRK-Prime	77
4.8 Đánh giá và so sánh các kiến trúc khác nhau	77
4.8.1. Tập dữ liệu.....	77
4.8.2. Đánh giá	78
4.9 Xây dựng ứng dụng	82
4.9.1. Tổng quan ứng dụng	82
4.9.2. Thiết kế cơ sở dữ liệu	82
4.9.3. Các chức năng chính của ứng dụng	83
4.9.4. Môi trường triển khai ứng dụng	100
CHƯƠNG 5. KẾT QUẢ ĐẠT ĐƯỢC	101
5.1 Nghiên cứu.....	101
5.2 Sản phẩm	101

5.3 Kết quả thực nghiệm.....	101
CHƯƠNG 6. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN.....	102
6.1 Kết luận.....	102
6.2 Hướng phát triển.....	102
CHƯƠNG 7. TÀI LIỆU THAM KHẢO	103
PHỤ LỤC.....	94
1. Chi tiết các bước thực hiện	94
1.1. Trích xuất dữ liệu	94
1.2. Tiền xử lý dữ liệu	95
1.3. Thêm dữ liệu vào Qdrant	97
1.4. Thêm dữ liệu vào Neo4j.....	100
1.5. Xây dựng mô-đun phê bình.....	105
2. Ứng dụng	109
2.1. Lược đồ cơ sở dữ liệu hệ thống.....	109
2.2. Chức năng tạo tri thức từ tài liệu bên ngoài	111
2.3. Chức năng hỏi đáp số tay sinh viên.....	114
2.4. Màn hình ứng dụng website	116

CHƯƠNG 1. MỞ ĐẦU

1.1 Lý do chọn đề tài

Chúng em chọn đề tài “Xây dựng chatbot tư vấn học vụ Nông Lâm kết hợp giữa hệ thống GRAG (Graph Retrieval-Augmented Generation) và RAG (Retrieval-Augmented Generation)” nhằm đáp ứng các nhu cầu ngày càng tăng trong lĩnh vực NLP, đã nêu lên trong việc phân loại văn bản đến các nhiệm vụ phức tạp như tóm tắt, dịch tự động và trả lời câu hỏi. Một lĩnh vực đặc biệt quan trọng trong NLP là Tạo sinh ngôn ngữ tự nhiên NLG. Mục tiêu chính của NLG là giúp máy tính tạo ra văn bản mạch lạc và phù hợp với ngữ cảnh. Với sự tiến bộ của AI, yêu cầu đối với các mô hình tạo sinh nội dung có ngữ cảnh và căn cứ thực tế ngày càng tăng, dẫn đến những thách thức và cải tiến mới trong NLG [1]. Kiến trúc sequence-to-sequence, đã đạt được những bước tiến lớn trong việc tạo ra văn bản tự nhiên và mạch lạc. Tuy nhiên, các mô hình này phụ thuộc nhiều vào dữ liệu huấn luyện và gặp khó khăn khi phải sản xuất thông tin chính xác hay giàu ngữ cảnh trong những trường hợp yêu cầu kiến thức vượt ra ngoài phạm vi dữ liệu huấn luyện [1].

Để giải quyết vấn đề này đã có các nghiên cứu nói về hệ thống RAG hỗ trợ cập nhập dữ liệu mới mà không cần huấn luyện lại từ đầu bằng phương pháp trích xuất dữ liệu từ PDF sau đó lưu trữ trong cơ sở dữ liệu vector từ đó người dùng có thể truy xuất thông tin mà mô hình LLM chưa được huấn luyện.

Hệ thống RAG thông thường có nhược điểm sau:

- Bỏ qua các mối quan hệ trong dữ liệu và không thể trích xuất thông tin một cách tổng quát [2].
- Đối với các truy xuất yêu cầu tài liệu bán cấu trúc cần có giải pháp riêng, nhưng RAG chỉ truy xuất được các yêu cầu tài liệu không cấu trúc [2].

Đối với GRAG lại chỉ có thể truy xuất được dữ liệu có cấu trúc. Điều này tạo ra hai thách thức lớn sau:

- Cần phải tạo ra một hệ thống duy nhất vừa truy xuất được nguồn dạng không cấu trúc, cấu trúc và kết hợp [2].

- Thậm chí nếu xây dựng được hệ thống, đối với các câu hỏi phức tạp có nhiều phần. Hệ thống có thể trả lời sai ngay từ đầu do đó cần phải có module để phản hồi và nhận xét câu trả lời [2].

Các nghiên cứu gần đây đã chỉ ra rằng việc kết hợp giữa RAG và GRAG, cùng việc sử dụng LLM để tự động lý luận, quan sát và hành động tạo ra Agent AI có thể giải quyết hai vấn đề trên.

1.2 Các công trình nghiên cứu liên quan

Gemini 1.5 Flash là phiên bản mới của mô hình ngôn ngữ phát triển bởi Google DeepMind. Được huấn luyện khoảng 2 nghìn tỷ token, được lấy từ tài liệu, code và các nghiên cứu khoa học trên mạng, có khả năng xử lý 1048576 token cho mỗi chuỗi đầu vào, 8192 token đầu ra và có hiệu suất cao đối với các văn bản dài và thông tin phức tạp thứ mà rất có lợi cho mục đích của chúng em. Gemini 1.5 Flash cũng có hiệu suất tốt đối với tiếng việt vì được huấn luyện trên một lượng lớn dữ liệu đa ngôn ngữ [3].

Để khắc phục sự thiếu dữ liệu khi train và cập nhật dữ liệu mới nhất, RAG đã được công bố vào năm 2020. Bằng cách truy xuất vào cơ sở dữ liệu không cấu trúc bên ngoài LLMs sẽ cải tiến được khả năng trả lời câu hỏi với dữ liệu mới nhất. Tuy nhiên, một nhược điểm lớn nhất của RAG là việc chunking, dù chunking tốt đến thế nào thì cũng bị gây ra việc mất thông tin của đoạn và RAG không xử lý tốt nhiệm vụ QFS. GRAG được công bố để giải quyết hai vấn đề trên, thay vì sử dụng cơ sở dữ liệu không cấu trúc, GRAG sử dụng cơ sở dữ liệu có cấu trúc. Bằng cách tận dụng thế mạnh của LLMs trong việc trích xuất thực thể và những mối quan hệ trong một đoạn, chúng em có thể xây dựng đồ thị kiến thức, điều này giúp nhiệm vụ QA trở nên tổng quát hơn.

Trong một nghiên cứu có tên “ReAct: Synergizing Reasoning and Acting in Language Models” được công bố năm 2022, được xây dựng dựa trên cách suy nghĩ của con người, khi gặp một vấn đề đầu tiên sẽ lý luận(Reason), sau đó là hành động(Action) và cuối cùng là quan sát(Observation) [4]. Xây dựng theo mô hình này giúp LLMs có khả năng lý luận và điều chỉnh hành vi một cách tự động, từ đó có thể xem cách lý luận theo trình tự và đưa ra câu trả lời cuối cùng tốt nhất.

Agent-G cũng được xây dựng theo logic của ReAct, được truy xuất vào wikipedia Agent-G sẽ xây dựng hai cơ sở dữ liệu bên ngoài không cấu trúc và có cấu trúc, kèm theo một critical module để có thể tự quyết định hành động.

Hai bài báo này đã chứng minh khả năng tự suy luận, hành động dựa trên khả năng LLMs có hiệu suất vượt trội hơn các mô hình thông thường khác.

1.3 Mục tiêu của đề tài

- + Sử dụng LLM Gemini-1.5-flash để làm base model.
- + Tạo ra hệ thống kết hợp giữa RAG và GRAG xây dựng hệ thống chatbot tư vấn học vụ Nông Lâm.
- + Đánh giá hệ thống trên các tập dataset StaRK-MAG, StaRK-Prime, CRAG và ZaloAI sử dụng các metric Hit@1, Hit@5, Recall@20m, MRR, Person Cosine và Spearman Cosine để so sánh với mô hình của đề tài trước. Chúng minh rằng kiến trúc này có khả năng trả lời câu hỏi tốt hơn kiến trúc của mô hình cũ.

1.4 Đối tượng và phạm vi nghiên cứu

Đối tượng nghiên cứu: Đề tài tập trung nghiên cứu xung quanh các đối tượng sau:

- + Nghiên cứu mô hình LLM Gemini-1.5-flash.
- + Nghiên cứu RAG và GRAG.
- + Nghiên cứu ba dataset StaRK-MAG, StaRK-Prime, CRAG và ZaloAI để đánh giá hệ thống sử dụng Hit@1, Hit@5, Recall@20m, MRR, Person Cosine và Spearman Cosine.
- + Nghiên cứu framework LangChain.
- + Nghiên cứu thư viện pyvis.
- + Nghiên cứu vector database Qdrant

Phạm vi nghiên cứu:

- + Dùng LLM Gemini-1.5-flash để làm base model.
- + Áp dụng dataset StaRK-MAG, StaRK-Prime, CRAG và ZaloAI để đánh giá hiệu suất của hệ thống sử dụng các metric Hit@1, Hit@5, Recall@20m, MRR, Person Cosine và Spearman Cosine.

- + Áp dụng dataset học vụ Nông Lâm trên Huggingface và bổ sung thêm dữ liệu nếu cần thiết.
- + Áp dụng kết hợp hệ thống GRAG và RAG để xây dựng chatbot tư vấn học vụ Nông Lâm.
- + Áp dụng framework Langchain giúp xây dựng model LLM dễ dàng.
- + Áp dụng thư viện pyvis để vẽ Knowledge Graph.
- + Áp dụng vector database Qdrant để lưu trữ các embedding xây dựng retriever bank.

1.5 Kết quả cần đạt

- + Hệ thống chatbot giúp hỗ trợ sinh viên thực hiện tư vấn học vụ Nông Lâm.
- + Đưa ra được tài liệu chi tiết áp dụng được model LLM Gemini-1.5-flash làm base model trong hệ thống kết hợp RAG và GRAG và lưu trữ trên vector database để xây dựng được chatbot truy vấn dữ liệu.

CHƯƠNG 2. CƠ SỞ LÝ THUYẾT

2.1 Tổng quan về chatbot

2.1.1. Giới thiệu

Chatbot là những chương trình máy tính được thiết kế để mô phỏng giao tiếp của con người thông qua ngôn ngữ tự nhiên, giúp tự động hóa các tác vụ liên quan đến ngôn ngữ tự nhiên, tạo điều kiện tương tác hiệu quả giữa máy tính và con người [5] [6]. Công nghệ chatbot bắt đầu từ những năm 1960 với ELIZA, một chatbot cơ bản phản hồi theo kịch bản định sẵn [5] [7]. Đến năm 2025, chatbot được dự đoán sẽ trở thành công cụ không thể thiếu trong nhiều lĩnh vực như dịch vụ khách hàng và tự động hóa quy trình kinh doanh [5].

2.1.2. Lịch sử hình thành

Công nghệ chatbot bắt đầu với ELIZA do Joseph Weizenbaum phát triển tại MIT vào giữa những năm 1960, mô phỏng nhà trí liệu tâm lý theo phương pháp Rogerian, sử dụng quy tắc khớp mẫu và thay thế từ khóa mà không hiểu ngôn ngữ thực sự [5] [7]. Bước đột phá lớn nhất là năm 2017 với kiến trúc Transformer với bài báo "Attention is All You Need", cho phép xử lý toàn bộ đầu vào cùng lúc, vượt qua hạn chế của RNN và LSTM, trở thành nền tảng cho các mô hình ngôn ngữ lớn hiện nay [5].

2.1.3. Phân loại chatbot

Chatbot được phân loại thành hai nhóm chính:

Loại chatbot	Cơ chế hoạt động	Ưu điểm	Hạn chế
Chatbot dựa trên quy tắc	Hoạt động theo kịch bản, nhận diện từ khóa, cây quyết định	Dễ triển khai, chi phí thấp, hành vi dự đoán được	Không học hỏi được, thiếu linh hoạt, xử lý truy vấn phức tạp hạn chế [5] [8] [9]
Chatbot tăng cường bởi AI	Sử dụng AI, học máy, NLP để hiểu	Xử lý truy vấn phức tạp, cải thiện liên tục	Chi phí cao, phức tạp phát triển, cần đào tạo liên tục, có

	và phản hồi linh hoạt		thể hiêu sai ý định, phụ thuộc dữ liệu [5] [8] [9]
--	-----------------------	--	--

2.1.4. Học máy trong chatbot

Học có giám sát: Chatbot được huấn luyện trên dữ liệu đã gán nhãn (cặp đầu vào - đầu ra), giúp đạt độ chính xác cao trong phân loại và dự đoán, nhưng đòi hỏi nhiều dữ liệu chất lượng cao, tốn thời gian và chi phí [5].

Học không giám sát: Chatbot học từ dữ liệu chưa gán nhãn, sử dụng phân cụm hoặc mô hình xác suất để phát hiện mẫu ngữ nghĩa, hiệu quả với dữ liệu lớn, tự động nhận diện ý định và trích xuất thực thể, nhưng độ chính xác thấp hơn và cần nhiều dữ liệu, chi phí huấn luyện cao [5].

2.1.5. Nhúng

Nhúng (embedding) là kỹ thuật biểu diễn từ, cụm từ hoặc văn bản dưới dạng vector số trong không gian đa chiều, mã hóa ý nghĩa ngữ nghĩa và ngữ pháp, giúp xử lý ngôn ngữ hiệu quả [5].

Word2Vec (Google, 2013): Dùng mạng nơ-ron Skip-gram hoặc CBOW để tạo vector từ, biểu diễn mối quan hệ ngữ nghĩa hiệu quả (ví dụ: "King - Man + Woman = Queen"), độ chính xác cao, chi phí tính toán thấp, nhúng hàng tỷ từ nhanh [5]. Hạn chế là vector cố định, không hiểu sâu ngữ cảnh.

GloVe (Stanford, 2014): Dựa trên ma trận đồng xuất hiện toàn cục, kết hợp thông tin ngữ cảnh cục bộ và toàn cục, chính xác với từ phổ biến, biểu diễn mối quan hệ tuyến tính trong vector [5]. Cũng là mô hình ngữ cảnh tĩnh.

Nhúng ngôn ngữ theo hướng ngữ cảnh động:

- Unidirectional Embeddings: Chỉ xem xét ngữ cảnh trước để dự đoán từ tiếp theo, ví dụ GPT ban đầu [5].
- Bidirectional Embeddings: Xem xét cả trước và sau, như BERT, RoBERTa, giúp hiểu sâu hơn ý nghĩa từ trong ngữ cảnh toàn diện, nâng cao hiệu suất mô hình [5].

2.2 Các chỉ số đánh giá mô hình

2.2.1. Hit@K

Hit@K đo lường liệu có ít nhất một mục liên quan xuất hiện trong K kết quả hàng đầu của mô hình hay không. Đây là chỉ số nhị phân, thường được dùng trong các hệ thống đề xuất hoặc tìm kiếm để kiểm tra xem mô hình có đưa ra kết quả đúng ngay trong những lựa chọn đầu tiên hay không.

Công thức tính trung bình trên Q truy vấn:

$$\text{Hit@K} = \frac{1}{Q} \sum_{i=1}^Q \text{Hit@K}_i$$

Trong nghiên cứu này chúng em sử dụng Hit@1 và Hit@5, cụ thể:

Hit@1: Kiểm tra xem kết quả đầu tiên có liên quan hay không.

Hit@5: Kiểm tra xem có ít nhất một mục liên quan trong 5 kết quả đầu tiên hay không.

Ví dụ: Trong một chatbot trả lời câu hỏi, nếu câu trả lời đúng nằm ở vị trí đầu tiên, Hit@1 = 1. Nếu câu trả lời đúng nằm ở vị trí thứ 3, Hit@1 = 0 nhưng Hit@5 = 1.

2.2.2. Recall@K

Recall@K đo lường tỷ lệ các mục liên quan được tìm thấy trong K kết quả hàng đầu so với tổng số mục liên quan có thể có. Chỉ số này tập trung vào việc đảm bảo mô hình thu hồi được càng nhiều mục liên quan càng tốt.

Công thức tính trên một truy vấn:

$$\text{Recall@K} = \frac{\text{Số mục liên quan trong top K}}{\text{Tổng số mục liên quan}}$$

Trong nghiên cứu này chúng em sử dụng Recall@20, cụ thể:

Recall@20: Tính tỷ lệ các mục liên quan trong 20 kết quả đầu tiên so với tất cả các mục liên quan.

Ví dụ: Nếu có 10 câu trả lời liên quan cho một câu hỏi và chatbot trả về 7 câu trong top 20, thì Recall@20 = $\frac{7}{10} = 0.7$.

2.2.3. MRR

MRR (Mean Reciprocal Rank) đo lường trung bình nghịch đảo của vị trí xếp hạng của mục liên quan đầu tiên trong danh sách kết quả. Chỉ số này tập trung vào việc đánh giá xem mô hình xếp hạng mục đúng sớm như thế nào.

Xếp Hạng Nghịch Đảo (Reciprocal Rank - RR):

$$RR = \frac{1}{\text{Vị trí của mục liên quan đầu tiên}}$$

Công thức tính trung bình trên Q truy vấn:

$$MRR = \frac{1}{Q} \sum_{i=1}^Q \frac{1}{\text{Vị trí của mục liên quan đầu tiên cho truy vấn } i}$$

Ví dụ: Nếu chatbot trả lời đúng ở vị trí 1 cho truy vấn A ($RR = \frac{1}{1} = 1$) và vị trí 2 cho truy vấn B ($RR = \frac{1}{2} = 0.5$), thì $MRR = \frac{(1 + 0.5)}{2} = 0.75$

2.2.4. Cosine similarity

Độ tương đồng Cosine là một khái niệm được sử dụng rộng rãi trong xử lý ngôn ngữ tự nhiên và khoa học dữ liệu để đo lường mức độ tương đồng giữa hai vector trong không gian nhiều chiều. Nó dựa trên góc giữa hai vector, thay vì khoảng cách Euclidean thông thường.

Cụ thể, độ tương đồng Cosine được tính bằng cách lấy tích vô hướng của hai vector chia cho tích độ dài của chúng. Độ tương đồng Cosine là giá trị cosine của góc giữa hai vector, nằm trong khoảng từ -1 đến 1 . Tuy nhiên, trong các ứng dụng thực tế với vector không âm (như biểu diễn văn bản bằng TF-IDF hoặc word embeddings), giá trị này thường chỉ dao động từ 0 đến 1 . Do đó, độ tương đồng Cosine sẽ có ý nghĩa như sau:

- 1: Hai vector hoàn toàn giống nhau (góc 0°).
- 0: Hai vector không có sự tương đồng (góc 90°).
- -1 : Hai vector ngược hướng hoàn toàn (góc 180°).

Công thức tính độ tương đồng Cosine:

Công thức tính độ tương đồng Cosine giữa hai vector A và B được định nghĩa như sau:

$$\text{Cosine Similarity} = \frac{\mathbf{A} \cdot \mathbf{B}}{|\mathbf{A}| \times |\mathbf{B}|}$$

Trong đó:

$\mathbf{A} \cdot \mathbf{B}$: Tích vô hướng của hai vector A và B, được tính bằng cách nhân từng cặp phần tử tương ứng của hai vector rồi cộng lại.

$|\mathbf{A}|$: Độ dài (norm) của vector A, tính bằng căn bậc hai của tổng bình phương các phần tử của A.

$|\mathbf{B}|$: Độ dài (norm) của vector B, tính tương tự như $|\mathbf{A}|$.

Ví dụ:

Có hai câu:

Câu 1: “Tôi thích môn máy học”

Câu 2: “Tôi yêu môn máy học”

Sử dụng thuật toán nhúng TF-IDF:

Nhúng câu 1:

[0.4090901, 0.4090901, 0.4090901, 0.57496187, 0.4090901, 0]

Nhúng câu 2:

[0.4090901, 0.4090901, 0.4090901, 0, 0.4090901, 0.57496187]

$$\begin{aligned} \mathbf{A} \cdot \mathbf{B} &: 0.4090901^2 + 0.4090901^2 + 0.4090901^2 + 0 \times 0.57496187 + \\ &0.4090901^2 + 0 \cdot 0.57496187 = 0.669451572 \end{aligned}$$

$$|\mathbf{A}| = \sqrt{3 \cdot 0.4090901^2 + 0.57496187^2 + 0.4090901^2 + 0^2} \approx 1.00001$$

$$|\mathbf{B}| = \sqrt{4 \cdot 0.167362893 + 0.57496187^2} \approx 1.00001$$

$$\text{similarity} = \frac{0.669451572}{1.00001 \times 1.00001} \approx 0.6694$$

2.2.5. Hallucination

Hallucination là hiện tượng mô hình tạo ra thông tin không chính xác, không có căn cứ hoặc mâu thuẫn với dữ liệu nguồn được truy xuất.

Các loại Hallucination

- Factual Hallucination: Tạo ra thông tin sai lệch về sự kiện, số liệu, ngày tháng.

- Contextual Hallucination: Đưa ra thông tin không liên quan đến ngữ cảnh câu hỏi.
- Source Hallucination: Tham chiếu đến nguồn không tồn tại hoặc sai.

Công thức tính toán

$$Hallucination\ Rate = \frac{\text{Số câu trả lời bị hallucination}}{\text{Tổng số câu trả lời}} \times 100\%$$

2.2.6. Accuracy

Accuracy đo lường mức độ chính xác của câu trả lời so với ground truth hoặc câu trả lời mong đợi.

Các loại Accuracy

- Exact Match Accuracy: Khớp hoàn toàn với câu trả lời chuẩn.
- Partial Accuracy: Chính xác một phần thông tin.

Công thức tính toán

$$Accuracy = \frac{\text{Số câu trả lời chính xác}}{\text{Tổng số câu hỏi}} \times 100\%$$

2.2.7. Missing

Missing đo lường mức độ thiếu sót thông tin quan trọng trong câu trả lời. Một câu trả lời có thể chính xác nhưng vẫn thiếu những thông tin cần thiết để trả lời đầy đủ câu hỏi.

Các loại Missing

- **Complete Missing:** Không có câu trả lời hoặc từ chối trả lời.
- **Partial Missing:** Thiếu một phần thông tin quan trọng.

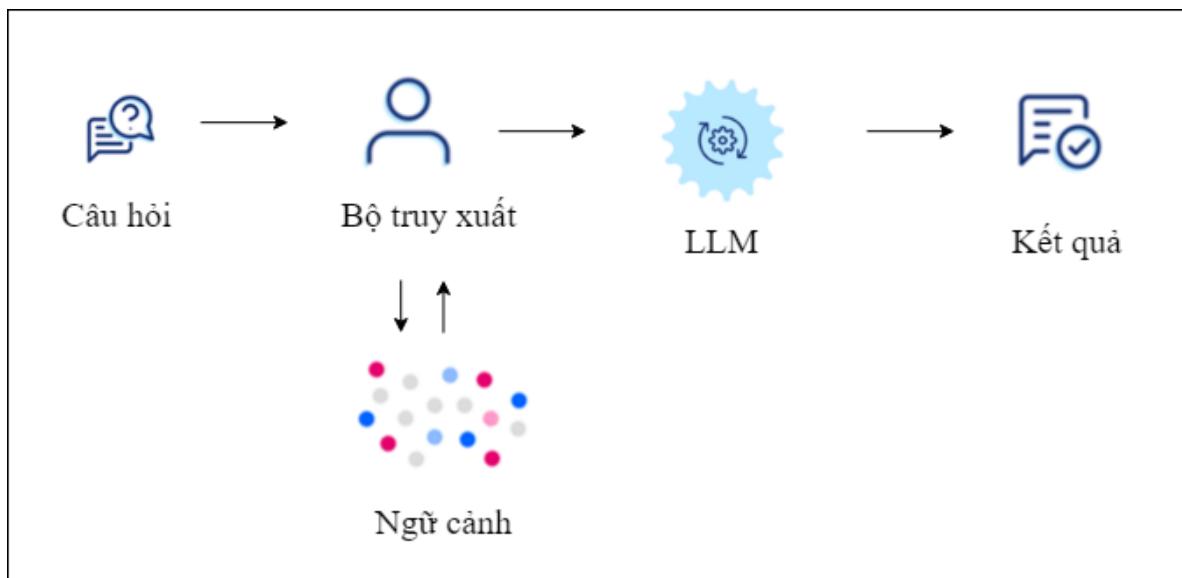
Công thức tính toán

$$Missing\ Rate = \frac{\text{Số thông tin quan trọng bị thiếu}}{\text{Tổng số thông tin cần thiết}} \times 100\%$$

2.3 RAG (Retrieval-Augmented Generation)

RAG đã nổi lên như một kỹ thuật mạnh mẽ để nâng cao khả năng của các mô hình ngôn ngữ lớn. Thay vì chỉ dựa vào kiến thức được mã hóa trong các tham số của mô hình trong quá trình huấn luyện trước, RAG tăng cường quá trình tạo sinh bằng cách truy xuất thông tin liên quan từ các nguồn dữ liệu

bên ngoài. Cách tiếp cận này đặc biệt hiệu quả trong việc giải quyết các hạn chế có hưu của LLM, chẳng hạn như xu hướng "ảo giác" (tạo ra thông tin không chính xác hoặc bịa đặt), thiếu kiến thức chuyên ngành cập nhật và khó khăn trong việc cung cấp các câu trả lời minh bạch, có thể truy xuất nguồn gốc. Bằng cách truy xuất các đoạn dữ liệu liên quan (thường là văn bản) và cung cấp chúng làm ngữ cảnh bổ sung cho LLM cùng với truy vấn của người dùng, RAG cho phép các mô hình tạo ra các câu trả lời chính xác hơn về mặt thực tế, phù hợp với ngữ cảnh hơn và đáng tin cậy hơn [10].



Hình 2.1 Kiến trúc RAG truyền thống

2.3.1. Các thành phần chính

Câu hỏi: Đầu vào do người dùng truy vấn.

Ví dụ: “ngành công nghệ thông tin thuộc khoa nào?”.

Context: Đây là cơ sở dữ liệu vector cũng là thành phần chính của thành phần RAG. Các đoạn văn bản nội bộ sẽ được chia ra thành các đoạn nhỏ, sau đó chuyển sang vector bằng cách sử dụng mô hình nhúng và được lưu trữ lại.

Ví dụ: Sử dụng model sentence-transformers/distiluse-base-multilingual-cased-v2 để nhúng:

Dòng 1:

“khoa công nghệ thông tin gồm ngành công nghệ thông tin”

Nhúng: $[-0.0486, 0.0157, 0.0119 \dots 0.0816, -0.002, 0.0055]$

Dòng 2:

“ngành công nghệ kỹ thuật cơ khí thuộc khoa cơ khí – công nghệ”

Nhúng: $[-0.0184, 0.0039, 0.0032 \dots 0.0459, 0.0312, 0.0138]$

Dòng 3:

“ngành Quản lý đất đai thuộc khoa quản lý đất đai và bất động sản

Nhúng: $[-0.0152, -0.0531, -0.0233 \dots 0.0027, -0.023, 0.0051]$

Retriever: Đây là thành phần quan trọng nhất trong mô hình RAG. Khi nhận được truy vấn từ người dùng, retriever sẽ đảm nhiệm việc truy xuất vào cơ sở dữ liệu vector, sử dụng cosine similarity để lấy ra các vector có cosine similarity so với truy vấn của người dùng.

Các bước để tính cosine similarity giữa câu hỏi và ngũ cành:

Câu hỏi(q): $[-0.0303, 0.0043, 0.0108 \dots 0.0721, -0.0106, 0.0015]$

Ngũ cành(d): $[-0.0486, 0.0157, 0.0119 \dots 0.0816, -0.002, 0.0055]$

$$\begin{aligned} \mathbf{q} \cdot \mathbf{d} &= 0.0303 \times 0.0486 + 0.0043 \times 0.0157 + 0.0108 \times 0.0119 + \dots \\ &\quad + 0.0721 \times 0.0816 + 0.0106 \times 0.002 + 0.0015 \times 0.0055 \\ &= 0.3179217 \end{aligned}$$

$$\begin{aligned} |\mathbf{q}| &= \sqrt{0.0303^2 + 0.0043^2 + 0.0108^2 + \dots + 0.0721^2 + 0.0106^2 + 0.0015^2} \\ &= 0.5468969 \end{aligned}$$

$$\begin{aligned} |\mathbf{d}| &= \sqrt{0.0486^2 + 0.0157^2 + 0.0119^2 + \dots + 0.0816^2 + 0.002^2 + 0.0055^2} \\ &= 0.64140844 \end{aligned}$$

$$|\mathbf{q}| \times |\mathbf{d}| = 0.35078428747$$

$$\cos(\mathbf{q}, \mathbf{d}) = \frac{0.3179217}{0.35078428747} = 0.9063$$

Áp dụng cho ngũ cành bên dưới ta có:

Nhúng câu hỏi: $[-0.0303, 0.0043, 0.0108 \dots 0.0721, -0.0106, 0.0015]$

Dòng 1: “ngành công nghệ thông tin thuộc khoa công nghệ thông tin” = 0.9063.

Dòng 2: “ngành công nghệ kỹ thuật cơ khí thuộc khoa cơ khí – công nghệ” = 0.5209.

Dòng 3: “ngành quản lý đất đai thuộc khoa quản lý đất đai và bất động sản” = 0.0088.

Lấy ra ngũ cành có Cosine Simalirity cao nhất và k=2: dòng 1 và dòng 2.

Khi có được các vector tương đồng, ta sẽ kết hợp với prompt và đưa đến cho LLM, có vai trò sinh ra câu trả lời cuối cùng cho người dùng.

Ví dụ: Hãy dựa vào tài liệu được cung cấp để câu hỏi:

Câu hỏi: ngành công nghệ thông tin thuộc khoa nào?.

Tài liệu:

“khoa công nghệ thông tin gồm ngành công nghệ thông tin”.

“ngành công nghệ kỹ thuật cơ khí thuộc khoa cơ khí – công nghệ”.

Phản hồi từ LLM: Ngành Công nghệ Thông Tin thuộc khoa Công nghệ Thông Tin.

Response (Output): Trả về kết quả.

2.3.2. Các phương pháp tối ưu RAG

Có 2 giai đoạn sẽ quyết định mô hình RAG có tốt hay không là: indexing(tăng tốc độ tìm kiếm) và chunking(chia nhỏ văn bản để lưu trữ trong cơ sở dữ liệu). Nếu hai giai đoạn này tối ưu không tốt sẽ dẫn tới hiệu suất không tốt và đây cũng là nhược điểm của RAG:

- **Chỉ mục:** cũng giống như các cơ sở dữ liệu có cấu trúc như MySQL, SQLServer, Oracle,... Chỉ mục dùng để tăng tốc độ tìm kiếm, tuy nhiên nó cũng làm giảm tốc độ thêm, xóa và cập nhật bởi vì mỗi lần thêm, xóa và cập nhật thì DBMS cũng sẽ phải xây dựng lại chỉ mục.

Index giúp giảm thiểu thời gian và tài nguyên cần thiết để xử lý các truy vấn phức tạp, bất kể là tìm kiếm vector hay dữ liệu dạng bảng. Trong nghiên cứu này chúng em sử dụng vector cơ sở dữ liệu Qdrant nên sẽ trình bày các loại chỉ mục mà Qdrant hỗ trợ [12].

- **Các loại chỉ mục:**

Có 1 triệu ảnh sinh viên, metadata của mỗi điểm có dạng như sau:

```
{"mssv": "21130448", "name": "cao thanh nam", "nganh": "công nghệ thông tin", "start": "2021-07-01T10:00:00Z", "end": "2025-12-01T10:00:00Z"}
```

- **Payload index:** Đóng vai trò quan trọng trong việc tối ưu hóa quá trình tìm kiếm và lọc dữ liệu [12].

Ví dụ: Khi tìm tất cả vector có name = “cao thành nam” → Qdrant sẽ dùng payload index để lọc nhanh thay vì quét toàn bộ.

- **Full-text Index:** Tìm kiếm văn bản trong payload. Dùng để tìm từ khóa hoặc cụm từ cụ thể [12].
- **Ví dụ: Khi tìm kiếm từ khóa “công nghệ” → Full-text index giúp tìm nhanh mà không cần duyệt toàn bộ văn bản từng vector.**
- **Parameterized Index:** Tối ưu tìm kiếm dữ liệu số (mã số sinh viên, điểm số). Hiệu quả với hàng triệu điểm dữ liệu [12].

Ví dụ: Khi tìm kiếm mssv > 2113000 & mssv < 21130448 → Parameterized Index tăng tốc đáng kể thay vì duyệt từng điểm.

- **On-disk Payload Index:** Lưu payload trên đĩa thay vì RAM. Tiết kiệm bộ nhớ nhưng truy cập chậm hơn [12].

Ví dụ: 10 triệu vector và mỗi vector có metadata lớn → dùng On-disk Payload Index giúp giảm bộ nhớ RAM, chấp nhận đánh đổi truy vấn chậm hơn, vì lưu ở đĩa thì truy vấn chậm hơn ở RAM

- **Tenant Index:** Hỗ trợ đa khách hàng(multitenancy). Tạo chỉ mục riêng cho từng tenant [12].

Ví dụ: Nếu chúng ta lưu dữ liệu của các công ty khác nhau vào cùng cụm của Qdrant thì khi tìm kiếm dữ liệu của công ty này sẽ tách biệt với công ty khác.

- **Principal Index :** Tối ưu khi truy vấn chủ yếu dựa trên 1 trường (như thời gian) [12].

Ví dụ: Khi thường xuyên truy vấn theo thời gian như timestamp >= 2024-01-01 → sử dụng Principal Index giúp truy vấn nhanh hơn so với index thường.

- **Vector Index:** Sử dụng thuật toán HNSW. Tìm kiếm vector tương tự hiệu quả. Cấu trúc đa lớp, lớp trên thưa hơn [12].
- **Ví dụ: Chúng ta có câu hỏi “sinh viên tên cao thành nam thuộc khoa công nghệ thông tin”, phải được nhúng thành vector: [0.0262, 0.0333, 0.0123, 0.096, 0.0821]**

Sau đó so sánh với các vector trong hệ thống.

- **Sparse Vector Index:** Tối ưu cho vector có nhiều giá trị 0. Phù hợp xử lý văn bản, từ khóa [12].

Ví dụ: Sử dụng thuật toán thích hợp để nhúng câu hỏi, nhưng nhúng theo từ khóa:

[0, 0, 0.3, 0, 0.5, 0, 0, 0.1, 0, 0, 0]

Sau đó so sánh với các vector trong hệ thống.

- **Filtrable Index:** Tìm kiếm vector có điều kiện lọc. Duy trì liên kết trong đồ thị HNSW khi có filter [12].

Ví dụ: Giống với vector index nhưng có thêm điều kiện lọc

Chúng ta có câu hỏi “sinh viên tên cao thành nam ” và {“mssv”: “21130448”} → Kết hợp filter + similarity.

Trong kiến trúc này chúng em sử dụng **Vector Index** vì chủ yếu truy vấn trên vector thưa thớt và sử dụng nhiều lớp vector khác nhau nhưng không tìm kiếm theo từ khóa.

- **Các bước chung để chọn chiến lược thực thi:**

- **Lập kế hoạch độc lập cho từng phân đoạn:** Qdrant tổ chức dữ liệu thành các phân đoạn (segments) riêng biệt, mỗi phân đoạn được xử lý độc lập trong quá trình thực hiện truy vấn. Cách tiếp cận này cho phép hệ thống áp dụng các chiến lược truy vấn tối ưu phù hợp với đặc điểm của từng phần dữ liệu, thay vì phải xử lý toàn bộ tập dữ liệu theo một cách thức duy nhất. Điều này giúp tăng hiệu suất tổng thể và tận dụng tối đa tài nguyên hệ thống [13].

- **Ưu tiên quét toàn bộ cho tập dữ liệu nhỏ:** Trong Qdrant – cơ sở dữ liệu vector dùng cho tìm kiếm và gợi ý – dữ liệu được chia thành các segment (đơn vị chứa một phần dữ liệu, tương tự như “mảnh” của toàn bộ tập dữ liệu). Khi một segment có số lượng điểm (points, tức các vector được lưu trữ) nhỏ hơn một ngưỡng nhất định (threshold), hệ thống sẽ ưu tiên sử dụng phương pháp quét toàn bộ (full scan).

Phương pháp này nghĩa là duyệt qua tất cả các điểm trong segment để tìm kết quả phù hợp, thay vì phải dùng đến các chỉ mục tìm kiếm phức tạp (complex indexes) như HNSW (Hierarchical Navigable

Small World Graph). Với khối lượng dữ liệu nhỏ, chi phí tính toán khi quét toàn bộ là rất thấp, và trong nhiều trường hợp còn nhanh hơn so với việc truy vấn qua chỉ mục [13].

- **Ước lượng số lượng kết quả trước khi chọn chiến lược:** Trước khi quyết định phương pháp thực hiện truy vấn, Qdrant thực hiện việc ước lượng số lượng điểm dữ liệu mà bộ lọc sẽ trả về. Thông tin này đóng vai trò quan trọng trong việc lựa chọn chiến lược tối ưu: nếu số lượng kết quả dự kiến thấp, hệ thống sẽ chọn phương pháp truy xuất đơn giản; ngược lại, nếu số lượng cao, sẽ cần sử dụng các chỉ mục chuyên biệt để đảm bảo hiệu suất [13].
- **Sử dụng chỉ mục payload cho kết quả lọc ít:** Khi quá trình ước lượng (estimation) cho thấy số lượng điểm dữ liệu (points – các vector được lưu trữ trong cơ sở dữ liệu) thỏa mãn điều kiện lọc (filter conditions) nằm dưới một ngưỡng quy định (threshold), Qdrant sẽ tận dụng chỉ mục payload (payload index) để truy xuất dữ liệu. Payload là dữ liệu phi-vector (ví dụ: nhãn, danh mục, thuộc tính mô tả) gắn liền với mỗi điểm. Chỉ mục payload được tối ưu hóa nhằm hỗ trợ việc truy cập nhanh chóng và hiệu quả đến các điểm dữ liệu trong trường hợp khối lượng cần xử lý không quá lớn. Cách tiếp cận này giúp giảm thiểu thời gian phản hồi của truy vấn, đồng thời nâng cao hiệu năng hệ thống [13].
- **Chuyển sang Vector Index có thể lọc cho kết quả lớn:** Được sử dụng khi ước lượng cho thấy số lượng kết quả lọc vượt ngưỡng. Loại chỉ mục này được thiết kế đặc biệt để xử lý hiệu quả các truy vấn với khối lượng dữ liệu lớn, đảm bảo quá trình tìm kiếm và truy xuất các điểm dữ liệu phù hợp được thực hiện một cách tối ưu [13].
- **Khả năng tùy chỉnh ngưỡng linh hoạt:** Qdrant cung cấp khả năng điều chỉnh các ngưỡng quyết định chiến lược thông qua file cấu hình hệ thống. Người dùng có thể thiết lập ngưỡng chung áp dụng cho toàn bộ hệ thống hoặc tùy chỉnh riêng biệt cho từng bộ sưu tập. Tính năng này cho phép tối ưu hóa hiệu suất dựa trên đặc thù và yêu cầu cụ thể

của từng tập dữ liệu, mang lại sự linh hoạt cao trong việc quản lý và vận hành hệ thống [13].

Ví dụ: Hệ thống hiện tại lưu trữ 1 triệu vector ảnh sinh viên kèm theo metadata có dạng như sau:

```
{"mssv": "21130448", "name": "cao thành nam", "ngành": "công nghệ thông tin", "start": "2021-07-01T10:00:00Z", "end": "2025-12-01T10:00:00Z"}
```

Cấu hình ban đầu gồm:

- Nguồn quét toàn bộ trong segment < 5000 .
- Nguồn sử dụng payload index khi ước lượng < 10000 vector.
- Nguồn sử dụng filtrable index khi ước lượng > 20000 .

Trong trường hợp số lượng điểm dữ liệu thỏa mãn điều kiện lọc không nằm trong nguồn nhỏ (ưu tiên quét toàn bộ) cũng như không thuộc nguồn ít (sử dụng chỉ mục payload) – tức là rơi vào vùng trung gian – thì Qdrant sẽ áp dụng phương pháp filter scan. Cách tiếp cận này nghĩa là hệ thống sẽ quét toàn bộ segment (tập dữ liệu con được lưu trữ độc lập) nhưng có kết hợp với điều kiện lọc (filter conditions) để loại bỏ các điểm không phù hợp. Phương pháp này phù hợp với khối lượng dữ liệu ở mức trung bình, nơi việc sử dụng chỉ mục phức tạp chưa chắc hiệu quả hơn so với việc duyệt qua dữ liệu trực tiếp.

Câu truy vấn gồm: {“name”: “cao thành nam”, “mssv”: 21130448} và câu hỏi “ngành công nghệ thông tin”

Bước 1: Chia tập dữ liệu thành nhiều segments (các đơn vị lưu trữ dữ liệu độc lập trong Qdrant, mỗi segment chứa một tập con các điểm dữ liệu – points, tức các vector, và tự quyết định cách truy vấn).

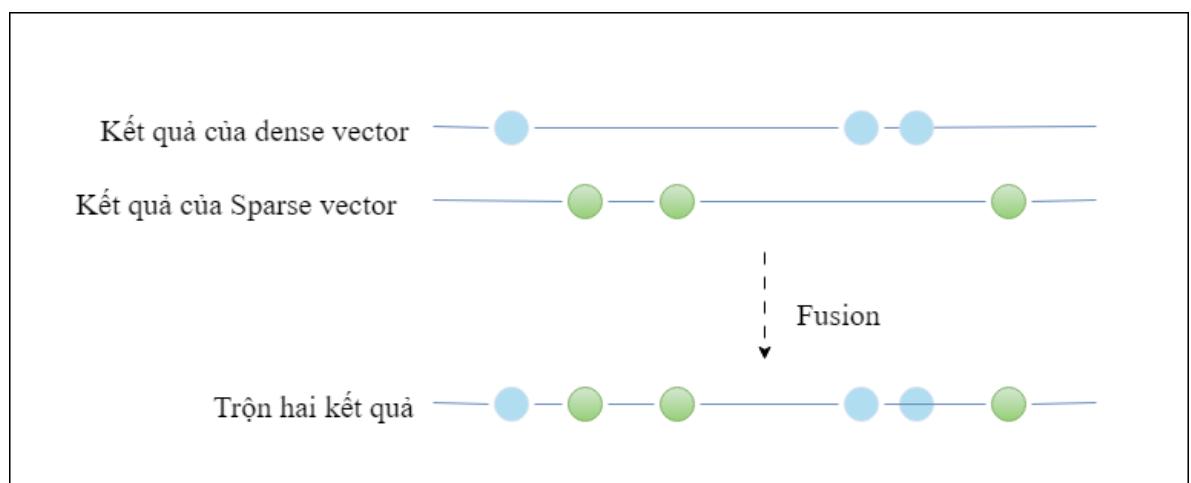
Bước 2: Nếu một segment có số lượng vector từ 3.000 đến 5.000 (vector – các điểm dữ liệu lưu dưới dạng mảng số, points trong Qdrant), hệ thống sẽ sử dụng full scan (quét toàn bộ các điểm trong segment để tìm kết quả phù hợp).

Bước 3: Trước khi lọc, Qdrant sẽ ước lượng có bao nhiêu vector (vector – các điểm dữ liệu dưới dạng mảng số, points trong Qdrant) khớp với

câu truy vấn. Ví dụ, nếu 5.000 vector khớp (số vector thỏa mãn điều kiện) < 10.000 vector, hệ thống sẽ sử dụng **payload index** (chỉ mục được xây dựng dựa trên dữ liệu phi-vector gắn với mỗi point, giúp truy xuất nhanh các điểm phù hợp khi khôi lượng cần xử lý không quá lớn).

Bước 4: Nếu một câu truy vấn có ước lượng 25.000 vector khớp (vector – các điểm dữ liệu dưới dạng mảng số, points trong Qdrant) > 20.000 vector, hệ thống sẽ sử dụng **filtrable index** (chỉ mục được tối ưu hóa cho các truy vấn lọc trên số lượng lớn điểm dữ liệu, kết hợp giữa chỉ mục payload và cơ chế quét để tăng hiệu suất khi khôi lượng dữ liệu lớn).

- **Tìm kiếm kết hợp:**



Hình 2.2 Kết hợp kết quả từ nhiều truy vấn

Một trong những vấn đề chung nhất là khi dữ liệu giống nhau nhưng được biểu diễn ở nhiều trạng thái khác nhau, cần kết hợp các điểm truy vấn (points – các vector dữ liệu) thành một kết quả duy nhất. Dưới đây là ví dụ về tìm kiếm chữ (text search), thường được biểu diễn ở dense vector (vector đặc, chứa đầy đủ các giá trị số thực) và sparse vector (vector thưa, hầu hết giá trị bằng 0, chỉ lưu trữ các giá trị khác 0 để tiết kiệm bộ nhớ). Qdrant hiện có hai thuật toán chính để kết hợp các dạng vector này. [14].

- **Thuật toán RRF(Reciprocal Rank Fusion):** Xem xét vị trí của các kết quả trong mỗi truy vấn và tăng trọng số cho những kết quả xuất hiện gần đầu trong nhiều truy vấn. Điều này đảm bảo rằng các kết

quả quan trọng, được xếp hạng cao trong nhiều truy vấn sẽ được ưu tiên [14].

- Công thức:

$$RRF\ score = \sum_{i=1}^n \frac{1}{k + rank_i}$$

k thường được đặt là 60 trong thực tế để giảm ưu tiên quá cao cho hạng đầu.

Ví dụ: Ở đây chọn $k = 0$

Kết quả của dense vector: A (hạng 1), B (hạng 2), C (hạng 3).

Kết quả của sparse vector: B (hạng 1), C (hạng 2), D (hạng 3).

Kết quả	Dense Rank	Sparse Rank	Điểm RRF
A	1	\emptyset	$\frac{1}{1} = 1$
B	2	1	$\frac{1}{2} + \frac{1}{1} = 1.5$
C	3	2	$\frac{1}{3} + \frac{1}{2} = 0.83$
D	\emptyset	3	$\frac{1}{3} = 0.33$

Kết quả xếp hạng sau fusion:

$B > A > C > D$.

- **Thuật toán DBSF(Distribution-Based Score Fusion):** Chuẩn hóa điểm số của các điểm trong mỗi truy vấn, sử dụng giá trị trung bình cộng và trừ ba độ lệch chuẩn làm giới hạn, sau đó cộng dồn điểm số của cùng một điểm qua các truy vấn khác nhau. Phương pháp này không lưu trạng thái và chỉ tính toán giới hạn chuẩn hóa dựa trên kết quả của mỗi truy vấn, không dựa trên tất cả các điểm số đã thấy [14].

Ví dụ:

Kết quả của dense vector:

A: 0.8, B: 0.6, C: 0.4, D: 0.2.

Kết quả của sparse vector:

A: 0.1, B: 0.3, C: 0.9, D: 0.6.

Bước 1: Sử dụng z-score để chuẩn hóa do sự chênh lệch điểm số giữa dense vector (độ tương đồng cosine từ 0 đến 1) và sparse vector (có thể từ 0 đến vài chục):

Công thức z-score: $z = \frac{x - \mu}{\sigma}$

Trong đó:

x là giá trị quan sát

μ là trung bình của tập quan sát

σ là độ lệch chuẩn của tập quan sát

Dense vector:

Trung bình: $\frac{0.8 + 0.6 + 0.4 + 0.2}{4} = 0.5$

Độ lệch chuẩn: $\sqrt{\frac{0.3^2 + 0.1^2 + 0.1^2 + 0.3^2}{4}} = \sqrt{\frac{0.2}{4}} = 0.2236$

Tài liệu	z-score
A	$\frac{0.8 - 0.5}{0.2236} \approx 1.34$
B	$\frac{0.6 - 0.5}{0.2236} \approx 0.45$
C	$\frac{0.4 - 0.5}{0.2236} \approx -0.45$
D	$\frac{0.2 - 0.5}{0.2236} \approx -1.34$

Sparse vector:

Trung bình: $\frac{0.1 + 0.3 + 0.9 + 0.6}{4} = 0.475$

Độ lệch chuẩn: tính tương tự ≈ 0.294

Tài liệu	z-score
A	$\frac{0.1 - 0.475}{0.294} \approx -1.27$
B	$\frac{0.3 - 0.475}{0.294} \approx -0.59$
C	$\frac{0.9 - 0.475}{0.294} \approx 1.44$

D	$\frac{0.6 - 0.475}{0.294} \approx 0.43$
---	--

Bước 2: Tính DBFS

Tài liệu	Dense z	Sparse z	DBFS
A	1.34	-1.27	0.07
B	0.45	-0.59	-0.14
C	-0.45	1.44	0.99
D	-1.34	0.43	-0.91

Kết quả xếp hạng sau fusion:

$C > A > B > D$

- **Tìm kiếm nhiều giai đoạn:** Cho phép xây dựng quy trình tìm kiếm phức tạp bằng cách kết hợp nhiều bước truy vấn, giúp tối ưu hóa hiệu suất và độ chính xác của quá trình tìm kiếm. Điều này đặc biệt hữu ích khi làm việc với các embedding lớn hoặc khi cần thực hiện các bước xử lý bổ sung giữa các giai đoạn tìm kiếm. Ví dụ: nếu cần tìm một dense vector có độ dài là 1024 thì có thể chia thành 3 giai đoạn:
 - Giai đoạn 1: Tìm kiếm bằng dense vector có độ dài là 512.
 - Giai đoạn 2: Sử dụng kết quả của giai đoạn 1, tiếp tục tìm kiếm bằng dense vector có độ dài là 768.
 - Giai đoạn 3: Sử dụng kết quả của giai đoạn 2, tiếp tục tìm kiếm bằng dense vector có độ dài là 1024.
- Trong kiến trúc này, sau nhiều lần thử nghiệm và trải nghiệm chúng em sử dụng tìm kiếm nhiều giai đoạn được tổ chức như sau:
 - **Giai đoạn 1:** Tên vector là “matryoshka-512dim”. Dense vector có độ dài là 512. Sử dụng Cosine để so sánh.
 - **Giai đoạn 2:** Tên vector là “matryoshka-768dim”. Dense vector có độ dài là 768. Sử dụng Cosine để so sánh.
 - **Giai đoạn 3:** Tên vector là “matryoshka-1024dim”. Dense vector có độ dài là 1024. Sử dụng Cosine để so sánh.

- **Giai đoạn 4:** Đây là bước reranking. Tên vector là “late interaction”. Độ dài của tập các vector không cố định, phụ thuộc vào độ dài của văn bản đầu vào do sử dụng ColBERT không nhúng toàn bộ câu mà nhúng từng token trong câu. Trong mỗi vector có kích thước là 128. Sử dụng cosine similarity để so sánh.

Ví dụ:

Cấu trúc của mỗi dữ liệu có dạng như sau:

```
{  
  "id": 1,  
  "payload": {  
    "text": "Trường Đại học Nông Lâm Thành phố Hồ Chí Minh (Nong  
Lam University - NLU) là một trường đa ngành, trực thuộc Bộ Giáo dục  
và Đào tạo, tọa lạc trên khu đất rộng 118 ha, thuộc Thành phố Thủ Đức,  
Thành phố Hồ Chí Minh và Thành phố Dĩ An - Tỉnh Bình Dương"  
  },  
  "vector": {  
    "matryoshka-1024dim": [...],      // kích thước = 1024  
    "matryoshka-768dim": [...],        // kích thước= 768  
    "matryoshka-512dim": [...],        // kích thước= 512  
    "late_interaction": [[...], [...], ...] // kích thước cho mỗi phần tử = 128  
  }  
}
```

Dưới đây là ví dụ cho tập dữ liệu gồm 20 vector:

ID	Text	Nhúng
1	Trường Đại học Nông Lâm Thành phố Hồ Chí Minh (Nong Lam University - NLU) là một trường đa ngành, trực thuộc Bộ Giáo dục và Đào	[0.01777222, -0.013084799, ...] [-0.019785576, 0.0038806216, -0.05994848 ...] [0.0054817693, -0.003421314, -0.00435195, -0.046457347 ...] [[0.015553812, -0.05922641 ...],

	tạo, tọa lạc trên khu đất rộng 118 ha, thuộc Thành phố Thủ Đức, Thành phố Hồ Chí Minh và Thành phố Dĩ An - Tỉnh Bình Dương.	[0.075613566, -0.040704682 ...]
2	Trải qua gần 70 năm xây dựng và phát triển, Trường đã đạt nhiều thành tích xuất sắc về đào tạo, nghiên cứu và ứng dụng khoa học kỹ thuật nông lâm ngư nghiệp, chuyển giao công nghệ, cũng như trong quan hệ quốc tế. Những đóng góp to lớn đó đã được ghi nhận qua các danh hiệu cao quý như Huân chương Lao động Hạng ba, Huân chương Lao động Hạng nhất, và Huân chương Độc lập Hạng ba.	[0.03675997, -0.03373935 ...] [-0.000003270404, -0.0033381246, -0.09713896 ...] [0.011382493, 0.013544297, -0.009362282, -0.049041405 ...] [[0.027184485, -0.0015634621 ...], [0.051737923, -0.019860707 ...]]
3	Sứ mệnh Trường Đại học Nông Lâm TP.HCM là một trường đại học đa ngành, đào tạo nguồn nhân lực có chuyên môn tốt và tư duy sáng tạo; thực hiện nhiệm vụ nghiên cứu, phát triển, phổ biến, chuyển	[0.031916216, 0.002697076, ...] [0.010824892, 0.0051929993, -0.012501313, ...] [-0.011135993, 0.022403724, -0.0072000464, -0.050051786 ...] [[0.0028684842, 0.0005985167 ...], [0.047634445, -0.0009187137 ...]]

	<p>giao tri thức - công nghệ, đáp ứng nhu cầu phát triển bền vững kinh tế - xã hội của Việt Nam và khu vực. Tầm nhìn Đến năm 2035, Trường Đại học Nông Lâm TP.HCM sẽ trở thành trường đại học nghiên cứu với chất lượng quốc tế.</p>	
4	<p>Giá trị cốt lõi - Nhân văn: Gìn giữ và phát huy các giá trị truyền thống nhân văn của dân tộc. - Nhân bản: Phát hiện, nâng đỡ tài năng và tính sáng tạo, chú trọng đào tạo kỹ năng và trách nhiệm nghề nghiệp cho người học.- Phục vụ: Tôn trọng lợi ích của người học và của cộng đồng. Xây dựng xã hội học tập. - Đổi mới: Đè cao chất lượng, hiệu quả và sự đổi mới trong các hoạt động của nhà trường. - Hội nhập: Hội nhập, hợp tác và chia sẻ.</p>	<p>$[-0.024286583, -0.034784004 \dots]$ $[0.025090175, -0.0265085, -0.023886617 \dots]$ $[0.021988858, 0.036558088, 0.0019345758, -0.024387557 \dots]$ $[[-0.035808846, 0.02374423 \dots], [0.019448284, 0.0056103044 \dots]]$</p>
5	<p>Phòng Công tác Sinh viên Điện thoại 028.38974560</p>	<p>$[-0.018496769, 0.05479265 \dots]$ $[-0.07630254, 0.046671845,$</p>

	Website http://nls.hcmuaf.edu.vn	0.021120025 ...] [0.0039662, 0.017439859, 0.009401997, -0.05031533 ...] [[0.040451255, -0.03158991 ...], [0.047951054, -0.009653229 ...]]
6	Phòng Đào tạo Điện thoại 028.38963350 Website http://pdt.hcmuaf.edu.vn	[-0.01419149, 0.045895364 ...] [-0.028536217, 0.0022096795, -0.011183853 ...] [0.013677991, 0.023950921, -0.009051133, -0.048517518 ...] [[0.042257577, -0.03914049 ...], [0.026349768, -0.010166951 ...]]
7	Phòng Đào tạo Sau đại học Điện thoại 028.38963339 Website http://pgo.hcmuaf.edu.vn	[0.006311831, 0.048534162 ...] [-0.04262717, 0.0061493693, -0.0140191205 ...] [0.009731341, 0.01822536, -0.0068231267, -0.049164254 ...] [[0.051415324, -0.07175055 ...], [0.052893948, -0.034299143 ...]]
8	Phòng Hành chính Điện thoại 028.38966780 Website https://ado.hcmuaf.edu.vn	[-0.013826931, 0.05116595 ...] [-0.076194376, 0.02974034, 0.0039018095 ...] [-0.0014351618, 0.021110175, 0.008636248, -0.059340235 ...] [[0.061008547, -0.04467458 ...], [0.034370508, -0.020055626 ...]]
9	Phòng Hợp tác Quốc tế Điện thoại 028.38966946	[-0.029082492, 0.03654399 ...] [-0.066106215, 0.018074885, 0.005093752 ...]

	Website http://iro.hcmuaf.edu.vn	[0.019867292, 0.016515927, 0.011468203, -0.048078734 ...] [[0.060379036, -0.022659628 ...], [0.04995528, -0.021238338 ...]]
10	Phòng Kế hoạch Tài chính Điện thoại 028.38963334 Website http://pkhtc.hcmuaf.edu.vn	[-0.029170105, 0.041191146 ...] [-0.081675164, 0.034092683, 0.0044204323 ...] [0.011499726, 0.028409377, 0.0037228006, -0.051746216 ...] [[0.020770984, -0.032702025 ...], [0.031627286, -0.010392281 ...]]
n+1

Câu hỏi: “đội khát vọng tuổi trẻ của khoa chăn nuôi thú y có slogan là gì và đơn vị nào quản lý đội này?”

Nhúng câu hỏi:

- Kích thước 1024:
[-0.002553306, 0.0043374747, -0.038316738, -0.057994306 ...]
- Kích thước 768:
[-0.046592344, 0.08493743, -0.08814325 ...]
- Kích thước 512:
[0.014260627, 0.002156657 ...]
- Late interaction:
[[-0.0761604905128479, 0.03039712831377983 ...],
[-0.05458960309624672, -0.016857564449310303 ...] ...]

Giai đoạn 1: Sử dụng vector kích thước 512 để lấy top 200 có độ tương đồng cao nhất trong tập dữ liệu.

Kết quả:

['7. FIRE ENGLISH CLUB\nSlogan: We are FIRE – FIRE KHÔNG PHAI MỒ\n\nLĩnh vực hoạt động: Ngoại ngữ.\nĐơn vị quản lý: Đoàn–Hội khoa

Ngoại ngữ-Sư phạm.\nChủ nhiệm CLB: Nguyễn Hoàng Nam Phương.\nEmail: fireenglishclub@tuoitrenonglam.com\nFanpage: <https://www.facebook.com/FiRE.EnglishClub>', '15. CLB TIẾNG ANH KHOA KINH TẾ EFB (English For Business Club) EFB\nSlogan: Will be the better not be the best\nLĩnh vực hoạt động: Học thuật, ngoại ngữ.\nĐơn vị quản lý: Đoàn - Hội khoa Kinh tế.\nChủ nhiệm CLB: Nguyễn Hoàng Tuấn.\nEmail: englishforbusinessnlu@gmail.com\nFanpage: <https://www.facebook.com/englishforbusinessnlu>'...]

Giai đoạn 2: Sử dụng vector kích thước 768 để lấy top 100 có độ tương đồng cao nhất ở giai đoạn 1.

Kết quả:

['23. ĐỘI KHÁT VỌNG TUỔI TRẺ KHOA CHĂN NUÔI THÚ Y\nSlogan: Gắn kết sức trẻ - chia sẻ yêu thương\nLĩnh vực hoạt động: Hỗ trợ hoạt động Đoàn - Hội khoa.\nĐơn vị quản lý: Đoàn - Hội khoa Chăn nuôi Thú y.\nĐội trưởng: Trần Viết Nguyên Chương.\nEmail: doikhatvongtuoitre@tuoitrenonglam.com\nFanpage: <https://www.facebook.com/khatvongtuoitrekhaoacnty>', '26. ĐỘI VĂN NGHỆ RẠNG ĐÔNG\nSlogan: Rạng Đông khoảng cách bằng 0\nLĩnh vực hoạt động: Văn nghệ, Truyền thông.\nĐơn vị quản lý: Đoàn Thanh niên.\nĐội trưởng: Lê Thành Tài.\nEmail: vanngherangdong@hcmuaf.edu.vn\nFanpage: <https://www.facebook.com/doivannghe.rangdong>'...]

Giai đoạn 3: Sử dụng vector kích thước 1024 để lấy top 50 có độ tương đồng cao nhất ở giai đoạn 2.

Kết quả:

['23. ĐỘI KHÁT VỌNG TUỔI TRẺ KHOA CHĂN NUÔI THÚ Y\nSlogan: Gắn kết sức trẻ - chia sẻ yêu thương\nLĩnh vực hoạt động: Hỗ trợ hoạt động Đoàn - Hội khoa.\nĐơn vị quản lý: Đoàn - Hội khoa Chăn nuôi Thú y.\nĐội trưởng: Trần Viết Nguyên Chương.\nEmail: doikhatvongtuoitre@tuoitrenonglam.com\nFanpage: <https://www.facebook.com/khatvongtuoitrekhaoacnty>', '5. CLB DUỢC THÚ

Y\nSlogan: Học tập – Chia sẻ - Nghiên cứu – Sáng tạo\nLĩnh vực hoạt động: Học thuật, Nghiên cứu khoa học.\nĐơn vị quản lý: Đoàn – Hội khoa Chăn nuôi Thú y.\nChủ nhiệm CLB: Nguyễn Nữ Mai Thơ.\nEmail: duocthuynlu@gmail.com\nFanpage: Câu lạc bộ Dược Thú Y Đại học Nông Lâm TPHCM'...]

Giai đoạn 4: Sử dụng vector late interaction để reranking lấy top 2 với các vector ở giai đoạn 3.

Kết quả:

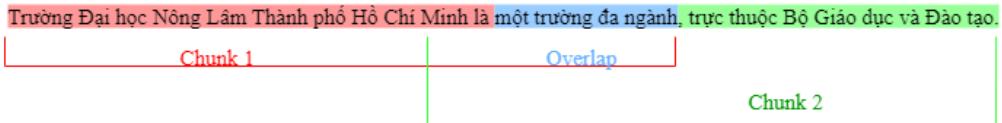
['23. ĐỘI KHÁT VỌNG TUỔI TRẺ KHOA CHĂN NUÔI THÚ Y\nSlogan: Gắn kết sức trẻ - chia sẻ yêu thương\nLĩnh vực hoạt động: Hỗ trợ hoạt động Đoàn - Hội khoa.\nĐơn vị quản lý: Đoàn - Hội khoa Chăn nuôi Thú y.\nĐội trưởng: Trần Viết Nguyên Chương.\nEmail: doikhatvongtuotitre@tuoitrenonglam.com\nFanpage <https://www.facebook.com/khatvongtuotrekhoacnty>', '5. CLB DUỢC THÚ Y\nSlogan: Học tập – Chia sẻ - Nghiên cứu – Sáng tạo\nLĩnh vực hoạt động: Học thuật, Nghiên cứu khoa học.\nĐơn vị quản lý: Đoàn – Hội khoa Chăn nuôi Thú y.\nChủ nhiệm CLB: Nguyễn Nữ Mai Thơ.\nEmail: duocthuynlu@gmail.com\nFanpage: Câu lạc bộ Dược Thú Y Đại học Nông Lâm TPHCM'...]

- **Chunking:**

Quá trình chia nhỏ các tài liệu lớn thành các đoạn nhỏ hơn, gọi là "chunk". Mỗi chunk có thể là một đoạn văn, câu hoặc một phần văn bản có ý nghĩa riêng biệt. Việc này giúp mô hình dễ dàng truy xuất và xử lý thông tin, cải thiện hiệu suất và độ chính xác của hệ thống. Kỹ thuật này giúp tối ưu hóa hiệu năng của RAG [15].

Chọn kỹ thuật để chunking là một quyết định khó khăn tùy theo cấu trúc và nội dung của tài liệu, tuy nhiên chúng em sẽ nêu ra một vài kỹ thuật khác nhau, trong đó chunking dựa trên LLM là kỹ thuật chúng em ưu tiên:

- **Chunking theo kích thước cố định:** Chia tài liệu thành những phần nhỏ theo kích thước cố định đã được cấu hình sẵn. Chúng ta cần chọn 2 tham số [16]:



Hình 2.3 Quá trình chunking theo kích thước cố định

- **Chunk size:** Số lượng ký tự hoặc token trong mỗi chunk. Việc lựa chọn kích thước phù hợp phụ thuộc vào giới hạn đầu vào của mô hình và đặc điểm của dữ liệu.
- **Chunk overlap:** Số lượng ký tự hoặc token được chia sẻ giữa các chunk liên tiếp. Độ chồng chéo giúp duy trì cảnh giác giữa các chunk và thường được thiết lập từ 10% đến 20% kích thước chunk.

Ví dụ: Có đoạn văn sau

“Trường có 6 giảng đường, 10 trung tâm, 01 viện nghiên cứu và ứng dụng, 01 thư viện trung tâm với trên 15.000 đầu sách, 01 bệnh viện thú y, 01 xưởng dược thú y, 01 trại thực nghiệm thủy sản và 04 trung tâm nghiên cứu thí nghiệm về nông học, lâm nghiệp, nuôi trồng thủy sản, chăn nuôi... Trường đã sử dụng thư viện điện tử góp phần nâng cao năng lực nghiên cứu và tự học của sinh viên. Trường có 6 ký túc xá nhiều năm liền đạt danh hiệu ký túc xá sinh viên văn hóa cấp thành phố, gồm 350 phòng, sức chứa 3.000 sinh viên với 1 sân đa môn, 3 sân bóng chuyên và 1 sân bóng đá cùng với Nhà thi đấu và luyện tập thể thao hiện đại có sức chứa 1.000, tạo sân chơi bổ ích, rèn luyện “tinh thần minh mẫn trong thân thể tráng kiện” cho sinh viên trong quá trình học tập tại trường.”

Các chunk được tạo với chunk size= 200 và chunk overlap= 20:

Chunk 1: “Trường có 6 giảng đường, 10 trung tâm, 01 viện nghiên cứu và ứng dụng, 01 thư viện trung tâm với trên 15.000 đầu sách, 01

bệnh viện thú y, 01 xưởng dược thú y, 01 trại thực nghiệm thủy sản và 04”

Chunk 2: “thủy sản và 04 trung tâm nghiên cứu thí nghiệm về nông học, lâm nghiệp, nuôi trồng thủy sản, chăn nuôi... Trường đã sử dụng thư viện điện tử góp phần nâng cao năng lực nghiên cứu và tự học của sinh”

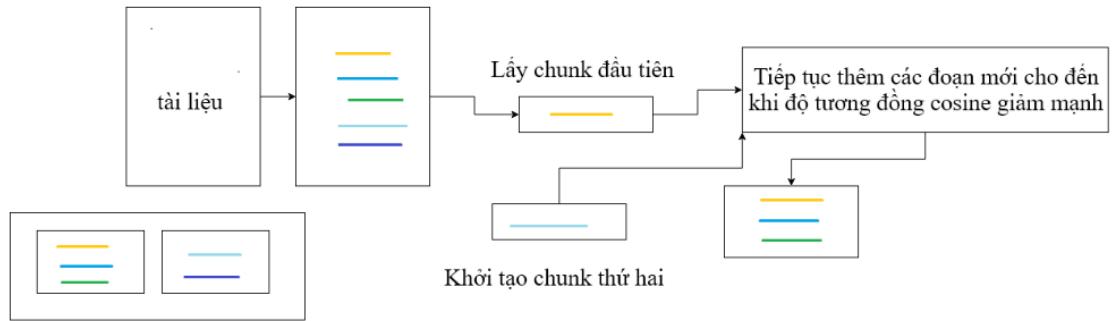
Chunk 3: “và tự học của sinh viên. Trường có 6 ký túc xá nhiều năm liền đạt danh hiệu ký túc xá sinh viên văn hóa cấp thành phố, gồm 350 phòng, sức chứa 3.000 sinh viên với 1 sân đa môn, 3 sân bóng chuyền và 1”

Chunk 4: “bóng chuyền và 1 sân bóng đá cùng với Nhà thi đấu và luyện tập thể thao hiện đại có sức chứa 1.000, tạo sân chơi bổ ích, rèn luyện “tinh thần minh mẫn trong thân thể tráng kiện” cho sinh viên trong”

Chunk 5: “cho sinh viên trong quá trình học tập tại trường.”

- **Chunking theo ngữ nghĩa:** Đây là chiến thuật mà chúng em sử dụng để chunking và cũng được khuyến nghị nhất vì có xác suất cao giữ được ý nghĩa trong một câu, đảm bảo ý nghĩa toàn vẹn của một đoạn văn bản dài mà không bị ngắt đoạn giữa chừng. Đây là các bước mà chúng em thực hiện [16]:

- Bước 1: Phân đoạn văn bản thành các đơn vị cơ bản như câu.
- Bước 2: Tạo vector embedding cho từng phân đoạn này.
- Bước 3: Lặp qua từng phân đoạn. Bắt đầu với phân đoạn đầu tiên. Sử dụng độ tương đồng similarity để so sánh giữa embedding của nó với phân đoạn tiếp theo. Nếu độ tương đồng vượt ngưỡng do chúng em đặt ra thì gộp lại với nhau. Còn không thì phân đoạn đó kết thúc và tiếp tục một chunk mới với phân đoạn tiếp theo.



Hình 2.4 Quá trình chuking theo ngữ nghĩa

Ví dụ: Các chunk được tạo, sử dụng mô hình ‘all-MiniLM-L6-v2’ để nhúng:

Chunk 1: “Trường có 6 giảng đường, 10 trung tâm, 01 viện nghiên cứu và ứng dụng, 01 thư viện trung tâm với trên 15”

Chunk 2: “000 đầu sách, 01 bệnh viện thú y, 01 xưởng dược thú y, 01 trại thực nghiệm thủy sản và 04 trung tâm nghiên cứu thí nghiệm về nông học, lâm nghiệp, nuôi trồng thủy sản, chăn nuôi”

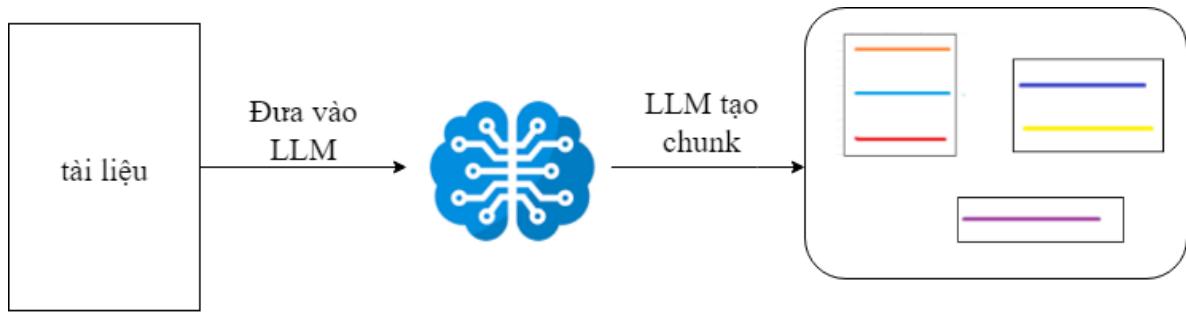
Chunk 3: “Trường đã sử dụng thư viện điện tử góp phần nâng cao năng lực nghiên cứu và tự học của sinh viên”

Chunk 4: “Trường có 6 ký túc xá nhiều năm liền đạt danh hiệu ký túc xá sinh viên văn hóa cấp thành phố, gồm 350 phòng, sức chứa 3”

Chunk 5: “000 sinh viên với 1 sân đa môn, 3 sân bóng chuyền và 1 sân bóng đá cùng với Nhà thi đấu và luyện tập thể thao hiện đại có sức chứa 1”

Chunk 6: “000, tạo sân chơi bổ ích, rèn luyện “tinh thần minh mẫn trong thân thể tráng kiện” cho sinh viên trong quá trình học tập tại trường”

- **Chunking dựa trên LLM:** Vì sao chúng ta không sử dụng LLMs để chia chunk? Với khả năng hiểu ngôn ngữ tự nhiên và các ý nghĩa trong câu thì việc chunking với LLMs được xem là đơn giản nhất [16].



Hình 2.5 Quá trình chunking dựa trên LLM

Ví dụ: Các chunk được tạo sử dụng GPT-5:

Chunk 1: “Trường có 6 giảng đường, 10 trung tâm, 01 viện nghiên cứu và ứng dụng, 01 thư viện trung tâm với trên 15.000 đầu sách, 01 bệnh viện thú y, 01 xưởng dược thú y, 01 trại thực nghiệm thủy sản.”

Chunk 2: “Trường còn có 04 trung tâm nghiên cứu thí nghiệm về nông học, lâm nghiệp, nuôi trồng thủy sản, chăn nuôi. Thư viện điện tử được sử dụng nhằm nâng cao năng lực nghiên cứu và tự học của sinh viên.”

Chunk 3: “Trường có 6 ký túc xá nhiều năm liền đạt danh hiệu ký túc xá sinh viên văn hóa cấp thành phố, gồm 350 phòng, sức chứa 3.000 sinh viên, cùng 1 sân đa môn, 3 sân bóng chuyền và 1 sân bóng đá.”

Chunk 4: “Nhà thi đấu và luyện tập thể thao hiện đại có sức chứa 1.000 chỗ, tạo sân chơi bổ ích, rèn luyện “tinh thần minh mẫn trong thân thể tráng kiện” cho sinh viên trong quá trình học tập.”

Tiêu chí	Chunking theo kích thước cố định	Chunking theo ngữ nghĩa	Chunking dựa trên LLM
Cách thực hiện	Chia văn bản theo số lượng token/ký tự đã cấu hình trước (ví dụ: 500 token, overlap 50 token).	Dựa vào ý nghĩa: phân tích từng câu, tính toán embedding và ghép các phân đoạn có độ tương tự.	Sử dụng LLM để phân tích và chia chunk trực tiếp dựa trên ngữ nghĩa và cấu trúc tự nhiên.

		đồng cao thành một chunk.	
Ưu điểm	<ul style="list-style-type: none"> - Dễ triển khai - Chunk đồng đều - Phù hợp cho xử lý hàng loạt 	<ul style="list-style-type: none"> - Bảo toàn ngữ cảnh và ý tưởng đầy đủ - Tăng độ chính xác trong truy xuất thông tin - Phù hợp cho các đoạn văn bản phức tạp dài 	<ul style="list-style-type: none"> - Hiểu sâu sắc ngữ nghĩa - Chunking gần với “con người” - Không cần các quy tắc heuristic phức tạp
Nhược điểm	<ul style="list-style-type: none"> - Dễ cắt câu, phá vỡ ngữ cảnh - Giảm chất lượng đầu vào cho LLM 	<ul style="list-style-type: none"> - Phụ thuộc ngữ境 tương đồng - Tốn tài nguyên tính toán cho embedding và so sánh - Có thể bị sai lệch nếu ngữ境 không hợp lý 	<ul style="list-style-type: none"> - Rất tốn tài nguyên (chi phí, thời gian) - Bị giới hạn bởi context window của LLM - Phụ thuộc vào chất lượng LLM và prompt
Tính phức tạp triển khai	Thấp	Trung bình đến cao	Cao
Chi phí xử lý	Thấp	Trung bình	Cao
Khả năng bảo toàn ngữ nghĩa	Thấp	Cao	Rất cao
Phù hợp với văn bản	Có cấu trúc đơn giản hoặc ngắn	Có cấu trúc rõ ràng, văn bản dài có ý tưởng phân	Phức tạp, giàu ngữ nghĩa hoặc cần độ chính xác cao

		tách tương đối độc lập	
--	--	---------------------------	--

Bảng 2.1 So sánh giữa các phương pháp chunking

2.3.3. Hạn chế của RAG truyền thống

Mặc dù có nhiều ưu điểm, các hệ thống RAG truyền thống, chủ yếu dựa trên việc truy xuất các đoạn văn bản phẳng, phải đối mặt với một số hạn chế đáng kể, đặc biệt là khi xử lý các tập dữ liệu phức tạp và các truy vấn đòi hỏi sự hiểu biết sâu sắc về mối quan hệ giữa các thông tin:

- Bỏ qua các mối quan hệ:** Các phương pháp RAG tiêu chuẩn thường xử lý các đoạn thông tin như những đơn vị riêng lẻ, bỏ qua các mối quan hệ cấu trúc và ngữ nghĩa phức tạp vốn có trong nhiều bộ dữ liệu thực tế. Kiến thức thường không tồn tại độc lập mà được kết nối với nhau, việc làm phẳng kiến thức này thành các đoạn rời rạc sẽ làm mất đi bối cảnh quan trọng và không nắm bắt được các kết nối quan trọng.
- Các vấn đề về độ sâu ngữ cảnh:** Quá trình chia tài liệu thành các đoạn nhỏ hơn, mặc dù cần thiết cho việc lập chỉ mục hiệu quả, có thể vô tình làm mất đi ngữ cảnh rộng hơn và các kết nối phân cấp trong tài liệu gốc. Điều này có thể ảnh hưởng tiêu cực đến độ chính xác của việc truy xuất và khả năng hiểu ngữ cảnh đầy đủ của LLM, vì kiến thức truy xuất có thể bị phân mảnh và thiếu cấu trúc phân cấp rõ ràng.
- Hiểu truy vấn phức tạp:** RAG truyền thống gặp khó khăn trong việc hiểu và trả lời các truy vấn phức tạp, đặc biệt là những truy vấn đòi hỏi suy luận đa bước (multi-hop reasoning). Việc truy xuất dựa trên sự tương đồng của các đoạn riêng lẻ có thể không đủ để thu thập tất cả các thông tin cần thiết và các mối liên hệ giữa chúng.

Những hạn chế này cho thấy sự cần thiết của một cách tiếp cận tinh vi hơn có thể tận dụng cấu trúc vốn có trong dữ liệu để cải thiện cả quá trình truy xuất và tạo sinh.

2.4 GRAG(Graph Retrieval-Augmented Generation)

- GRAG nổi lên như một sự phát triển của RAG nhằm giải quyết những hạn chế nói trên bằng cách tận dụng sức mạnh của các cấu trúc đồ thị, đặc biệt là KG. Thay vì xử lý dữ liệu dưới dạng văn bản phẳng, GRAG tổ chức thông tin thành một đồ thị, nơi các nút đại diện cho các thực thể (ví dụ: người, địa điểm, khái niệm) và các cạnh đại diện cho các mối quan hệ giữa chúng.
- Ý tưởng cốt lõi là chuyển đổi hoặc biểu diễn dữ liệu nguồn thành một KG. Khi nhận được truy vấn, thay vì chỉ truy xuất các đoạn văn bản, hệ thống GRAG sẽ truy xuất các phần tử đồ thị có liên quan – chẳng hạn như các nút cụ thể, các bộ ba biểu thị các mối quan hệ (ví dụ: đầu, quan hệ, đuôi), các đường dẫn thể hiện chuỗi kết nối hoặc toàn bộ đồ thị con. Các phần tử đồ thị được truy xuất này, mang thông tin cấu trúc và quan hệ rõ ràng, sau đó được sử dụng để tăng cường LLM tạo ra phản hồi.
- Bằng cách này, GRAG được thiết kế để nắm bắt rõ ràng các mối quan hệ giữa các thực thể, cho phép truy xuất kiến thức bảo toàn ngữ cảnh hơn và tạo điều kiện cho suy luận đa bước phức tạp bằng cách duyệt qua các kết nối trong đồ thị. Nó hứa hẹn cải thiện độ chính xác, độ sâu ngữ cảnh và khả năng giải thích của các hệ thống RAG, đặc biệt đối với các nhiệm vụ đòi hỏi sự hiểu biết về các mối quan hệ phức tạp. Các nghiên cứu ban đầu cho thấy rằng việc cấu trúc hóa kiến thức ngầm định từ văn bản thành đồ thị có thể mang lại lợi ích cho một số nhiệm vụ nhất định, mở rộng ứng dụng của GRAG từ dữ liệu đồ thị sang dữ liệu dựa trên văn bản nói chung.

2.4.1. Ưu điểm cốt lõi của GRAG

Việc sử dụng KG làm nền tảng mang lại cho GRAG một số lợi thế đáng kể so với các phương pháp RAG truyền thống:

- **Giảm ảo giác:** Bằng cách cung cấp cho LLM kiến thức có cấu trúc, dựa trên thực tế từ KG, GRAG có thể giảm đáng kể xu hướng ảo giác của mô hình.
- **Hiểu biết ngữ cảnh sâu sắc hơn:** Thay vì các đoạn văn bản riêng lẻ, GRAG truy xuất các phần tử đồ thị được kết nối với nhau (đường dẫn, đồ thị con). Điều này cung cấp cho LLM một ngữ cảnh phong phú hơn, đa diện hơn, nắm

bắt được các mối quan hệ và cấu trúc xung quanh thông tin liên quan. Việc bảo tồn thông tin cấu trúc này cho phép hiểu sâu hơn về chủ đề được đề cập.

- **Suy luận đa bước:** Cấu trúc đồ thị vốn dĩ hỗ trợ việc điều hướng qua nhiều bước hoặc "bước nhảy" (hops) giữa các thực thể được kết nối. GraphRAG vượt trội trong các nhiệm vụ đòi hỏi phải liên kết các mẩu thông tin khác nhau nằm rải rác trên các nút hoặc tài liệu khác nhau bằng cách đi theo các đường dẫn quan hệ trong KG. Điều này cho phép trả lời các câu hỏi phức tạp mà RAG truyền thống thường gặp khó khăn.²
- **Khả năng giải thích:** Vì thông tin được truy xuất đến từ một KG có cấu trúc, nên thường dễ dàng hơn để truy tìm nguồn gốc của một câu trả lời. Điều này cung cấp một "dấu vết kiểm toán" hoặc khả năng giải thích rõ ràng hơn cho các kết quả đầu ra của LLM, điều này rất quan trọng đối với việc xây dựng lòng tin và cho các ứng dụng trong các lĩnh vực được quản lý chặt chẽ.

2.4.2. Neo4j

- **Mô hình Đồ thị Thuộc tính Gắn nhãn**

- Neo4j sử dụng mô hình Đồ thị Thuộc tính Gắn nhãn để biểu diễn dữ liệu. Mô hình này trực quan và linh hoạt, phản ánh chặt chẽ cách chúng ta thường hình dung các hệ thống được kết nối. Các thành phần cốt lõi của nó bao gồm:
 - **Nút:** Đại diện cho các thực thể hoặc đối tượng riêng biệt trong miền dữ liệu (ví dụ: sinh viên, nhà trường, chương trình đào tạo).
 - **Nhãn:** Được sử dụng để phân loại các nút thành các nhóm hoặc loại (ví dụ: :email, :organization, :student). Một nút có thể có không hoặc nhiều nhãn, cho phép phân loại đa diện. Nhãn rất quan trọng cho việc lập chỉ mục và tổ chức lược đồ, mặc dù Neo4j về cơ bản là tùy chọn lược đồ (schema-optional).
 - **Mối quan hệ:** Đại diện cho các kết nối có ý nghĩa giữa các nút. Các mối quan hệ trong Neo4j luôn có hướng, có chính xác

một loại để xác định ngữ nghĩa của kết nối (ví dụ: :bao_gồm, :tọa_lạc_tại, :ghi_vào).

- **Thuộc tính:** Là các cặp value-key trị được lưu trữ trên cả nút và mỗi quan hệ để chứa dữ liệu chi tiết. Giá trị có thể là các kiểu dữ liệu nguyên thủy (số, chuỗi, boolean) hoặc danh sách đồng nhất của các kiểu nguyên thủy đó.

- **Cypher**

- Cypher là ngôn ngữ truy vấn đồ thị khai báo, mạnh mẽ của Neo4j. Nó được thiết kế đặc biệt để thể hiện các mẫu đồ thị và điều hướng các mối quan hệ một cách hiệu quả và trực quan. Các tính năng chính của Cypher bao gồm:

Khớp mẫu: Cypher sử dụng cú pháp giống như nghệ thuật ASCII để mô tả các mẫu nút và mối quan hệ cần tìm trong đồ thị.

Ví dụ:



```
MATCH (document:Document {name: {document_id}})-[*]->(predict:Section {name: 'quá trình hình thành và phát triển'})
MATCH p=(predict)-[r*1..2]->(e)
RETURN p
```

Hình 2.6 Câu cypher lấy ra đồ thị “quá trình hình thành và phát triển”

Câu cypher trên dùng để lấy ra các mối quan hệ đến hai node tính từ node có name là “quá trình hình thành và phát triển”.

Khai báo: Người dùng chỉ định cái gì cần truy xuất (mẫu cần khớp), chứ không phải cách thực hiện việc duyệt đồ thị cụ thể. Công cụ truy vấn Neo4j sẽ tối ưu hóa việc thực thi.

Tối ưu hóa cho Đồ thị: Không giống như SQL, vốn dựa vào các phép toán JOIN tốn kém để điều hướng các mối quan hệ trong cơ sở dữ liệu quan hệ, Cypher được tối ưu hóa cho việc duyệt đồ thị hiệu quả. Điều này đặc biệt quan trọng đối với các truy vấn đa bước phổ biến trong GRAG.

2.5 So sánh RAG và GRAG

Đặc điểm/Khía cạnh	RAG	GRAG
Cấu trúc Dữ liệu Cơ bản	Phi cấu trúc	Cấu trúc
Cơ chế Truy xuất	Tương đồng ngữ nghĩa (tìm kiếm vector), tìm kiếm từ khóa	Duyệt đồ thị (BFS, DFS), khớp mẫu (Cypher), có thể kết hợp tìm kiếm vector
Xử lý Mối quan hệ & Ngữ cảnh	Bỏ qua mối quan hệ cấu trúc rõ ràng; ngữ cảnh giới hạn trong đoạn	Tận dụng mối quan hệ rõ ràng; ngữ cảnh sâu hơn thông qua cấu trúc đồ thị
Hiệu suất QA Đơn bước	Tốt	Kém
Hiệu suất QA Đa bước	Kém	Tốt
Điểm yếu Chính	"Mù" mối quan hệ, hạn chế ngữ cảnh	Phụ thuộc chất lượng đồ thị, phức tạp hơn, chi phí xây dựng đồ thị

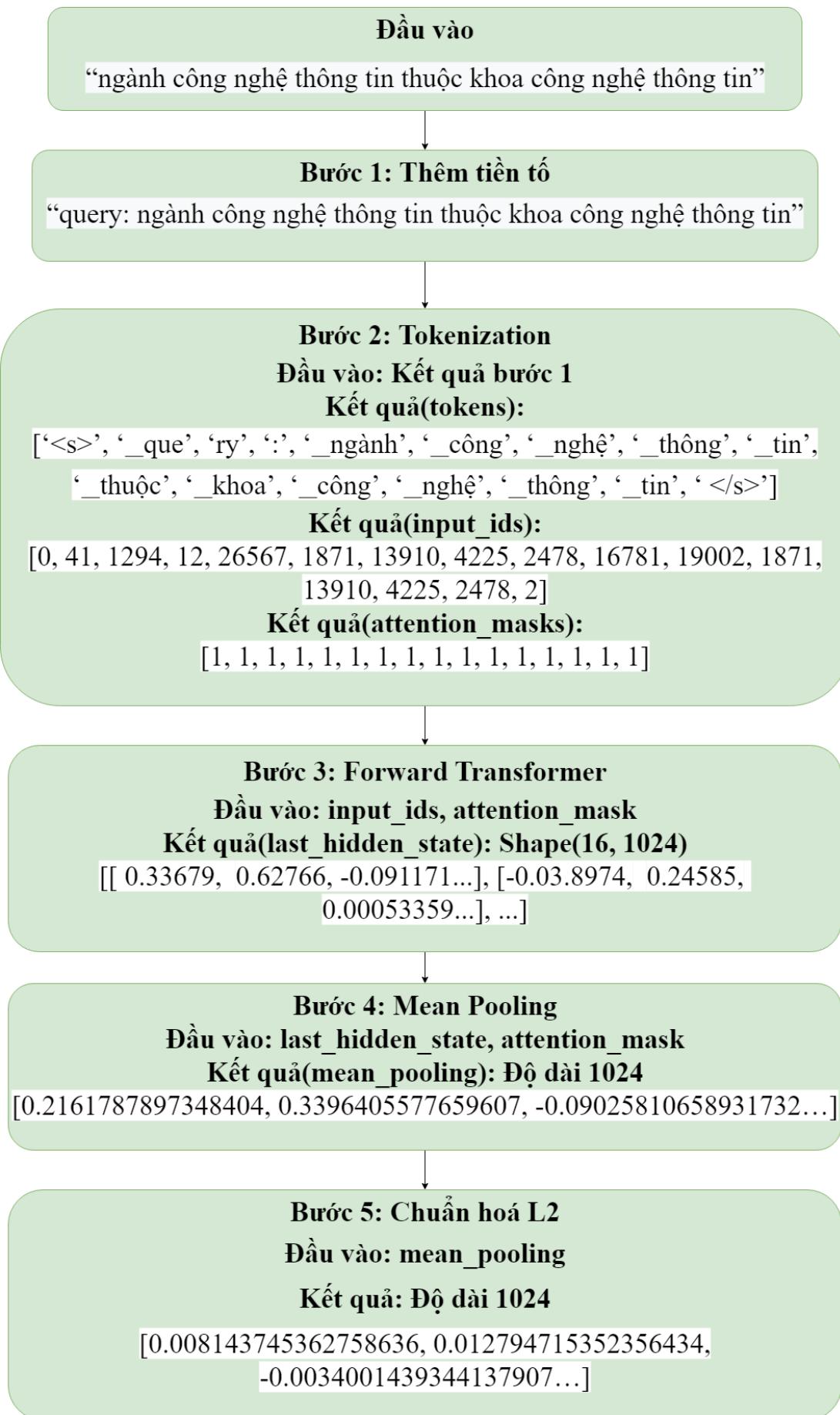
Bảng 2.2 So sánh giữa RAG và GRAG

2.6 Các mô hình nhúng được sử dụng

`intfloat/multilingual-e5-large-instruct: 1024` chiều [17].

- Đây là một mô hình nhúng đa ngôn ngữ thuộc họ E5.
- Được huấn luyện theo kiểu instruction-tuning (học từ các chỉ dẫn). Nghĩa là khi nhúng, không chỉ đưa tài liệu gốc vào, mà còn có thể thêm tiền tố như "query: ..." hoặc "passage: ..." để giúp mô hình phân biệt ngữ cảnh tìm kiếm.

Ví dụ:



Hình 2.7 Quy trình tạo ra nhúng của mô hình 1024 chiều

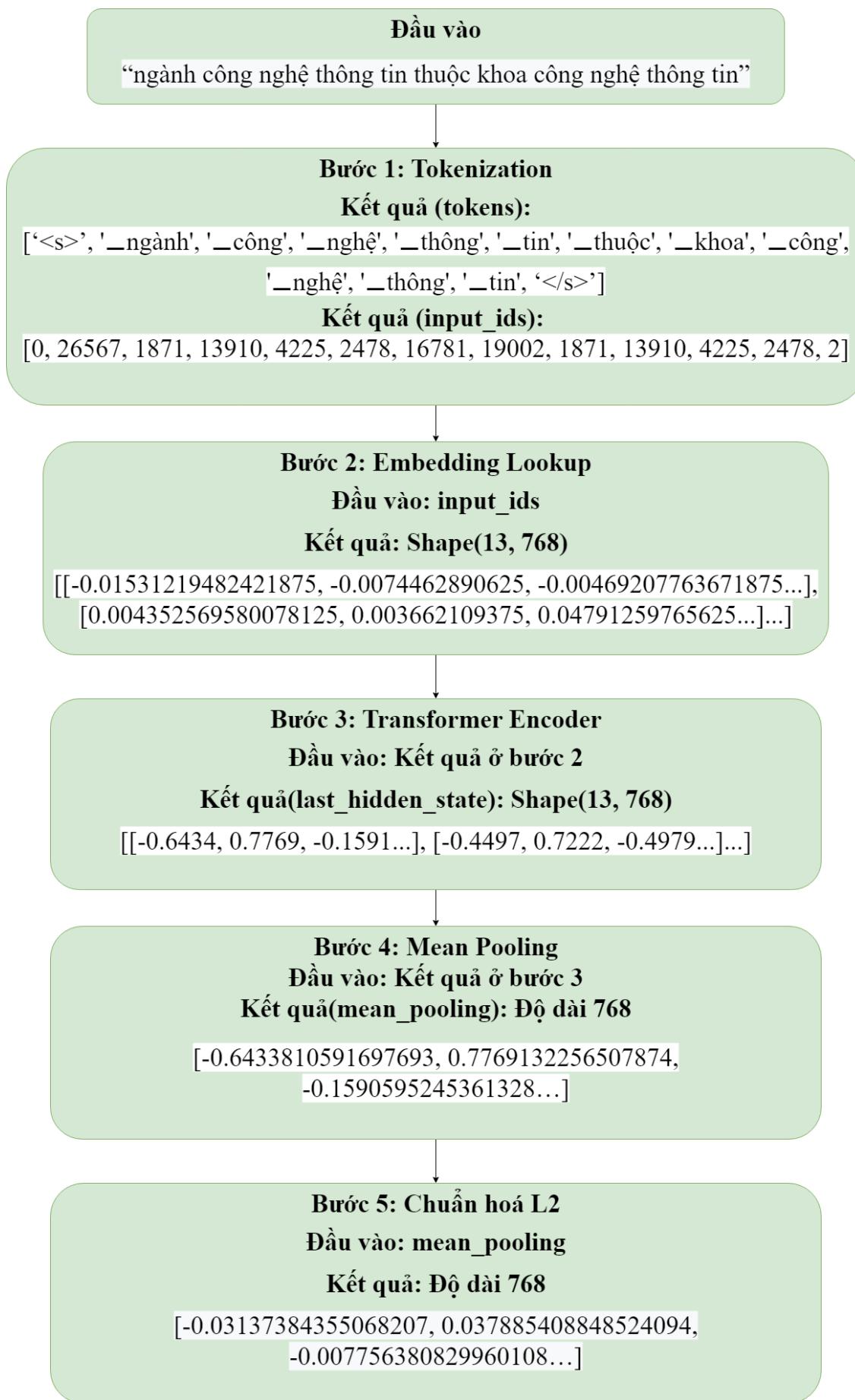
Alibaba-NLP/gte-multilingual-base: 768 chiều [18].

- Đây là mô hình General Text Embedding của Alibaba, hỗ trợ đa ngôn ngữ.

Đặc điểm:

- Hiệu năng khá cao trong các nhiệm vụ IR.
- Nhẹ hơn so với mô hình E5.

Ví dụ:



Hình 2.8 Quy trình tạo ra nhúng của mô hình 768 chiều **sentence-transformers/distiluse-base-multilingual-cased-v2: 512 chiều** [19].

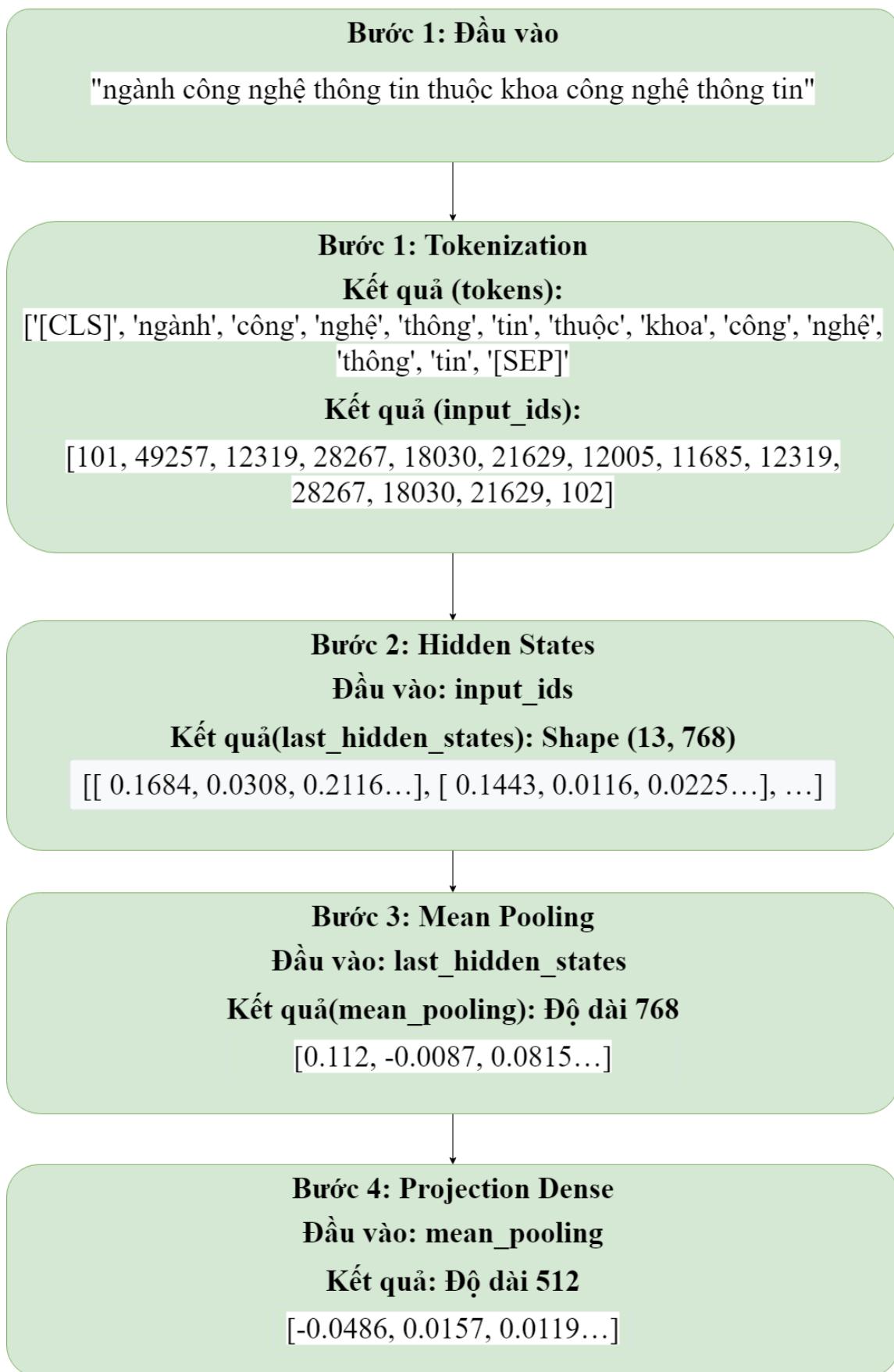
- Đây là một phiên bản rút gọn của Universal Sentence Encoder cho đa ngôn ngữ.
- Universal Sentence Encoder: Là mô hình của Google, tạo nhúng câu 512 chiều, ứng dụng trong tìm kiếm ngữ nghĩa.
- Được huấn luyện bằng phương pháp Knowledge Distillation → nhẹ hơn, tốc độ nhanh hơn, phù hợp môi trường tài nguyên hạn chế.
- Knowledge Distillation:
 - Là kỹ thuật trong học máy, trong đó một mô hình nhỏ (student) được huấn luyện để bắt chước mô hình lớn (teacher) thông qua “soft labels” (xác suất đầu ra của teacher) thay vì chỉ dùng nhãn cứng (hard labels).
 - Nhờ vậy, student học được nhiều thông tin ngữ nghĩa ẩn mà teacher đã học được từ dữ liệu lớn.

Ưu điểm:

- Kích thước nhúng nhỏ tiết kiệm bộ nhớ.
- Rất nhanh, thích hợp cho hệ thống cần tốc độ cao.

Nhược điểm: Độ chính xác thấp hơn các mô hình mới như E5 hay GTE.

Ví dụ:



Hình 2.9 Quy trình tạo ra nhúng của mô hình 512 chiều

colbert-ir/colbertv2.0: số lượng vector tùy thuộc vào độ dài văn bản mỗi vector 128 chiều [20].

- Đây là mô hình đặc biệt dành cho IR, không giống các mô hình nhúng thông thường(mỗi tài liệu tạo ra một vector duy nhất).
- Thay vì nén cả câu thành một vector duy nhất, ColBERT mã hóa mỗi token trong tài liệu thành vector 128 chiều.

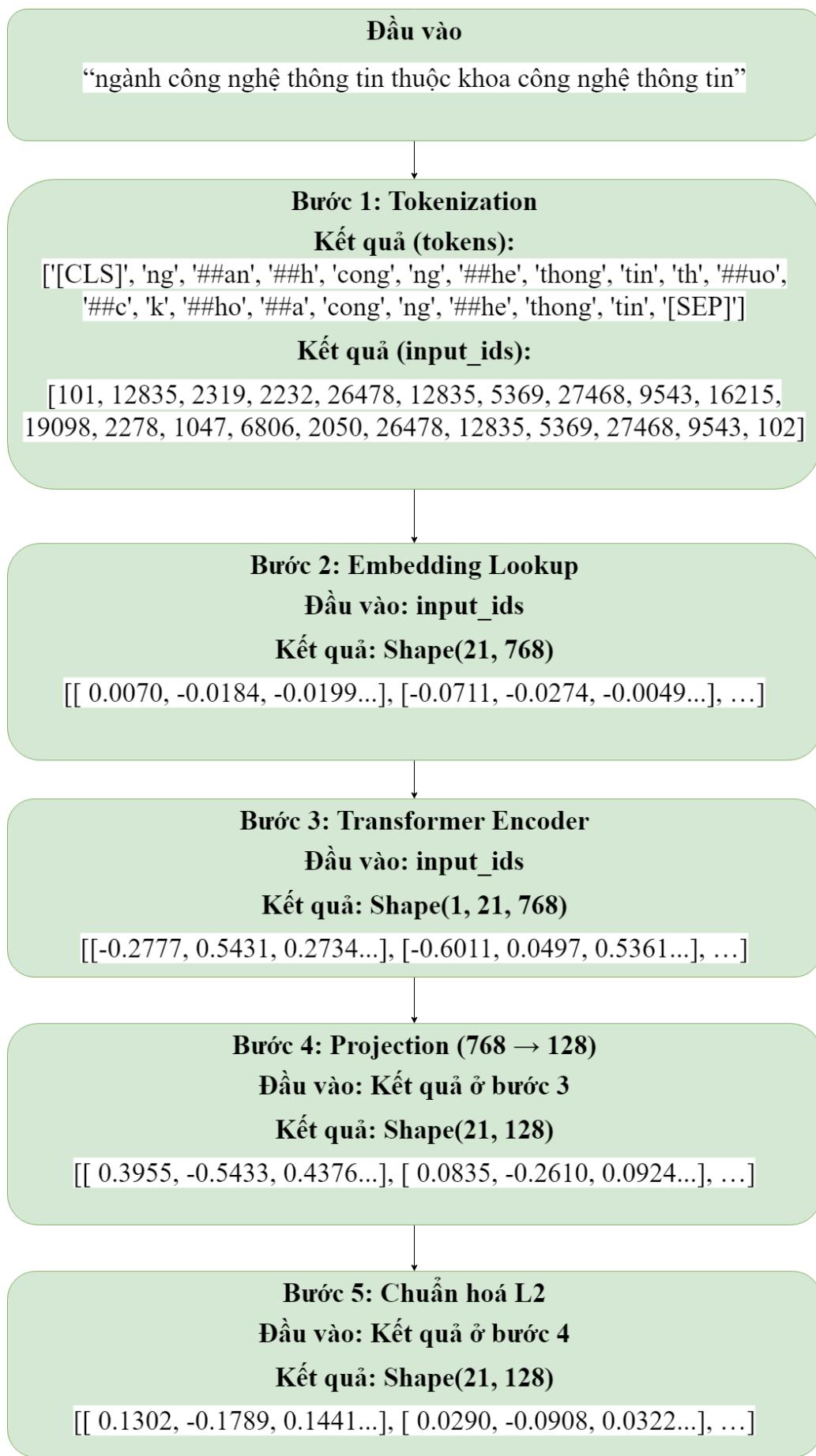
Ưu điểm:

- Khả năng tìm kiếm chính xác hơn trong hàng triệu vector.
- Giữ được thông tin chi tiết trong từng token, thay vì nén hết thành 1 vector.

Nhược điểm:

- Tốn bộ nhớ hơn, vì mỗi tài liệu là một đa vector.

Ví dụ:



Hình 2.10 Quy trình tạo ra nhúng của mô hình late interaction

nvidia/llama-3.2-nv-rerankqa-1b-v2

- Đây là mô hình reranking (xếp hạng lại), được tối ưu cho các nhiệm vụ IR.
- Khác với ColBERT hay các phương pháp truyền thống (lấy k tài liệu có độ đo cao nhất), mô hình này hoạt động ở bước rerank để sắp xếp lại các tài liệu sao cho phù hợp nhất với truy vấn <- giải thích lại.

Khác biệt chính:

- ColBERT tạo ra đa vector và dùng late interaction để xếp hạng lại các vector.
- Late interaction:
 - Còn nvidia/llama-3.2-nv-rerankqa-1b-v2 sử dụng cross-encoder và xếp hạng lại các vector liên quan.
- Cross-encoder và bi-encoder:
 - o Bi-encoder (ColBERT): Nhúng riêng truy vấn và tài liệu → xếp hạng.
 - o Cross-encoder (nvidia/llama-3.2-nv-rerankqa-1b-v2): Đưa truy vấn và tài liệu vào mô hình → mô hình học cách hiểu ngữ cảnh kết hợp.

Ưu điểm:

- Độ chính xác cao hơn các phương pháp đơn thuần, vì hiểu được ngữ cảnh sâu giữa truy vấn và tài liệu.

Nhược điểm:

- Tốn chi phí tính toán hơn vì phải chạy mô hình cho từng cặp truy vấn và tài liệu.
- Không thể áp dụng trực tiếp cho hàng trăm vector → cần kết hợp với các phương pháp khác để giảm số lượng tài liệu đầu vào.

CHƯƠNG 3. PHƯƠNG PHÁP LUẬN

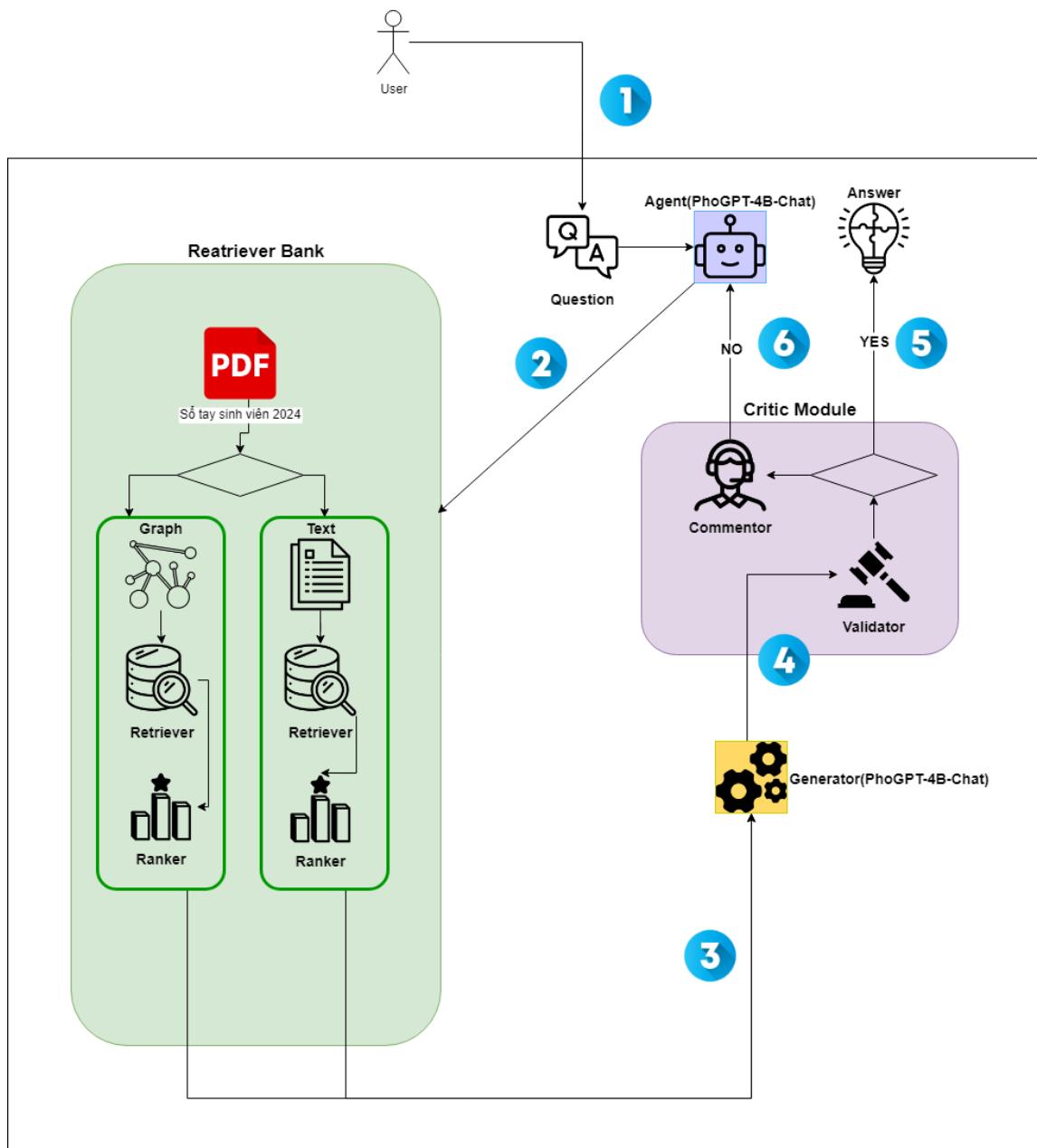
3.1 Phát biểu bài toán

Mặc dù LLMs đã có những tiến bộ đáng kể, chúng vẫn đối mặt với các hạn chế như tạo ra thông tin không chính xác(hallucination) và thiếu khả năng cập nhật kiến thức. Kỹ thuật RAG giải quyết một phần vấn đề này bằng cách truy xuất từ các nguồn dữ liệu, nhưng lại thường bỏ qua các mối quan hệ cấu trúc phức tạp giữa các mẩu thông tin. Ngược lại, GRAG tận dụng KG để nắm bắt các mối quan hệ này,

nhưng có thể bị giới hạn trong việc truy cập ngữ cảnh văn bản rộng lớn. Thách thức cốt lõi hiện nay là nhiều truy vấn trong thế giới thực đòi hỏi sự kết hợp của dữ liệu phi cấu trúc và dữ liệu cấu trúc. Các hệ thống hiện có thường chỉ chuyên biệt hóa vào một trong hai phương thức truy xuất này, thiếu khả năng lựa chọn và tích hợp linh hoạt nguồn kiến thức phù hợp nhất dựa trên yêu cầu của từng câu hỏi cụ thể. Điều này tạo ra nhu cầu về một hệ thống làm việc thống nhất, có khả năng thích ứng, có thể hoạt động hiệu quả việc sử dụng cả truy xuất dựa trên văn bản và truy xuất dựa trên đồ thị để tạo ra các câu trả lời chính xác và toàn diện hơn.

3.2 Phương pháp giải quyết bài toán

Để giải quyết bài toán được nêu ra trong đề bài chúng em xây dựng một hệ thống làm việc duy nhất, gồm các thành phần sau đây:



Hình 3.1 Kiến trúc chính của hệ thống

- **Agent:** Thành phần trung tâm, chịu trách nhiệm trích xuất các thực thể hữu ích có trong câu hỏi. Dựa trên câu hỏi đầu vào và phản hồi từ vòng lặp trước đó (nếu có), Agent xác định các thực thể cần truy xuất trong mỗi vòng lặp.
- **Retriever Bank:** Tập hợp các module truy xuất, bao gồm một mô-đun truy xuất văn bản và một mô-đun truy xuất đồ thị. Mỗi mô-đun thường có bộ truy xuất và bộ xếp hạng riêng.
- **Generator:** LLM chịu trách nhiệm tổng hợp câu trả lời cuối cùng. Nó nhận đầu vào là câu hỏi gốc và tài liệu tham khảo được truy xuất.

- **Critic Module:** Thành phần đánh giá và cung cấp phản hồi. Chức năng của nó là đánh giá kết quả đầu ra của Generator và cung cấp hướng dẫn sửa chữa cho Agent nếu câu trả lời bị coi là không chính xác. Module này bao gồm hai thành phần con:
 - **Validator:** Thành phần quyết định xem câu trả lời đảm bảo sự chính xác và toàn vẹn câu hỏi hay không. Nhiệm vụ nhị phân.
 - **Commentor:** Thành phần phản hồi về Agent nếu câu trả lời không chính xác, để Agent có thể tự điều chỉnh hành vi của mình.

3.3 Quy trình tương tác của hệ thống

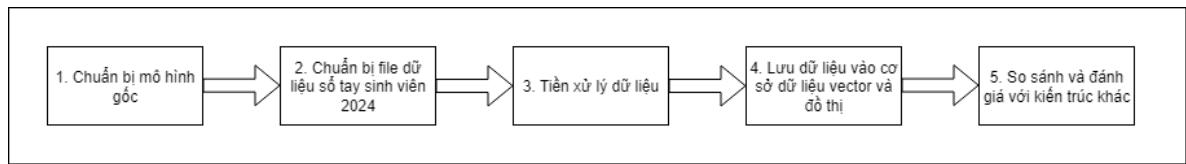
Một chu trình tự điều chỉnh hành động lặp đi lặp lại:

- **Agent:** Nhận câu hỏi (và phản hồi từ vòng lặp trước, nếu có) và xác định các thực thể cần trả lời trong câu hỏi.
- **Retrieval Bank:** Từ phản hồi của Agent hệ thống sẽ vào ngân hàng truy xuất phù hợp để truy xuất thông tin.
- **Sinh tạo câu trả lời:** Generator sẽ sử dụng câu hỏi và ngữ cảnh được truy xuất để tạo ra một câu trả lời.
- **Đánh giá phản biện:** Validator đánh giá câu trả lời vừa được sinh tạo, đưa ra quyết định chấp nhận hoặc từ chối dựa trên định tính và định lượng.
- **Tạo bình luận (nếu bị từ chối):** Nếu câu trả lời bị từ chối, Commentor sẽ tạo ra phản hồi chi tiết cho Agent.
- **Kết thúc hoặc Lặp lại:**
 - Nếu câu trả lời được chấp nhận hoặc đạt đến số vòng lặp tối đa, hệ thống xuất ra câu trả lời cuối cùng.
 - Nếu câu trả lời bị từ chối và số vòng lặp chưa đạt tối đa, Agent nhận phản hồi từ Commentor và bắt đầu một vòng lặp mới, thực hiện cơ chế tự suy ngẫm (Self-Reflection).

Cơ chế tự suy ngẫm này là cốt lõi của khả năng học hỏi và thích ứng ngay trong quá trình xử lý một truy vấn duy nhất. Agent sử dụng phản hồi cụ thể từ Commentor để tinh chỉnh sự hiểu biết của nó về truy vấn hoặc chiến lược truy xuất trong vòng lặp tiếp theo. Thiết kế lặp đi lặp lại kết hợp với phản hồi từ Commentor tạo ra một

vòng lặp nhân quả trực tiếp để cải thiện hiệu suất. Một thất bại ban đầu (ví dụ: chọn sai bộ truy xuất) trực tiếp kích hoạt phản hồi, thông báo cho một hành động sửa chữa trong vòng lặp tiếp theo, làm tăng khả năng thành công. Agent sử dụng phản hồi cho vòng lặp kế tiếp và phản hồi này rất cụ thể (ví dụ: thực thể không chính xác, đề xuất bộ truy xuất khác). Tính cụ thể này cho phép Agent giải quyết trực tiếp lỗi đã được chẩn đoán. Điều này trái ngược với các hệ thống tĩnh, nơi lỗi truy xuất chỉ đơn giản dẫn đến đầu ra cuối cùng kém chất lượng.

3.4 Quy trình thực hiện



Hình 3.2 Quy trình thực hiện

3.4.1. Chuẩn bị các mô hình gốc

Trong hệ thống của chúng em mỗi thành phần đều có vai trò khác nhau bao gồm:

- **Agent:**
 - **Nhiệm vụ:**
 - Đóng vai trò trung tâm, tương tác với Retriever Bank để trả lời câu hỏi.
 - Xác định các thực thể hữu ích để trả lời câu hỏi.
 - Tự phản ánh (self-reflection – quá trình hệ thống đánh giá lại hành động hoặc quyết định của mình dựa trên phản hồi) dựa trên phản hồi từ Commentor để cải thiện hành động ở các lần lặp tiếp theo.
 - **Yêu cầu:**
 - Phải phân tích câu hỏi để xác định thực thể hữu ích.
 - Cải thiện hành động dựa trên phản hồi từ Commentor để giảm lỗi trong việc chọn thực thể sai.
 - **Model:** Gemini-1.5-flash.
- **Generator:**
 - **Nhiệm vụ:**

- Đây là bộ phận tạo sinh câu trả lời. Nhận câu hỏi và thông tin khả thi từ nguồn truy xuất. Từ đó dựa vào thông tin khả thi để trả lời câu hỏi một cách chính xác, toàn diện và có cơ sở.
- Sử dụng prompting(hướng dẫn) kiểu chain-of-thought (CoT – phương pháp yêu cầu mô hình AI trình bày tuần tự các bước suy nghĩ hoặc lý luận) để giải quyết câu hỏi một cách có hệ thống (theo trình tự logic, từng bước một, thay vì trả lời trực tiếp).

- **Yêu cầu:**

- Phải tích hợp thông tin từ tài liệu tham chiếu để tạo câu trả lời chính xác.

- **Model:** Gemini-1.5-flash

- **Validator:**

- **Nhiệm vụ:**

- Đây là bộ phận phân loại nhị phân. Dựa trên các độ đo:
qa_mean: Trung bình cosine simalirity của ba mô hình nhúng giữa câu hỏi và câu trả lời.
qd_mean: Trung bình cosine simalirity cao nhất của ba mô hình nhúng giữa câu hỏi và tài liệu tham chiếu.
ad_mean: Trung bình cosine simalirity cao nhất của ba mô hình nhúng giữa câu trả lời và tài liệu tham chiếu.
Xác minh tính chính xác của đầu ra được tạo bởi Generator, dựa trên câu hỏi.
- Quyết định xem đầu ra có được chấp nhận làm câu trả lời cuối cùng hay không.

- **Yêu cầu:**

- Phải hoạt động hiệu quả với bối cảnh được cung cấp, tránh bị phân tâm bởi thông tin không liên quan.
- Không yêu cầu fine-tuning, tận dụng khả năng của LLM pre-trained.

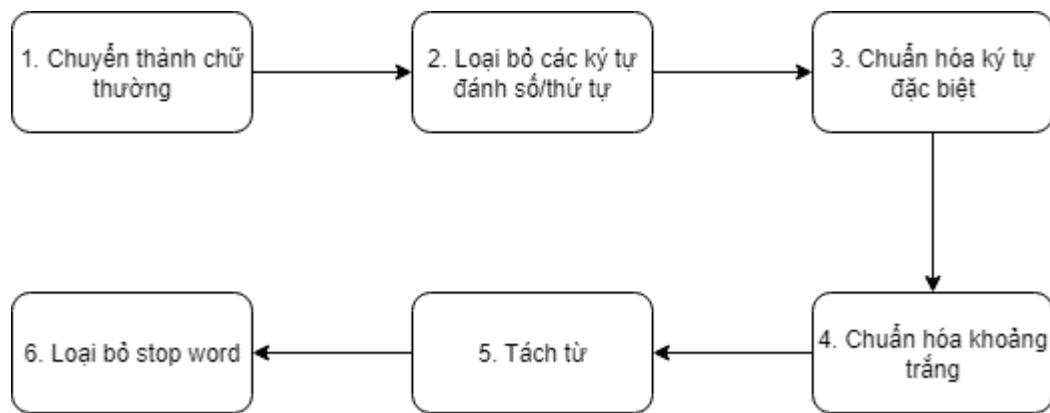
- Đánh giá bằng định tính thông qua mức độ hiểu biết ngôn ngữ tự nhiên của Gemini-1.5-flash và định lượng thông qua các độ đo được cung cấp.
- **Model:** Gemini-1.5-flash
- **Commentor:**
 - **Nhiệm vụ:**
 - Đây là bộ phận tạo sinh phản hồi. Nhận câu hỏi, thông tin khả thi và mô-đun truy xuất trước đó để tạo ra bình luận gửi đến Agent tự điều chỉnh hành vi.
 - Nếu đâu ra bị Validator từ chối, Commentor cung cấp phản hồi để giúp Agent cải thiện hành động ở lần lặp tiếp theo.
 - **Yêu cầu:**
 - Phản hồi phải mang tính xây dựng, tránh gây nhầm lẫn hoặc cung cấp thông tin không cần thiết.
 - **Model:** Gemini-1.5-flash

3.4.2. Chuẩn bị tập dữ liệu

- Tập dữ liệu là sổ tay sinh viên Nông Lâm 2024:
<https://nls.hcmuaf.edu.vn/contents.php?ur=nls&ids=42921>.
- Gồm 76 trang. Nhưng chỉ lấy 65 trang(bỏ 9 trang đầu và 2 trang cuối).

3.4.3. Tiền xử lý dữ liệu

- Tiền xử lý dữ liệu đóng vai trò then chốt trong quy trình lưu dữ liệu vào nguồn truy xuất. Giai đoạn này bao gồm hàng loạt các bước nhằm làm sạch, chuyển đổi và chuẩn hóa dữ liệu thô, giúp cải thiện chất lượng thông tin đầu vào và nâng cao hiệu suất nhận diện văn bản.



Hình 3.3 Quy trình tiền xử lý dữ liệu

- Chuyển thành chữ thường: Chuyển tất cả các ký tự về dạng chữ thường để đồng nhất hóa dữ liệu văn bản, tránh sự khác biệt giữa chữ hoa và chữ thường (ví dụ: "Nhà" và "nhà" được coi là giống nhau).
- Loại bỏ các ký tự đánh số/thứ tự: Xóa các ký tự số hoặc ký tự thứ tự (ví dụ: "1.", "2)", "a.", v.v.) không cần thiết để làm sạch văn bản.
- Chuẩn hóa ký tự đặc biệt: Thay thế hoặc loại bỏ các ký tự đặc biệt (ví dụ: @, #, \$, %, &, v.v.) bằng ký tự phù hợp hoặc khoảng trắng.
- Chuẩn hóa khoảng trắng: Xóa các khoảng trắng thừa, thay thế nhiều khoảng trắng liên tiếp bằng một khoảng trắng duy nhất, đồng thời đảm bảo không có khoảng trắng ở đầu hoặc cuối chuỗi.
- Tách từ: Tách các từ hoặc cụm từ trong văn bản tiếng Việt bằng cách sử dụng các công cụ tách từ chuyên biệt VnCoreNLP để đảm bảo phân biệt được từ đơn và từ ghép.
- Loại bỏ stop word: Xóa các từ không mang nhiều ý nghĩa trong phân tích ngữ nghĩa (stop words) như "là", "của", "và", "thì", để giảm nhiễu và tập trung vào các từ khóa quan trọng.

3.4.4. Tạo ngân hàng dữ liệu

- Lưu dữ liệu vào Qdrant

Chúng em sử dụng LLM để chia chunk. Sau đó xem xét lại ý nghĩa của từng chunk.

Sau đó tiến hành nhúng từng chunk thành các vector 1024, 768, 512 và vector late interacion và lưu vào Qdrant.

- Lưu dữ liệu vào neo4j

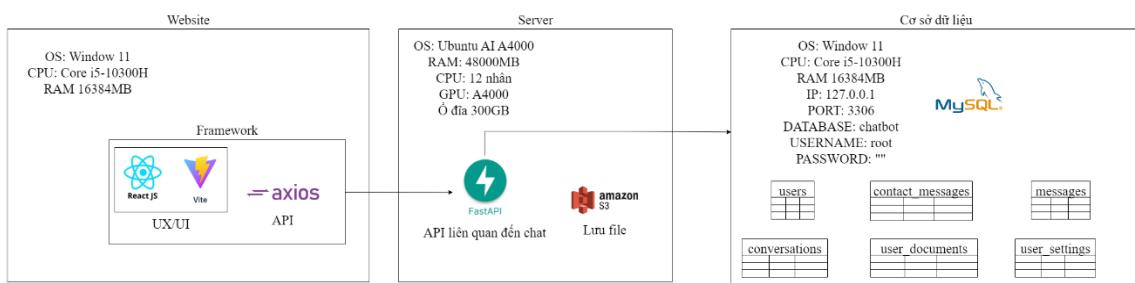
Chúng em sử dụng LLM (Large Language Model – mô hình ngôn ngữ lớn có khả năng sinh văn bản và hiểu ngữ cảnh) để tạo ra các nodes (các thực thể trong knowledge graph), properties (thuộc tính của từng thực thể) và relationships (mối quan hệ giữa các thực thể). Sau đó, thêm vào các tiêu đề tương ứng với từng nội dung (để giữ cấu trúc và định danh thông tin giống như trong file ban đầu).

3.4.5. So sánh và đánh giá với kiến trúc khác

Chúng em sử dụng LLM và nội dung trong file ban đầu để tạo ra các câu hỏi và câu trả lời và sử dụng 3 kiến trúc: RAG+GRAG+Agent, RAG và GRAG để tiến hành so sánh và đánh giá trên các chỉ số: độ tương đồng, accuracy, hallucination và missing.

CHƯƠNG 4. TRIỂN KHAI THỰC HIỆN

4.1 Tổng quan hệ thống



Hình 4.1 Các bước thực hiện hệ thống

4.2 Xây dựng nguồn kiến trúc

4.2.1. Trích xuất dữ liệu cho lưu trữ vector

Chúng em sử dụng Gemini 2.5 Pro Preview 05-06 để tách văn bản thành các đoạn nhỏ khác nhau được gọi là chunk.

Từ mỗi chunk đã được trích xuất trước đó. Chúng em tiếp tục sử dụng Gemini 2.5 Pro Preview 05-06 để tóm tắt chunk nhằm bổ sung ý nghĩa cho chunk.

4.2.2. Trích xuất dữ liệu cho lưu trữ đồ thị

Chúng em sử dụng Gemini 2.5 Pro Preview 06-06 để trích xuất thực thể và mối quan hệ.

Sau quá trình tự động chúng em sẽ kiểm tra thủ công để đảm bảo chất lượng của thực thể và quan hệ.

4.3 Xây dựng cơ sở dữ liệu vector Qdrant

Qdrant là một cơ sở dữ liệu vector mã nguồn mở được thiết kế để lưu trữ và tìm kiếm các vector cao chiều một cách hiệu quả. Nó thường được sử dụng trong các ứng dụng trí tuệ nhân tạo như tìm kiếm ngữ nghĩa, hệ thống gợi ý, hay xử lý dữ

liệu đa phương thức (văn bản, hình ảnh, âm thanh). Với khả năng xử lý nhanh các vector và hỗ trợ dữ liệu bổ sung, Qdrant đang trở thành lựa chọn phổ biến trong các dự án AI hiện đại.

4.3.1. Thiết kế mô hình dữ liệu cho Qdrant

Mô hình dữ liệu trong Qdrant đóng vai trò như một bản thiết kế, giúp tổ chức dữ liệu sao cho việc lưu trữ, tìm kiếm và quản lý trở nên tối ưu. Một mô hình được thiết kế tốt không chỉ tăng tốc độ xử lý mà còn đảm bảo dữ liệu dễ dàng được truy cập và mở rộng khi cần.

4.3.2. Các bước thiết kế mô hình dữ liệu

Mô hình dữ liệu của Qdrant được xây dựng dựa trên ba thành phần chính:

- **Points:** Đây là đơn vị cơ bản nhất trong Qdrant. Mỗi điểm bao gồm:
 - Các vector: Một dãy số đại diện cho dữ liệu.
 - Vector 512 chiều:
Sử dụng model “sentence-transformers/distiluse-base-multilingual-cased-v2”
 - Vector 768 chiều:
Sử dụng model “Alibaba-NLP/gte-multilingual-base”
 - Vector 1024 chiều:
Sử dụng model “intfloat/multilingual-e5-large-instruct”
 - Vector late interaction số lượng vector tùy thuộc vào độ dài văn bản mỗi vector có 128 chiều:
Sử dụng model “colbert-ir/colbertv2.0”
 - ID: Một giá trị duy nhất để phân biệt từng điểm.
 - Payload: Dữ liệu bổ sung dạng JSON chứa thông tin meta gồm: nội dung văn bản, tên phần, tên mục.
- **Collections:** Đây là nơi chứa các điểm, tương tự như bảng trong cơ sở dữ liệu truyền thống.
 - Index HNSW với cấu hình:
 - hnsw_config = {
"m": 16,

```
    "ef_construct": 1000,  
    "full_scan_threshold": 10000  
}
```

trong đó:

m: Số kết nối tối đa cho mỗi nút trong đồ thị (mặc định: 16)

ef_construct: Yếu tố xây dựng, ảnh hưởng đến chất lượng index (mặc định: 100).

full_scan_threshold: Ngưỡng để chuyển sang quét toàn bộ nếu tập dữ liệu nhỏ.

- Quantization_config: Để tối ưu hóa việc lưu trữ và tìm kiếm, sử dụng kỹ thuật nén vector gọi là quantization. Một cách thiết lập phổ biến là:
 - Dùng kiểu nén scalar với định dạng INT8 (số nguyên 8-bit) để giảm kích thước lưu trữ.
 - Đặt ngưỡng phân vị (quantile) là 0.99 để giữ lại 99% giá trị quan trọng của vector, tránh mất mát dữ liệu quá nhiều.
 - Luôn giữ dữ liệu trong RAM (always_ram=True) để tăng tốc độ truy cập thay vì đọc từ ổ cứng.
- Vectors_config: Vector là trái tim của Qdrant, và cấu hình này quyết định cách chúng được lưu trữ và xử lý. Một thiết kế linh hoạt có thể bao gồm:
 - Nhiều kích thước vector trong cùng một bộ sưu tập:
 - “matryoshka-1024dim”: Vector 1024 chiều.
 - “matryoshka-768dim”: Vector 768 chiều.
 - “matryoshka-512dim”: Vector 512 chiều.
 - “late_interaction”: số lượng vector tùy thuộc vào ngữ cảnh mỗi vector có 128 chiều với cơ chế so sánh đa vector, sử dụng MAX_SIM để ưu tiên độ tương đồng cao nhất.
 - Tất cả đều dùng kiểu dữ liệu FLOAT32 (số thực 32-bit) và thước đo tương đồng Cosine

4.3.3. Tạo embedding từ văn bản

- **Chọn mô hình nhúng**

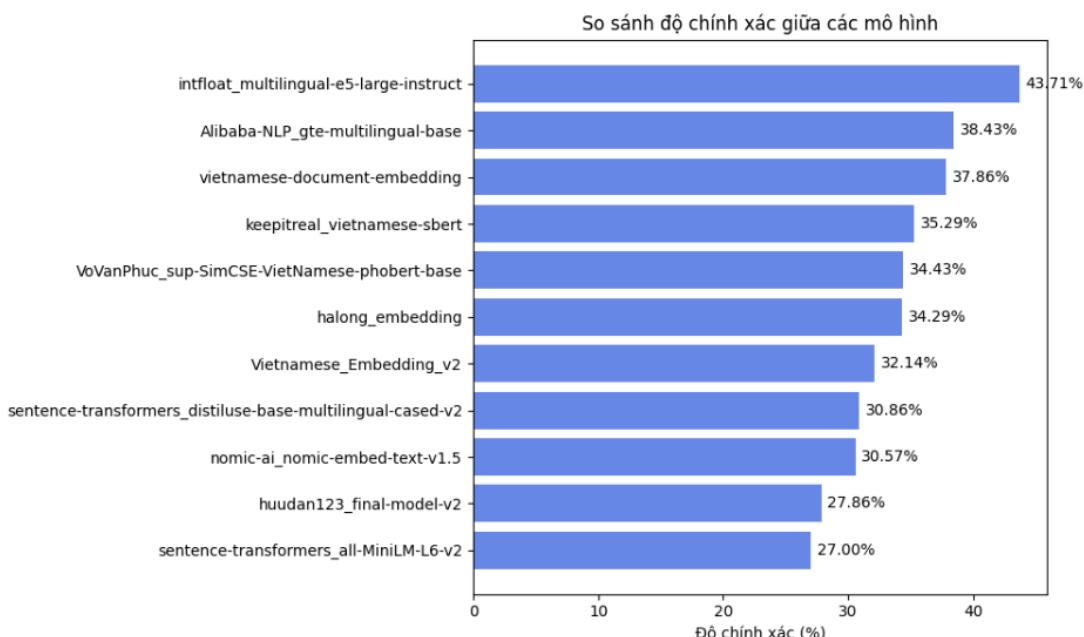
- **Tạo dữ liệu**

Hiện nay có rất nhiều mô hình nhúng khác nhau phù hợp với từng ngôn ngữ cụ thể và cũng có những mô hình đa ngôn ngữ. Để chọn mô hình nhúng tốt nhất với tiêu chí nhẹ và miễn phí, đồng thời vẫn giữ được hiệu suất tốt trong việc tìm kiếm văn bản. Chúng em đã tạo ra một tập dữ liệu tiếng việt gồm 700 câu hỏi với 7 lĩnh vực: sức khỏe, lịch sử, địa lý, văn hóa, thể thao, môi trường và giáo dục. Mỗi lĩnh vực gồm 100 câu hỏi. Mỗi câu hỏi tương ứng với 5 câu trả lời.

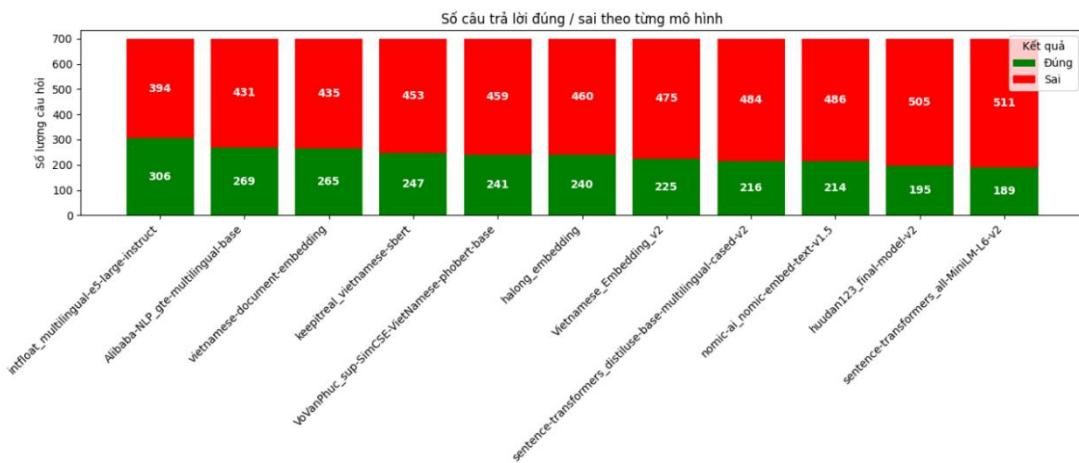
- **Kiểm tra mô hình nhúng**

Để chọn ra các mô hình nhúng, chúng em sử dụng GPT4, Grok3, bộ lọc của Hugging Face, MTEB ranking để tìm các mô hình nhúng phổ biến hiện nay.

Kết quả đã chọn ra 11 mô hình nhúng. Sau đó tiến hành nhúng câu hỏi và câu trả lời để tính độ tương đồng simaliriry với từng cặp câu hỏi và câu trả lời, câu trả lời là cặp có simalirity cao nhất.



Hình 4.2 Biểu đồ so sánh độ chính xác của các mô hình



Hình 4.3 Thống kê số lượng câu hỏi đúng và sai của các mô hình

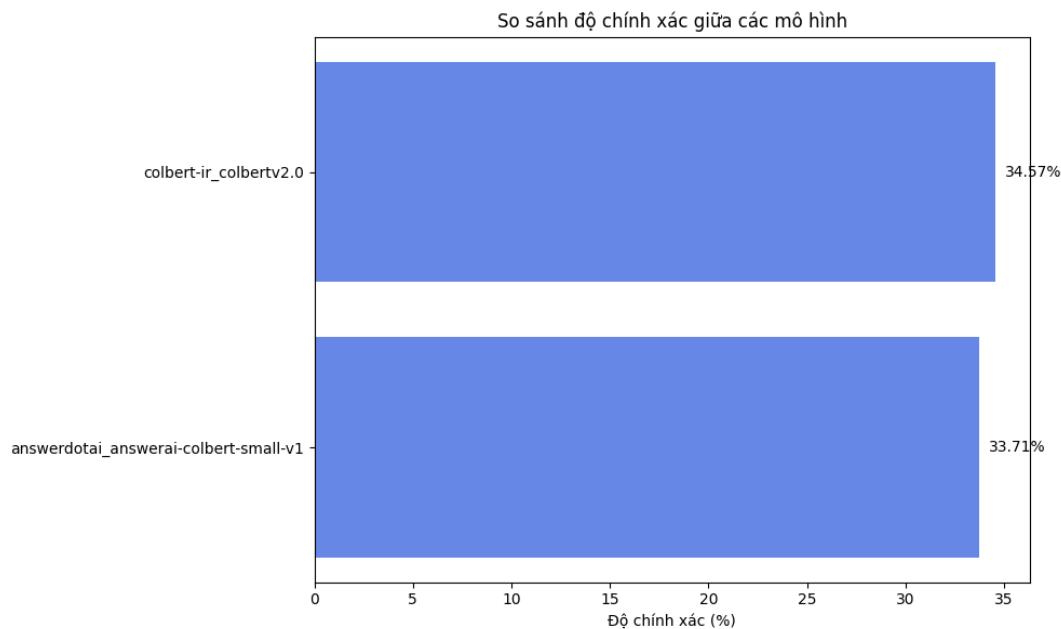
Nhận xét: Hiệu suất của mô hình nhúng không nằm ở độ dài của các vector.

Ví dụ như Alibaba-NLP/gte-multilingual-base và dangvantuan-vietnamese-document-embedding dù chỉ có kích thước là 768 nhưng hiệu suất vẫn cao hơn các mô hình có kích thước 1024. Các mô hình nằm trong top 3 vượt trội hơn hẳn phần còn lại.

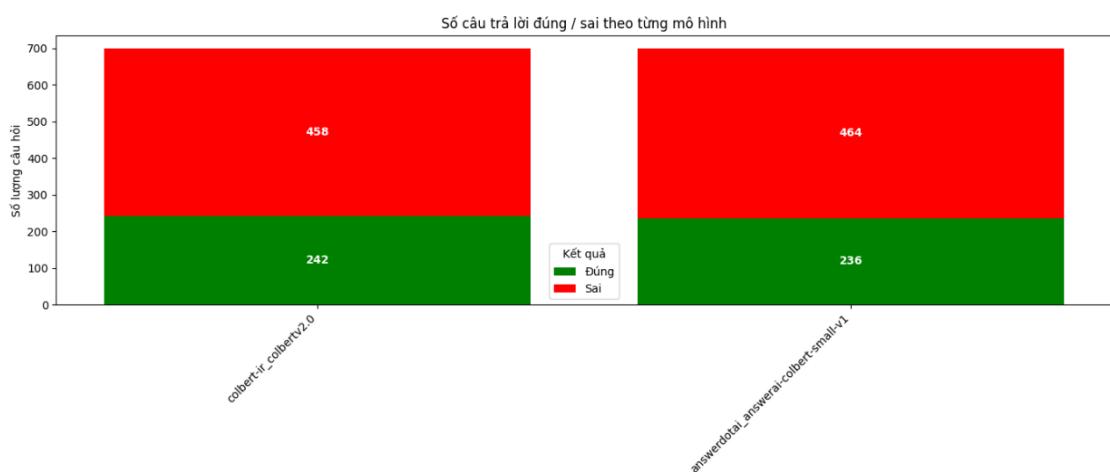
Dựa trên các thống kê và chiến lược truy vấn nhiều giai đoạn trong quadrant. Chúng em sẽ chọn 3 mô hình có 1024, 758 và 512 có độ chính xác cao nhất lần lượt là:

- **intfloat/multilingual-e5-large-instruct.**
- **Alibaba-NLP/gte-multilingual-base.**
- **sentence-transformers/distiluse-base-multilingual-cased-v2.**

Đối với mô hình late interaction chúng em cũng chọn 2 mô hình phổ biến nhất trên Huggingface và so sánh trên tập dữ liệu.



Hình 4.4 Biểu đồ độ chính xác của mô hình rerank



Hình 4.5 Thống kê số lượng câu hỏi đúng và sai của các mô hình

Nhận xét: Sự chênh lệch giữa 2 mô hình không đáng kể chỉ chênh lệch 3.71%.

Dựa trên thống kê chúng em chọn mô hình: **colbert-ir/colbertv2.0**

- **Nhúng**

Sau khi tiền xử lý dữ liệu xong, chúng em sử dụng các model sau để embedding văn bản:

- intfloat/multilingual-e5-large-instruct: 1024 chiều.
- Alibaba-NLP/gte-multilingual-base: 768 chiều.

- sentence-transformers/distiluse-base-multilingual-cased-v2: 512 chiều.
- colbert-ir/colbertv2.0: số lượng vector tùy thuộc vào độ dài văn bản mỗi vector 128 chiều.

4.3.4. Phát triển module tìm kiếm RAG

- **Bước 1: Bắt đầu quá trình truy vấn trong Qdrant**

Mục tiêu là tìm các vector tương tự với các vector đầu vào được cung cấp (các vector đã được nhúng sẵn với các kích thước khác nhau).

Đây là một dạng tìm kiếm tương tự trong không gian vector. Qdrant là một cơ sở dữ liệu vector tối ưu hóa cho việc tìm kiếm nhanh các vector gần giống dựa trên độ tương đồng.

Câu hỏi: “Lịch sử trường đại học Nông Lâm TP. HCM”.

Nhúng:

Vector 1024:

$[-0.0191897, 0.00810107, -0.00485557, -0.02533872, \dots]$

Vector 512: $[-0.04124799, 0.02815446, 0.02005879, 0.00089387, \dots]$

Vector 768: $[0.02712573, 0.04140819, -0.0041654, -0.0103465, \dots]$

Vector late interaction, mỗi vector kích thước là 128:

$[[-0.55281669, 0.48043659, \dots], [0.49336037, -0.23651391, \dots],$
 $[-0.90313280, 0.14040835, \dots], \dots]$

- **Bước 2: Truy vấn phân cấp với Embedding**

Phương thức sử dụng cơ chế "prefetch" lồng nhau (nested prefetch – kỹ thuật truy xuất trước dữ liệu từ nhiều không gian vector khác nhau theo thứ tự phân cấp) để thực hiện truy vấn phân cấp qua nhiều không gian vector với kích thước khác nhau: 512 chiều, 768 chiều, 1024 chiều và late interaction (giai đoạn kết hợp thông tin từ các vector khác nhau để đưa ra kết quả cuối cùng). Đây là một kỹ thuật tối ưu hóa hiệu suất, lấy ý tưởng từ mô hình búp bê "Matryoshka". Truy vấn ở không gian 512 chiều: Đầu tiên, nó tìm kiếm 200 vector tương tự nhất trong không gian 512 chiều. Vì kích thước nhỏ hơn, việc tìm kiếm ở đây nhanh hơn và ít tốn tài nguyên tính toán hơn.

Truy vấn ở không gian 768 chiều: Kết quả từ bước 512 chiều được dùng làm đầu vào để tiếp tục tìm kiếm trong không gian 768 chiều. Lần này, nó giới hạn ở 100 vector. Việc thu hẹp số lượng kết quả giúp giảm tải tính toán ở bước tiếp theo.

Truy vấn ở không gian 1024 chiều: Từ 100 vector ở bước trước, nó tiếp tục lọc ra 50 vector tương tự nhau trong không gian 1024 chiều. Không gian này có độ chi tiết cao hơn, giúp tăng độ chính xác của kết quả.

Sau khi đã lọc được 50 vector từ không gian 1024 chiều, phương thức thực hiện một bước cuối cùng sử dụng vector late interaction. Bước này giới hạn kết quả cuối cùng xuống còn 25 vector tương tự nhau.

"Matryoshka Embedding" là một khái niệm trong học máy, nơi các vector có kích thước lớn hơn (như 1024 chiều) chứa thông tin chi tiết hơn, nhưng các phiên bản nhỏ hơn (512, 768 chiều) vẫn giữ được các đặc điểm chính của dữ liệu. Điều này giống như búp bê Matryoshka của Nga – một búp bê lớn chứa các búp bê nhỏ hơn bên trong. Kỹ thuật này giúp cân bằng giữa tốc độ và độ chính xác trong tìm kiếm vector.

Kết quả truy vấn, gồm 25 tài liệu:

[TRUNG TÂM DỊCH VỤ SINH VIÊN Địa chỉ: Văn phòng Trung tâm Dịch vụ Sinh viên, đường số 6, trường Đại học Nông Lâm TP.HCM Website : <https://ttdvsv.hcmuaf.edu.vn> Điện thoại : 028-38963346 | Email: ttdvsv@hcmuaf.edu.vn', 'Tạp chí Nông nghiệp và phát triển trường Đại học Nông Lâm TP.HCM Điện thoại 028.37245670 Website <https://jad.hcmuaf.edu.vn>',...]

- **Bước 4: Reranking chuyên biệt**

Sử dụng model “nvidia/llama-3.2-nv-rerankqa-1b-v2”, một mô hình được huấn luyện để ranking.

Kết quả sau khi rerank, gồm 25 tài liệu:

[Document(metadata={'relevance_score': 14.7890625}, page_content='Quá trình hình thành và phát triển của trường gắn liền với các tên gọi và mốc thời gian sau: Năm 1955 là Trường Quốc gia Nông Lâm Mục Bảo, đến năm 1963 đổi tên thành Trường Cao đẳng Nông Lâm Súc. Năm 1972, trường trở thành

Học viện Quốc gia Nông nghiệp Sài Gòn. Năm 1975, trường được biết đến với tên gọi Trường Đại học Nông nghiệp 4. Đến năm 1985, tên trường được đổi thành Trường Đại học Nông Lâm nghiệp TP. Hồ Chí Minh. Năm 1995, trường trở thành Trường Đại học Nông Lâm, một thành viên của Đại học Quốc gia TP. Hồ Chí Minh. Năm 2000, trường mang tên Trường Đại học Nông Lâm TP. Hồ Chí Minh'), ...]

- **Bước 5: Trả về kết quả**

Kết quả cuối cùng sẽ lấy các metadata của các vector có *relevance score* > 0 và nối thành chuỗi như một tài liệu hoàn chỉnh:

Tài liệu cuối cùng: Quá trình hình thành và phát triển của trường gắn liền với các tên gọi và mốc thời gian sau: Năm 1955 là Trường Quốc gia Nông Lâm Mục Bảo, đến năm 1963 đổi tên thành Trường Cao đẳng Nông Lâm Súc. Năm 1972, trường trở thành Học viện Quốc gia Nông nghiệp Sài Gòn. Năm 1975, trường được biết đến với tên gọi Trường Đại học Nông nghiệp 4. Đến năm 1985, tên trường được đổi thành Trường Đại học Nông Lâm nghiệp TP. Hồ Chí Minh. Năm 1995, trường trở thành Trường Đại học Nông Lâm, một thành viên của Đại học Quốc gia TP. Hồ Chí Minh. Năm 2000, trường mang tên Trường Đại học Nông Lâm TP. Hồ Chí Minh

Trường Đại học Nông Lâm Thành phố Hồ Chí Minh (Nong Lam University - NLU) là một trường đa ngành, trực thuộc Bộ Giáo dục và Đào tạo, tọa lạc trên khu đất rộng 118 ha, thuộc Thành phố Thủ Đức, Thành phố Hồ Chí Minh và Thành phố Dĩ An - Tỉnh Bình Dương.

- Trường Đại học Nông Lâm TP.HCM tiếp tục xây dựng, phát triển thành một trường đại học có chất lượng về đào tạo, nghiên cứu, chuyển giao công nghệ và hợp tác quốc tế, sánh vai với các trường đại học tiên tiến trong khu vực và trên thế giới. - Đổi mới hệ thống quản lý - quản trị, nhân sự, thúc đẩy tinh thần sáng tạo và khởi nghiệp, phát huy mọi tài năng và các nguồn lực. - Xây dựng môi trường học thuật, nghiên cứu khoa học và phục vụ cộng đồng, trở thành một trong những cơ sở giáo dục đại học hàng đầu Việt Nam. - Xây dựng hệ thống cơ sở vật chất, công nghệ thông tin hiện đại, đáp ứng công

cuộc đổi mới quản lý - quản trị, đào tạo, nghiên cứu khoa học và phục vụ cộng đồng.

Sứ mạng

Trường Đại học Nông Lâm TP.HCM là một trường đại học đa ngành, đào tạo nguồn nhân lực có chuyên môn tốt và tư duy sáng tạo; thực hiện nhiệm vụ nghiên cứu, phát triển, phổ biến, chuyển giao tri thức - công nghệ, đáp ứng nhu cầu phát triển bền vững kinh tế - xã hội của Việt Nam và khu vực.

Tầm nhìn

Đến năm 2035, Trường Đại học Nông Lâm TP.HCM sẽ trở thành trường đại học nghiên cứu với chất lượng quốc tế.

Các đơn vị trong trường Trường Đại học Nông Lâm Tp. Hồ Chí Minh có 12 phòng ban, 07 trung tâm, 01 viện nghiên cứu, 12 khoa đào tạo chuyên môn, 01 khoa cơ bản và 01 bộ môn trực thuộc Trường.

Trường Đại học Nông Lâm TP.HCM công nhận học phần đã tích lũy tại các cơ sở đào tạo có ký kết tương đương, với điều kiện học phần thuộc chương trình đào tạo đã được kiểm định chất lượng. Việc công nhận phải đảm bảo nguyên tắc nghiêm túc, khách quan, phù hợp về nội dung, khối lượng, kỹ năng và đảm bảo công bằng cho sinh viên. Hội đồng Khoa xét công nhận theo học phần hoặc nhóm học phần dựa trên đối sánh chuẩn đầu ra và nội dung đào tạo. Trường Khoa đề xuất danh sách học phần được công nhận, Hiệu trưởng phê duyệt và công bố. Tổng khối lượng được công nhận không vượt quá 50% chương trình đào tạo (trừ ngành sư phạm theo quy định riêng). Sinh viên được ghi điểm M cho các học phần này.

4.4 Xây dựng cơ sở dữ liệu đồ thị Neo4j

Neo4j là một hệ quản trị cơ sở dữ liệu đồ thị mã nguồn mở, được thiết kế để lưu trữ, quản lý và truy vấn dữ liệu dưới dạng các mối quan hệ. Không giống như cơ sở dữ liệu quan hệ truyền thống, Neo4j sử dụng cấu trúc đồ thị bao gồm các nút (nodes) đại diện cho các thực thể và các mối quan hệ (relationships) thể hiện sự liên kết giữa các thực thể đó. Điều này làm cho Neo4j đặc biệt mạnh mẽ trong việc xử

lý các bài toán có mối quan hệ phức tạp, như trong mạng xã hội, hệ thống gợi ý sản phẩm, quản lý chuỗi cung ứng, hay phát hiện gian lận.

4.4.1. Thiết kế mô hình dữ liệu cho Neo4j

Mô hình dữ liệu trong Neo4j đóng vai trò như một bản thiết kế để tổ chức và biểu diễn dữ liệu dưới dạng đồ thị, nó xác định cách các nút và mối quan hệ được cấu trúc, đặt tên và liên kết với nhau.

- Các bước thiết kế mô hình dữ liệu**

- Cấu trúc của đồ thị**

Chúng em tổ chức dữ liệu đồ thị thành các lớp phân tầng, có thể cải thiện đáng kể hiệu suất truy vấn trong Neo4j bằng cách thu hẹp không gian tìm kiếm và giảm số lượng quan hệ cần duyệt. Điều này đặc biệt có lợi trong các đô thị lớn và phức tạp.

Cấu trúc đồ thị phân tầng đóng vai trò như một hình thức phân vùng thông minh trong cơ sở dữ liệu Neo4j. Bằng cách tổ chức dữ liệu một cách logic, nó cho phép module GRAG hướng dẫn quá trình tìm kiếm hiệu quả hơn. Khả năng LLM dự đoán lớp liên quan dựa trên truy vấn của người dùng là rất quan trọng để nhận ra những lợi ích về hiệu suất của cấu trúc này, vì nó cho phép hệ thống bỏ qua các phần không liên quan của đồ thị.

Cấu trúc được chúng em xây dựng thủ công. Tùy theo cấu trúc của từng phần mà được tổ chức khác nhau:

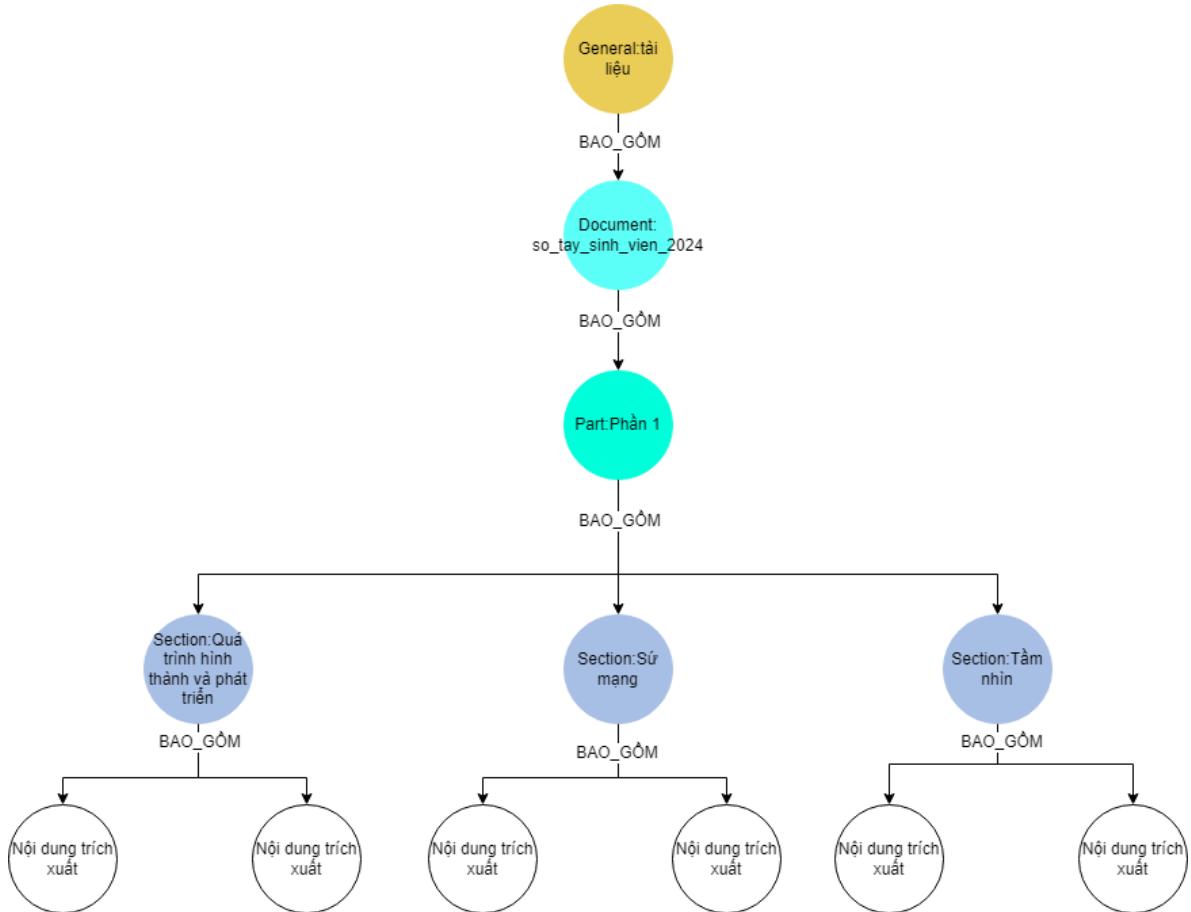
Node cao nhất là đại diện cho toàn bộ hệ thống(có thể là nhiều tài liệu khác nhau)

Node thứ hai là tên của tài liệu(ở đây là so_tay_sinh_vien_2024 được chúng em xây dựng thủ công). Nếu có tài liệu mới thì sẽ lưu UUID.

Từ node thứ hai trở đi là các “phần” bao gồm 3 phần tương ứng trong sổ tay sinh viên.

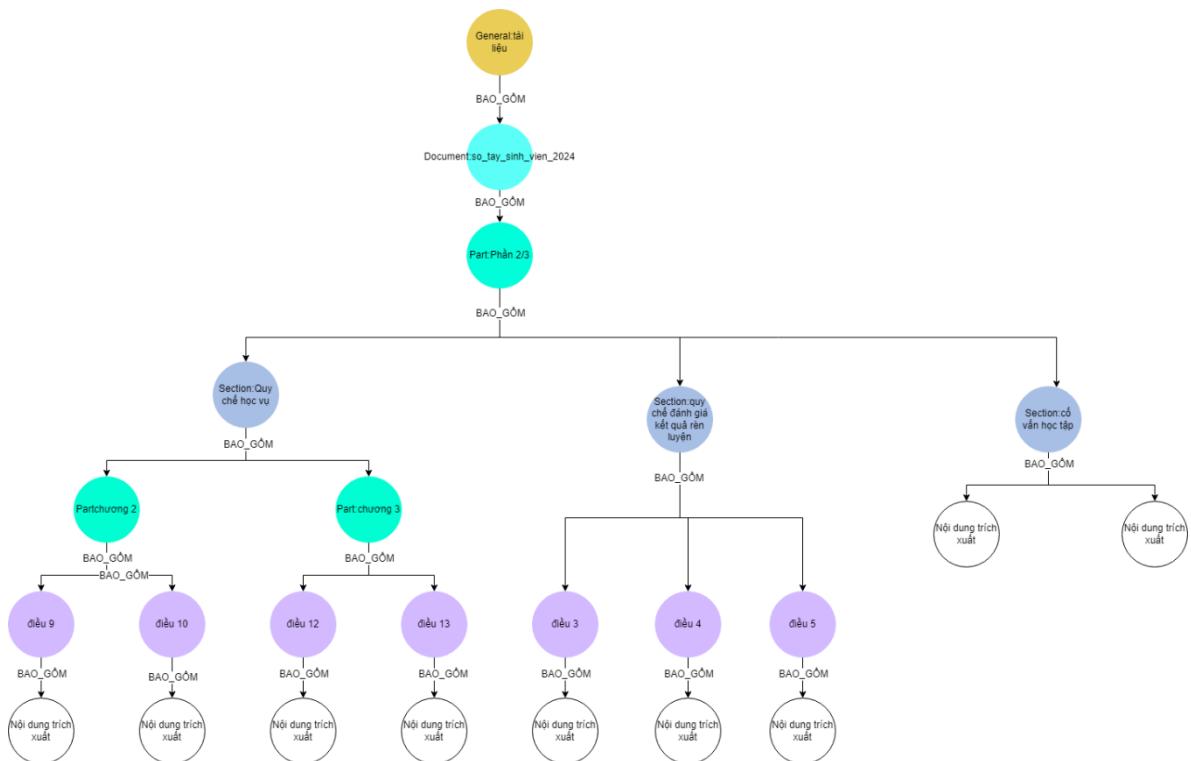
Node ở các tầng tiếp theo phụ thuộc vào cấu trúc ở các phần:

Đối với phần 1, nội dung của mỗi node ở tầng thứ hai chỉ là một đoạn văn cho nên sẽ kết nối trực tiếp đến nội dung tương ứng. Cho nên cấu trúc của phần 1 được biểu diễn như sau:



Hình 4.6 Tổ chức dữ liệu đồ thị của phần 1

Đối phần 2 và 3 nội dung của mỗi node ở tầng hai được chia nhỏ ra thành mỗi “**chương**”, mỗi chương gồm các “**điều**”, mỗi điều gồm “**thứ tự 1, 2, 3...**” Cho nên cấu trúc của phần 2, 3 được biểu diễn như sau:



Hình 4.7 Tô chức dữ liệu đồ thị của phần 2 và 3

Các node và relationship ở các phần “Nội dung trích xuất” được thiết kế như sau:

B1: Lấy từng đoạn nội dung tương ứng.

B2: Đưa từng nội dung vào LLM để trích xuất node và relationship, có dạng như sau:

Câu hỏi: "Trường Đại học Nông Lâm Thành phố Hồ Chí Minh (Nong Lam University - NLU) là một trường đa ngành, trực thuộc Bộ Giáo dục và Đào tạo, tọa lạc trên khu đất rộng 118 ha, thuộc Thành phố Thủ Đức, Thành phố Hồ Chí Minh và Thành phố Dĩ An - Tỉnh Bình Dương."

Trích xuất:

```
{
  "relationships": [
    {
      "source": {
        "name": "Trường Đại học Nông Lâm Thành phố Hồ Chí Minh",
        "english_name": "Nong Lam University",
      }
    }
  ]
}
```

```
        "acronym": "NLU",
        "school_type": "đa ngành",
        "campus_size": "118 ha"
    },
    "target": {
        "name": "Bộ Giáo dục và Đào tạo",
        "ministry_level": "Bộ",
        "focus_area": "Giáo dục và Đào tạo"
    },
    "type_source": "organization",
    "type_target": "government_organization",
    "relation": "trực_thuộc"
},
{
    "source": {
        "name": "Trường Đại học Nông Lâm Thành phố Hồ Chí Minh",
        "english_name": "Nong Lam University",
        "acronym": "NLU",
        "school_type": "đa ngành",
        "campus_size": "118 ha"
    },
    "target": {
        "name": "Thành phố Thủ Đức",
        "admin_level": "Thành phố",
        "subordinate_to_city": "Thành phố Hồ Chí Minh"
    }...
]
}
```

- **Xử lý ngoại lệ**
 - **Node và relation lặp lại**

Trong quá trình trích xuất node và relationship tự động của LLM. Các vấn đề gặp phải là:

Nodes, types và relationship có ý nghĩa giống nhau nhưng mặt chữ lại khác nhau:

Ví dụ: “trường đại học nông lâm tp. hcm” và “trường đại học nông lâm tp. hồ chí minh”.

Điều này làm tăng kích thước của đồ thị, đặc biệt là sự trùng lặp về relationship.

Nếu sự trùng lặp này xảy ra đồng thời ở node và relationship thì kích thước đồ thị tăng rất nhiều làm cho khó bảo trì, thời gian và hiệu suất truy xuất.

Ví dụ:

“trường đại học nông lâm tp. hcm” –tọa_lạc_tại→ “phường linh trung”

“trường đại học nông lâm tp. hcm” –địa_chỉ → “phường linh trung”

“trường đại học nông lâm tp. hồ chí minh” –tọa_lạc_tại→ “phường linh trung”

“trường đại học nông lâm tp. hồ chí minh” –địa_chỉ → “phường linh trung”

Để giải quyết vấn đề này sau khi đã thêm vào đồ thị. Chúng em thực hiện thủ công bằng cách lấy toàn bộ node, type, relationships đưa vào LLM để lọc ra các đối tượng giống nhau và thống nhất sử dụng một đối tượng. Nếu dư thừa có thể xóa bỏ.

Ví dụ:

“department” và “department_subject” thành department

“group” và “group_of_people” thành group

“email” và “email_address” thành email.

Sau đó sử dụng cypher để cập nhật để thống nhất.

4.4.2. Phát triển module tìm kiếm GRAG

- **Quy trình truy vấn:**

- **Đầu vào:** Người dùng nhập câu hỏi của họ bằng ngôn ngữ tự nhiên.

Câu hỏi: “Năm 2000 đại học Nông Lâm Tp.HCM tên là gì”

- **Dự đoán ngữ cảnh:** Đưa câu hỏi kèm theo prompt đã được thiết kế vào LLM(đầu ra là một câu truy vấn Cypher hoàn chỉnh).

LLM dự đoán và trả về câu cypher:

“MATCH

```
p=(document:Document {name: 'so_tay_sinh_vien_2024'})-  
[*]->(predict:Section {name: 'quá trình hình thành và phát triển'})-  
[r*1..2]->(e) RETURN p”
```

- **Truy xuất kết quả:** Cơ sở dữ liệu Neo4j xử lý truy vấn.

Kết quả của các nodes:

```
{'4:d931436f-22b8-460d-a3fd-ae6dab9ca2ef:2440': {'properties':  
  {'name': 'so_tay_sinh_vien_2024'}}, '4:d931436f-22b8-460d-a3fd-  
  ae6dab9ca2ef:0': {'properties': {'name': 'phản 1: nlu - định hướng  
  trường đại học nghiên cứu'}}, '4:d931436f-22b8-460d-a3fd-  
  ae6dab9ca2ef:1': {'properties': {'name': 'quá trình hình thành và phát  
  triển'}}},...}
```

Kết quả các quan hệ:

```
[{'source': '4:d931436f-22b8-460d-a3fd-ae6dab9ca2ef:2440', 'target':  
  '4:d931436f-22b8-460d-a3fd-ae6dab9ca2ef:0', 'type': 'BAO_GÒM',  
  'properties': {}}, {'source': '4:d931436f-22b8-460d-a3fd-  
  ae6dab9ca2ef:0', 'target': '4:d931436f-22b8-460d-a3fd-  
  ae6dab9ca2ef:1', 'type': 'BAO_GÒM', 'properties': {}}, ...]
```

- **Rerank lại kết quả:** Bước này cần thiết để tập trung vào chuỗi có ý nghĩa và lấy ra top 20 có độ tương đồng cosine cao nhất.

Kết quả:

```
['chính_thức_mang_tên_năm_2000_campus_area_118_ha_name  
Trường Đại học Nông Lâm Thành phố Hồ Chí Minh location Thành  
phố Hồ Chí Minh standard_name Trường Đại học Nông Lâm  
TP.HCM abbreviation NLU type organization english_name Nong  
Lam University',
```

'name Trường Quốc gia Nông Lâm Mục Blao thành_lập_năm 1955
status tên_cũ',

'name Trường Đại học Nông Lâm standard_name Trường Đại học
Nông Lâm TP.HCM đổi_tên_năm 1995 status tên_cũ',

'name Trường Đại học Nông Lâm nghiệp TP. Hồ Chí Minh
đổi_tên_năm 1985 status tên_cũ',

'name 1955 description năm thành lập',

'province Tỉnh Bình Dương name Thành phố Dĩ An',

'name Trường Đại học Nông nghiệp số 4 đổi_tên_năm 1975 status
tên_cũ',

'name Trường Cao đẳng Nông Lâm Súc đổi_tên_năm 1963 status
tên_cũ',

'name Huân chương Lao động Hạng ba',

'name Học viện Quốc gia Nông nghiệp Sài Gòn đổi_tên_năm 1972
status tên_cũ']

- **Đầu ra cuối cùng:** Nối chuỗi để đưa vào LLM.

Xử lý kết quả thành các chuỗi có ý nghĩa:

“name Đại học Quốc gia TP.HCM name Trường Đại học Nông
nghiệp số 4 đổi_tên_năm 1975 status tên_cũ name Trường Đại học
Nông Lâm nghiệp TP. Hồ Chí Minh đổi_tên_năm 1985 status tên_cũ
name Trường Cao đẳng Nông Lâm Súc đổi_tên_năm 1963 status
tên_cũ name Trường Quốc gia Nông Lâm Mục Blao thành_lập_năm
1955 status tên_cũ name Học viện Quốc gia Nông nghiệp Sài Gòn
đổi_tên_năm 1972 status tên_cũ name Huân chương Độc lập Hạng
ba **chính_thức_mang_tên_năm 2000 campus_area 118 ha name**
Trường Đại học Nông Lâm Thành phố Hồ Chí Minh location
Thành phố Hồ Chí Minh standard_name Trường Đại học Nông Lâm
TP.HCM abbreviation NLU type organization english_name Nong
Lam University country Việt Nam role Cơ quan ban hành văn bản
acronym MOET name Bộ Giáo dục và Đào tạo name Trường Đại học

Nông Lâm standard_name Trường Đại học Nông Lâm TP.HCM
đổi_tên_năm 1995 status tên_cũ”

• **Lợi ích Tiềm năng:**

- **Tăng cường độ chính xác tìm kiếm:** Tận dụng sức mạnh hiểu ngôn ngữ tự nhiên của LLM và khả năng truy vấn chính xác của Cypher trên một đồ thị có cấu trúc dẫn đến kết quả tìm kiếm chính xác và liên quan hơn so với tìm kiếm dựa trên từ khóa truyền thống hoặc tìm kiếm tương tự vector đơn giản [21].
- **Cải thiện hiệu suất truy vấn:** Cấu trúc đồ thị phân tầng được sử dụng hiệu quả có thể giảm đáng kể không gian tìm kiếm và dẫn đến thời gian thực hiện truy vấn nhanh hơn, đặc biệt là trong các tập dữ liệu lớn.
- **Khám phá các kết nối ẩn:** Cấu trúc đồ thị của Neo4j vốn cho phép khám phá các mối quan hệ và kết nối giữa các điểm dữ liệu mà có thể không dễ nhận thấy trong dữ liệu dạng bảng hoặc tài liệu, dẫn đến những hiểu biết sâu sắc hơn.

• **Thách thức Tiềm năng:**

- **Độ phức tạp của tích hợp LLM:** Việc tích hợp LLM để dự đoán một cách hiệu quả để đạt được ngữ cảnh và tạo truy vấn Cypher chính xác có thể phức tạp và đòi hỏi thử nghiệm và tinh chỉnh đáng kể [22].
- **Sai sót Tiềm năng:** LLM đôi khi đưa ra các dự đoán không chính xác về ngữ cảnh đồ thị liên quan hoặc tạo ra các truy vấn Cypher sai cú pháp hoặc không phản ánh chính xác ý định của người dùng có thể gây ra lỗi, tốn thời gian tìm kiếm, do đó chúng em đã thiết kế thêm một LLM để xác thực lại câu truy vấn Cypher đảm bảo cơ chế xác thực và xử lý lỗi mạnh mẽ [22].

4.5 Xây dựng module phê bình(Critical Module)

Chúng em nhận thấy rằng LLM thường gặp khó khăn trong việc trả lời chính xác đối với các truy vấn đòi hỏi nhiều bước suy luận hoặc kết hợp thông tin từ các nguồn khác nhau ngay từ lần đầu tiên. Nguyên nhân có thể xuất phát từ sự không

chắc chắn trong việc lựa chọn nguồn tri thức phù hợp và những nhầm lẫn trong quá trình thao tác với các module truy xuất thông tin. Module phê bình được tạo ra để giải quyết trực tiếp vấn đề này bằng cách cung cấp cơ chế phản hồi, giúp tác nhân điều chỉnh hành động của mình [2].

- **Chức năng cốt lõi của module phê bình bao gồm hai nhiệm vụ chính:**

Thẩm định kết quả đầu ra của Agent và cung cấp phản hồi mang tính xây dựng để Agent có thể cải thiện trong các bước tiếp theo. Cụ thể, module này giúp xác định và sửa chữa các lỗi trong hành động của Agent, chẳng hạn như việc lựa chọn không đúng module truy xuất hoặc cung cấp đầu vào không chính xác cho module truy xuất [2].

Một đặc điểm nổi bật trong kiến trúc của module phê bình là cách tiếp cận "chia để trị", tách biệt quá trình thẩm định kết quả đầu ra với quá trình phản hồi sửa lỗi thành hai thành phần riêng biệt. Cách tiếp cận này bao gồm hai thành phần chính: **Validator** và **Commentor** [2].

Validator: Quyết định xem câu trả lời có chính xác và toàn vẹn với câu hỏi hay không. Về cơ bản, đây là một nhiệm vụ phân loại nhị phân (đúng hoặc sai). Để nâng cao độ chính xác của quá trình thẩm định này, chúng em cung cấp thêm một ngữ cảnh thẩm định cho validator. Ngữ cảnh này có thể được sử dụng để xác minh xem đầu ra có đáp ứng các yêu cầu cụ thể được đề cập trong câu hỏi hay không từ đó quyết định hệ thống nên dừng lại hay tiếp tục vòng lặp để cải thiện câu trả lời [2].

Câu hỏi: “Địa chỉ trường đại học nông lâm ở đâu”

Trả lời: “Khu phố 6, Phường Linh Trung, Quận Thủ Đức, Thành phố Hồ Chí Minh”

Cho kết quả là: “yes”.

Commentor: Khi một câu trả lời chưa đáp ứng các tiêu chí của câu hỏi, Commentor sẽ phản hồi sửa lỗi về các thực thể chủ đề và quan hệ hữu ích mà Agent tạo ra trước đó từ đó cải thiện hành động của Agent cho vòng lặp kế tiếp. Để hướng dẫn Agent một cách hiệu quả, Commentor tạo ra phản hồi mang tính sửa lỗi, dễ hiểu đối với Agent. Đồng thời nó cũng đóng góp vai trò quan trọng cho Agent tự phản ánh có mục đích chứ không phải ngẫu

nhiên, làm cho điểm hội tụ xảy ra nhanh hơn. Để tránh việc phải tinh chỉnh mô hình, Commentor tận dụng khả năng học trong ngữ cảnh (ICL) để cung cấp phản hồi phức tạp. Điều này được thực hiện bằng cách sử dụng các ví dụ đã được thu thập trước về các trải nghiệm thành công, trong đó mỗi ví dụ bao gồm một cặp hành động và phản hồi đã được xác minh dựa trên dữ liệu thực tế. Trong quá trình suy luận, Commentor sử dụng các ví dụ này để tạo ra phản hồi chất lượng cao [2].

Phản hồi: “1. Tài liệu khả thi từ nguồn GRAPH cung cấp thông tin về nhiều địa điểm liên quan đến Trường Đại học Nông Lâm TP.HCM, bao gồm cả cơ sở chính và các phân hiệu tại Ninh Thuận và Gia Lai. Tuy nhiên, câu hỏi yêu cầu địa chỉ cụ thể của trường (cơ sở chính). Tài liệu cung cấp địa chỉ của phân hiệu tại Gia Lai (Đường Trần Nhật Duật, Thông 01, Xã Diên Phú, Thành phố Pleiku, Tỉnh Gia Lai) và tên đường của cơ sở chính (Đường Trần Nhật Duật), nhưng thiếu só nhà và thông tin chi tiết hơn về địa chỉ cơ sở chính tại TP.HCM. Do đó, thực thể địa chỉ cụ thể của cơ sở chính Trường Đại học Nông Lâm TP.HCM bị thiếu.

2. Đề xuất chuyển sang nguồn truy xuất 'TEXT' để tìm kiếm thông tin địa chỉ cụ thể của trường Đại học Nông Lâm TP.HCM.”

- **Cách tiếp cận chia để trị này mang lại hai ưu điểm chính:** Bằng cách chia một nhiệm vụ phức tạp thành hai nhiệm vụ đơn giản hơn, các LLM đã được huấn luyện trước có thể được sử dụng hiệu quả cho cả việc thẩm định và tạo phản hồi mà không đòi hỏi nguồn lực đáng kể cho việc tinh chỉnh.

Sự tách biệt này cho phép Validator và Commentor có ngữ cảnh riêng biệt, ngăn chặn việc đưa vào các thông tin không liên quan có thể làm xao nhãng chúng khỏi các nhiệm vụ cụ thể của mình [2].

4.6 Đánh giá tập dữ liệu ZaloAI

VMLU là một bộ tập dữ liệu được thiết kế để đánh giá LLM dành cho ngôn ngữ tiếng việt, gồm 10880 trắc nghiệm thuộc 58 môn học khác nhau, được phân bổ trong 4 lĩnh vực chính: STEM, Nhân văn, Khoa học xã hội và các lĩnh vực mở rộng khác [23].

Gồm 4 bộ:

- Vi-MQA: Gồm 10.880 câu hỏi trắc nghiệm, trải đều trên 58 chủ đề khác nhau để đánh giá hiểu biết cơ bản lần chuyên sâu [24].

Cấu trúc:

```
{  
  "id": "28-0001",  
  "question": "Nếu GDP bình quân thực tế của năm 2000 là 18,073$ và GDP bình quân thực tế của năm 2001 là 18,635$ thì tỷ lệ tăng trưởng của sản lượng thực tế trong thời kỳ này là bao nhiêu?",  
  "choices": [  
    "A. 3.0%",  
    "B. 3.1%",  
    "C. 5.62%",  
    "D. 18.0%",  
    "E. 18.6%"  
  ],  
  "answer": "B"  
}
```

- ViDialog: Gồm 210 đoạn hội thoại đa lượt (multi-turn dialogues). Đánh giá khả năng mô hình hiểu và duy trì ngữ cảnh hội thoại, sự liên kết câu, và sự mạch lạc trong đối thoại [24].

Cấu trúc:

```
{  
  "id": "dialog-000",  
  "queries": [  
    "Mình hiện đang dùng iPhone 14 Pro Max. Giới thiệu game cho mình nhé.",  
    "Mình vừa bị mất cáp sạc điện thoại di động. Mình nên mua loại cáp sạc nào?",  
    "Dung lượng pin điện thoại của mình là bao nhiêu?"  
  ]
```

}

- Vi-Drop: Gồm 3.090 câu hỏi chia theo sáu loại lý luận khác nhau [24].

Cáu trúc:

{

question_id": 0,

"category": "count",

"context": "\n\n Thành lập xã Ia Khai (Ia Grai) trên cơ sở một phần xã Ia Krái. Xã Ia Khai có 16.518,5 ha diện tích tự nhiên và 2.475 nhân khẩu.\n\n Thành lập xã Ia Nhìn (Chư Păh) trên cơ sở một phần xã Ia Ka. Xã Ia Nhìn có 3.205 ha diện tích tự nhiên và 3.758 nhân khẩu.\n\n Điều chỉnh địa giới hành chính 2 xã Ia Mơ Nông và xã Ia Phí (Chư Păh). Thành lập xã Ia Ly (Chư Păh) trên cơ sở một phần xã Ia Mơ Nông. Xã Ia Ly có 4.844 ha diện tích tự nhiên và 4.570 nhân khẩu. Xã Ia Mơ Nông có 16.951 ha diện tích tự nhiên và 4.071 nhân khẩu. Xã Ia Phí có 6.995 ha diện tích tự nhiên và 5.172 nhân khẩu.",

"question": "Có bao nhiêu xã được thành lập?"

}

- Vi-SquAD: Bao gồm 3.310 câu hỏi được xây dựng dựa trên văn bản tiếng Việt. Đánh giá khả năng đọc hiểu và trích xuất thông tin từ đoạn văn bản [24].

Cáu trúc:

{

"id": 0,

"question": "Công ty mẹ của Fenway Sports Group được gọi là gì ban đầu?",
"context": "Tập đoàn Thể thao Fenway(tiếng Anh: Fenway Sports Group ; viết tắt: FSG) là một công ty đầu tư thể thao của Mĩ. Đây là công ty mẹ của câu lạc bộ bóng chày Mĩ Boston Red Sox và câu lạc bộ bóng đá Anh Liverpool. Fenway Sports Group được thành lập vào năm 2001 (tên ban đầu là New England Sports Ventures, ghi tắt NESV) khi John W.Henry gia nhập lực lượng cùng Tom Werner, Les Otten, The New York Times Company

(công ty này sở hữu 16,58%) và các nhà đầu tư khác để trở thành nhà thầu cho Red Sox."

}

Dựa trên các thông tin về từng bộ dữ liệu chúng em đã nhận thấy tập VMLU không phù hợp để đánh giá. Hệ thống của chúng em là một hệ thống tìm kiếm thông tin dựa trên tài liệu sẵn có và có ý nghĩa liên quan tới nhau từ đầu đến cuối. Bộ có thể xem xét đánh giá là Vi-SquAD và Vi-Drop nhưng các đoạn thông tin là rời rạc và không có ý nghĩa với nhau.

4.7 Đánh giá tập dữ liệu StaRK-Prime

StaRK-Prime là một phần của bộ benchmark STARK để đánh giá khả năng truy xuất của mô hình trên SKB trong lĩnh vực y sinh. Kết hợp giữa KB và cơ sở dữ liệu vector [25].

PrimeKG là một KB gồm 10 kiểu thực thể(Disease, Drug, Gene/Protein, Pathway, Anatomy, Phenotype, Exposure, Biological process, Molecular function, Cellular component) và 18 kiểu quan hệ(ppi, carrier, enzyme, target, transporter, contraindication, indication, off-label use, synergistic interaction, associated with, parent-child, phenotype absent, phenotype present, side effect, interacts with, linked to, expression present, expression absent), gần 129000 thực thể và gần 8 triệu quan hệ [25].

StarRK-PRIME gồm 11204 truy vấn có cả single-hop(lý luận đơn lẻ) và multi-hop(lý luận nhiều bước) [25].

Kết quả đánh giá:

Hit@1: 0.140759

Hit@5: 0.177479

Recall@20: 0.142282

MRR: 0.157059

4.8 Đánh giá và so sánh các kiến trúc khác nhau

4.8.1. Tập dữ liệu

Chúng em sử dụng mô hình tạo sinh ngôn ngữ kèm prompt phù hợp để tạo ra các câu hỏi kết hợp và câu trả lời dựa trên số tay sinh viên năm 2024.

Chúng em sử dụng từng đoạn nội dung có liên quan đến nhau được lọc thủ công và đưa vào mô hình tạo sinh ngôn ngữ để có câu hỏi và trả lời chất lượng nhất, có đầy đủ ngữ cảnh và đủ khó để đánh giá một cách khách quan.

Tập dữ liệu gồm 2 cột question và answer, có tổng cộng 543 cặp câu hỏi và trả lời.

4.8.2. Đánh giá

Chúng em sẽ chia ra ba kiến trúc đánh giá nhằm khẳng định sự quan trọng của Agent trong kiến trúc. Các đánh giá sẽ được lưu vào file .csv với các cột question và answer.

Để đánh giá độ tương đồng chúng em sử dụng các model nằm trong top 3 ở hình 4.2 để embedding câu hỏi dự đoán và câu trả lời thực sau đó tính độ tương đồng trung bình cho từng model sau đó tiếp tục trung bình của 3 model.

Ký hiệu:

dangvantuan: dangvantuan_vietnamese_document-embedding

infloate: intfloat_multilingual-e5-large-instruct

alibaba: Alibaba-NLP_gte-multilingual-base

- So sánh độ tương đồng trung bình**

Kiến trúc/ Mô hình	dangvantuan	infloate	alibaba
Toàn bộ kiến trúc	0.826	0.950	0.869
Chỉ sử dụng RAG	0.756	0.936	0.824
Chỉ sử dụng GRAG	0.412	0.862	0.613

Bảng 4.1 So sánh độ tương đồng trung bình của ba kiến trúc

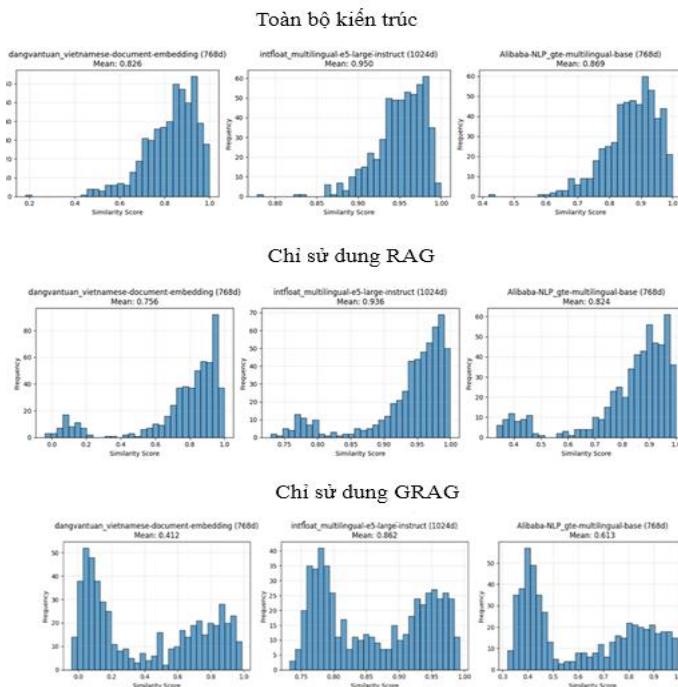
Nhận xét:

- **Toàn bộ kiến trúc:** Điểm trung bình giữa ba mô hình không cho quá chênh lệch và đều trên 0.8(thấp nhất là 0.826) điều này cho thấy khả năng trả lời câu hỏi ổn định, chính xác và không gặp ảo giác khi sử dụng Agent.
- **Chỉ sử dụng RAG:** Điểm trung bình giữa ba mô hình có sự chênh lệch hơn so với toàn bộ kiến trúc. Tuy mô hình infloate và alibaba cho điểm trung bình gần bằng nhau($0.95 \approx 0.936$ và $0.896 \approx 0.824$) nhưng mô hình còn lại bị lệch rất nhiều($0.826 > 0.756$). Cho thấy

một phần của sự không ổn định bắt đầu xuất hiện có thể là do không có thông tin để trả lời câu hỏi hoặc có câu trả lời nhưng LLM không nhận diện ra ngay lần đầu tiên hoặc bị ảo giác(khả năng thấp).

- **Chỉ sử dụng GRAG:** Ở kiến trúc này là tệ nhất khi điểm trung bình chênh lệch quá lớn giữa 3 mô hình của nó và với các mô hình tương ứng của hai kiến trúc khác. Cho thấy hiệu suất kém, không ổn định khi sử dụng kiến trúc này mà không kèm thêm Agent.

- **So sánh phân phối độ tương đồng dạng histogram**

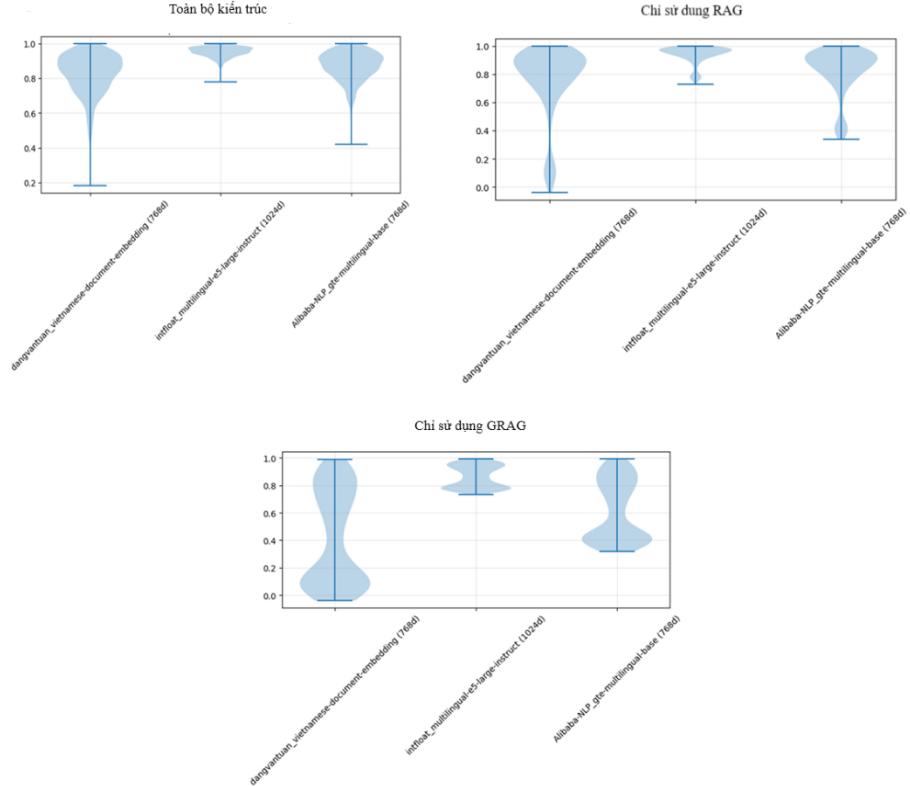


Hình 4.8 Biểu đồ histogram thống kê phân phối độ tương đồng

Nhận xét:

- **Toàn bộ kiến trúc:** Hầu hết phân phối về bên phải(thậm chí có câu hỏi bằng 1) trong khi bên trái thì rất ít chỉ tầm 1 đến 5 câu thường thuộc trường hợp không trả lời được hoặc bị ảo giác.
- **Chỉ sử dụng RAG:** Hầu hết phân phối về bên phải nhưng bên trái bắt đầu xuất hiện các câu hỏi không chính xác, điều này đúng khi đã biểu hiện ở bảng 4.1.

- **Chỉ sử dụng GRAG:** Số lượng phân phối về bên trái đột nhiên tăng bất thường ở kiến trúc này, cho thấy kiến trúc này dần mất đi khả năng trả lời câu hỏi rõ rệt hơn.
- **So sánh phân phối độ tương đồng dạng violin**



Hình 4.9 Biểu đồ violin thống kê phân phối chi tiết độ tương đồng

Nhận xét:

- **Toàn bộ kiến trúc:** Cho thấy rõ hơn hiệu suất khi sử dụng Agent. Biểu đồ không bị lưỡng cực ở hai đầu cho thấy sự ổn định của kiến trúc. Tuy mô hình đầu tiên đột nhiên có điểm thấp nhất là dưới 0.2 nhưng số lượng ít và không đáng kể phản ánh đúng sự đồng nhất với biểu đồ ở bảng 4.1 và hình 4.6.
- **Chỉ sử dụng RAG:** Sự lưỡng cực bắt đầu xuất hiện nhưng chưa rõ ràng.
- **Chỉ sử dụng GRAG:** Sự lưỡng cực xuất hiện rõ ràng và tệ hơn cho thấy khi sử dụng GRAG hoặc RAG mà không có Agent thì hiệu suất giảm đáng kể.

- **Thống kê phân phối Accuracy, Hallucination và Missing**

Thống kê này em tạo ra bằng cách sử dụng mô hình gemini-2.5-flash để dự đoán giữa hai câu trả lời dự đoán và thực tế.

- 1 là accuracy
- 2 là missing
- 3 là hallucination

Metric	Toàn bộ kiến trúc	Chỉ sử dụng RAG	Chỉ sử dụng GRAG
Accuracy	95.20%	84.32%	28.41%
Hallucination	1.48%	1.11%	17.16%
Missing	3.32%	14.58%	54.43%

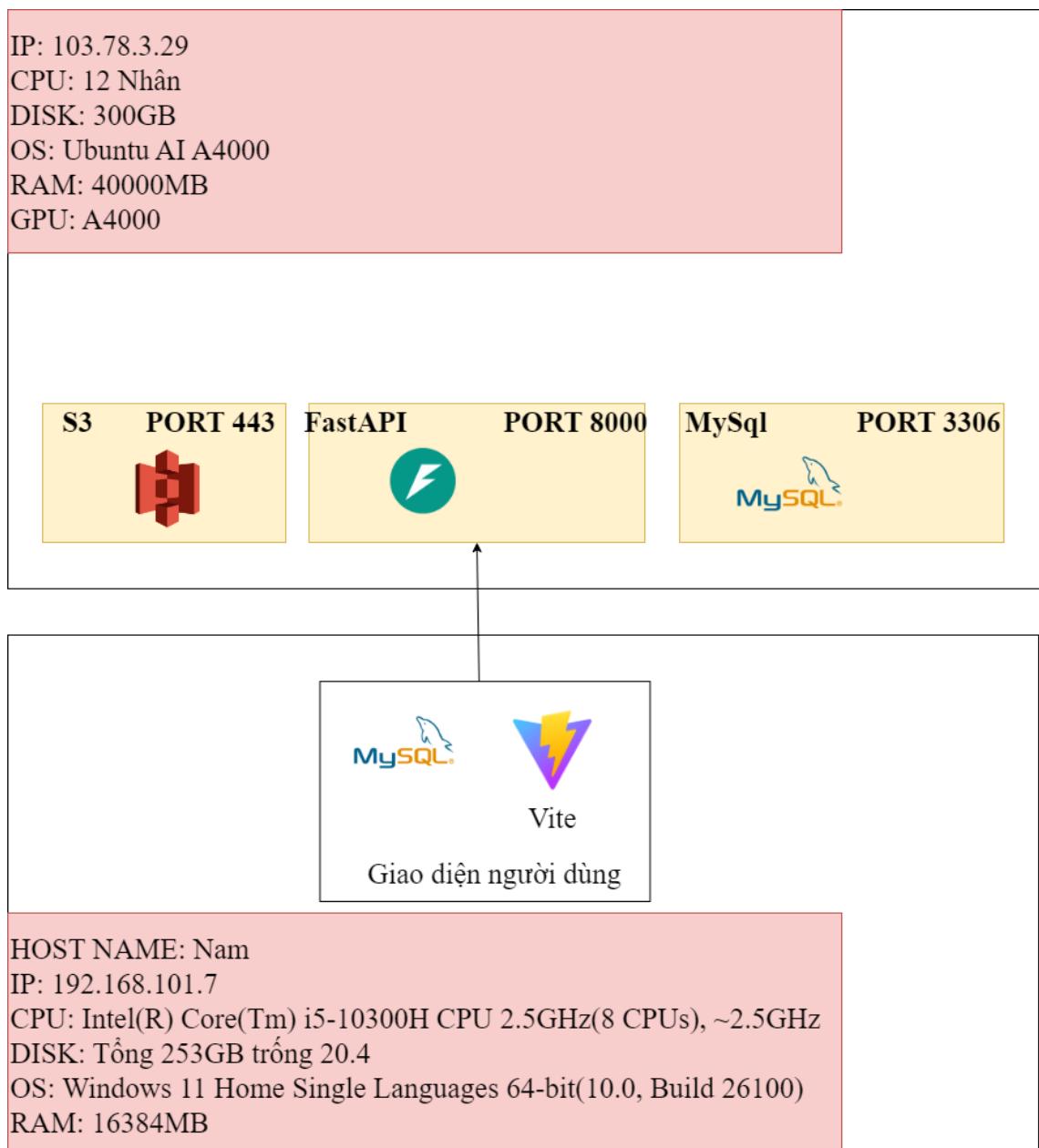
Bảng 4.2 Bảng thống kê Accuracy, Halluc và Missing

Nhận xét:

- **Toàn bộ kiến trúc:** Vì sử dụng prompt thích hợp nên việc thống kê hallucination rất thấp(kể cả 2 kiến trúc còn lại). tuy nhiên missing sẽ tăng cao do khi không có dữ liệu thì mô hình sẽ trả lời “không có thông tin”.
- **Chỉ sử dụng RAG:** Accuracy giảm nhẹ do missing đã tăng lên khá cao so với toàn bộ kiến trúc. Lý do là chỉ truy vấn với văn bản phẳng thì khó có đủ thông tin và trả lời đúng ngay từ lần đầu tiên.
- **Chỉ sử dụng GRAG:** kiến trúc này rất tệ đã được thể hiện qua nhiều biểu đồ khác nhau ở trên. Bảng này chỉ làm rõ lên sự yếu kém đó.

4.9 Xây dựng ứng dụng

4.9.1. Tổng quan ứng dụng



Hình 4.10 Kiến trúc ứng dụng

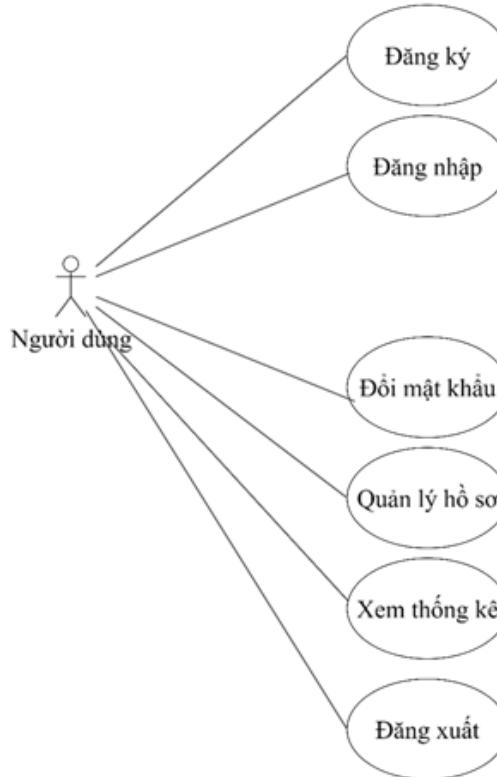
4.9.2. Thiết kế cơ sở dữ liệu

- Người dùng và kiến thức:** Một người dùng có thể sở hữu nhiều tài liệu kiến thức.
- Người dùng và hội thoại:** Một người dùng có thể sở hữu nhiều hội thoại.
- Người dùng và tin nhắn:** Một người dùng có thể sở hữu nhiều tin nhắn.

- 4) **Hội thoại và kiến thức:** Mỗi hội thoại chỉ trả lời một tài liệu kiến thức.
- 5) **Hội thoại và bot:** Mỗi hội thoại sở hữu một bot.
- 6) **Hội thoại và tin nhắn:** Mỗi hội thoại sở hữu nhiều tin nhắn.

4.9.3. Các chức năng chính của ứng dụng

Quản lý người dùng



Hình 4.11 Lược đồ Usecase cho chức năng Quản lý người dùng

Mô tả lược đồ Usecase

Tên Use Case	Đăng ký
Mô tả Use Case	Người dùng tạo tài khoản mới để sử dụng hệ thống chatbot tư vấn học vụ.
Tác nhân	Người dùng
Mức độ ưu tiên	Cao
Điều kiện kích hoạt	Người dùng truy cập trang đăng ký và nhập đầy đủ thông tin bắt buộc

Điều kiện tiền đề	<ul style="list-style-type: none"> - Hệ thống hoạt động bình thường - Người dùng chưa có tài khoản - Kết nối internet ổn định - Email chưa được sử dụng
Ràng buộc	<ul style="list-style-type: none"> - Email phải duy nhất trong hệ thống - Mật khẩu: tối thiểu 8 ký tự, có chữ hoa, thường, số - Tên người dùng: 3-50 ký tự - Thời gian phản hồi < 3 giây
Luồng	<p>B1. Người dùng truy cập trang đăng ký.</p> <p>B2. Người dùng nhập các thông tin bắt buộc như tên (3-50 ký tự), email và mật khẩu (tối thiểu 8 ký tự, có chữ hoa, chữ thường và số).</p> <p>B3. Hệ thống kiểm tra tính hợp lệ của thông tin, đặc biệt là email phải là duy nhất trong hệ thống.</p> <p>Nếu thành công: Một tài khoản mới được tạo, hệ thống gửi email xác nhận và chuyển người dùng đến trang đăng nhập. Thời gian phản hồi phải dưới 3 giây.</p> <p>Nếu thất bại: Hệ thống hiển thị thông báo lỗi cụ thể (ví dụ: "Email đã tồn tại").</p>

Bảng 4.3 Mô tả Usecase cho chức năng đăng ký tài khoản

Tên Use Case	Đăng nhập
Mô tả Use Case	Người dùng xác thực danh tính để truy cập vào hệ thống chatbot.
Tác nhân	Người dùng
Mức độ ưu tiên	Cao

Điều kiện kích hoạt	Người dùng nhập email/username và mật khẩu, nhấn đăng nhập
Điều kiện tiền đề	<ul style="list-style-type: none"> - Người dùng có tài khoản hợp lệ - Tài khoản chưa bị khóa - Hệ thống xác thực hoạt động - Thông tin đăng nhập chính xác
Ràng buộc	<ul style="list-style-type: none"> - Tối đa 5 lần sai → khóa 15 phút - JWT token: hết hạn 24h - Bắt buộc HTTPS - Thời gian phản hồi < 2 giây - Log tất cả attempts
Luồng	<p>B1. Người dùng truy cập trang đăng nhập.</p> <p>B2. Người dùng nhập email (hoặc username) và mật khẩu.</p> <p>B3. Hệ thống xác thực thông tin đăng nhập.</p> <p>Nếu thành công: Hệ thống tạo một JWT token có hiệu lực trong 24 giờ, lưu phiên làm việc</p> <p>Nếu thất bại: Hiển thị thông báo lỗi. Nếu đăng nhập sai quá 5 lần, tài khoản sẽ bị khóa tạm thời trong 15 phút</p>

Bảng 4.4 Mô tả Usecase cho chức năng đăng nhập

Tên Use Case	Đăng xuất
Mô tả Use Case	Người dùng kết thúc phiên làm việc và thoát khỏi hệ thống an toàn.
Tác nhân	Người dùng
Mức độ ưu tiên	Trung bình

Điều kiện kích hoạt	Người dùng nhấn nút đăng xuất hoặc session hết hạn
Điều kiện tiền đề	<ul style="list-style-type: none"> - Người dùng đang đăng nhập - Session/token còn hiệu lực
Ràng buộc	<ul style="list-style-type: none"> - Token vô hiệu hóa ngay lập tức - Xóa tất cả dữ liệu phía client - Thời gian phản hồi < 1 giây
Luồng	<ul style="list-style-type: none"> B1. Người dùng nhấn vào nút "Đăng xuất" B2. Hệ thống xác minh JWT token B3. Hệ thống thêm token vào blacklist B4. Hệ thống lưu vào cơ sở dữ liệu B5. Hệ thống thông báo thành công

Bảng 4.5 Mô tả Usecase cho chức năng đăng xuất

Tên Use Case	Quên mật khẩu
Mô tả Use Case	Người dùng yêu cầu đặt lại mật khẩu khi không nhớ mật khẩu hiện tại.
Tác nhân	Người dùng
Mức độ ưu tiên	Cao
Điều kiện kích hoạt	Người dùng nhập email và yêu cầu reset password
Điều kiện tiền đề	<ul style="list-style-type: none"> - Email tồn tại trong hệ thống - Dịch vụ Email hoạt động - Tài khoản chưa bị khóa - Chưa vượt quá giới hạn yêu cầu
Ràng buộc	<ul style="list-style-type: none"> - Reset link: 1h hết hạn

	<ul style="list-style-type: none"> - Tối đa 3 yêu cầu/email/ngày - Secure token cho reset link - Email gửi trong 30 giây
Luồng	<p>B1. Người dùng truy cập trang đổi mật khẩu</p> <p>B2. Hệ thống hiển thị form đổi mật khẩu</p> <p>B3. Người dùng nhập mật khẩu hiện tại, mật khẩu mới và nhập lại mật khẩu mới</p> <p>B4. Người dùng ấn nút “đổi mật khẩu”</p> <p>B5. Hệ thống xác minh đầu vào</p> <p>B6. Hệ thống xác thực JWT token</p> <p>B7. Hệ thống mã hóa mật khẩu và cập nhật mật khẩu mới vào cơ sở dữ liệu</p>

Bảng 4.6 Mô tả Usecase cho chức năng quên mật khẩu

Tên Use Case	Quản lý hồ sơ
Mô tả Use Case	Người dùng xem và cập nhật thông tin cá nhân trong hệ thống
Tác nhân	Người dùng
Mức độ ưu tiên	Trung bình
Điều kiện kích hoạt	Người dùng truy cập trang cá nhân và thực hiện thay đổi
Điều kiện tiền đề	<ul style="list-style-type: none"> - Đã đăng nhập - Có quyền truy cập trang cá nhân - Dữ liệu đầu vào hợp lệ
Ràng buộc	<ul style="list-style-type: none"> - Chỉ sửa thông tin cá nhân của mình - Một số trường chỉ đọc - Xác thực đầu vào

	<ul style="list-style-type: none"> - Xác thực thời gian thực
Luồng	<p>B1. Người dùng truy cập trang hồ sơ cá nhân</p> <p>B2. Hệ thống hiển thị các thông tin cá nhân của người dùng.</p> <p>B3. Người dùng thay đổi các thông tin cho phép và lưu lại.</p> <p>B4. Hệ thống cập nhật thông tin cá nhân vào cơ sở dữ liệu. Nếu thất bại: Hiển thị thông báo lỗi nếu dữ liệu nhập vào không hợp lệ hoặc có lỗi hệ thống.</p>

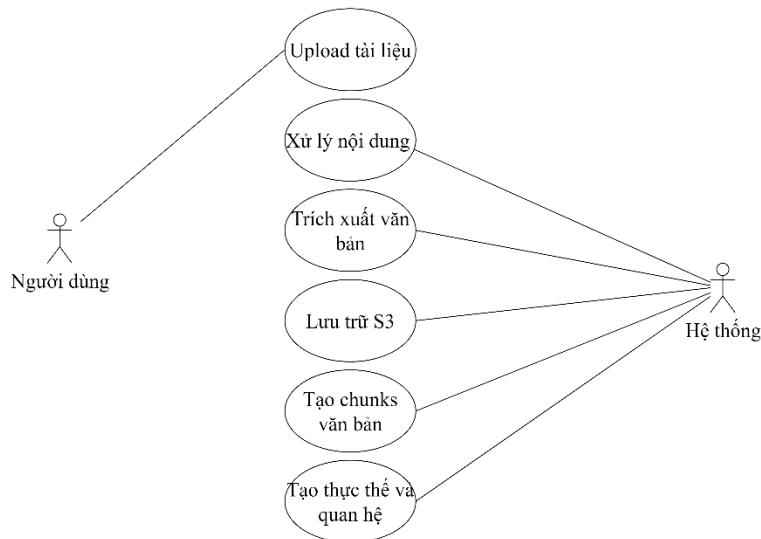
Bảng 4.7 Mô tả Usecase cho chức năng quản lý hồ sơ

Tên Use Case	Xem thông kê
Mô tả Use Case	Người dùng xem các thông kê về hoạt động sử dụng hệ thống của mình.
Tác nhân	Người dùng
Mức độ ưu tiên	Thấp
Điều kiện kích hoạt	Người dùng truy cập trang thông kê cá nhân
Điều kiện tiền đề	<ul style="list-style-type: none"> - Đã đăng nhập - Có dữ liệu hoạt động - Analytics service hoạt động
Ràng buộc	<ul style="list-style-type: none"> - Chỉ data của user hiện tại - Cache 15 phút - Cập nhật thời gian thực - Bộ lọc khoảng thời gian
Luồng	B1. Người dùng truy cập trang thông kê cá nhân.

	<p>B2. Hệ thống tổng hợp và hiển thị dữ liệu hoạt động của người dùng.</p> <p>B3. Một bảng điều khiển (dashboard) với các biểu đồ và số liệu được hiển thị. Dữ liệu được cache trong 15 phút để tối ưu hiệu suất.</p>
--	---

Bảng 4.8 Mô tả Usecase cho chức năng thống kê

Quản lý tài liệu



Hình 4.11 Lược đồ Usecase cho chức năng quản lý tài liệu

Tên Use Case	Upload tài liệu
Mô tả Use Case	Người dùng tải lên tài liệu PDF để hệ thống xử lý và tạo cơ sở tri thức cho chatbot
Tác nhân	Người dùng
Mức độ ưu tiên	Cao
Điều kiện kích hoạt	<ul style="list-style-type: none"> - Người dùng chọn file PDF hợp lệ và nhấn nút upload.
Điều kiện tiền đề	<ul style="list-style-type: none"> - Đã đăng nhập hệ thống - File định dạng PDF

	<ul style="list-style-type: none"> - Kích thước $\leq 10\text{MB}$ - Kết nối internet ổn định
Ràng buộc	<ul style="list-style-type: none"> - Chỉ chấp nhận định dạng PDF - Upload timeout: Vô hạn
Luồng	<ul style="list-style-type: none"> B1. Người dùng chọn tài liệu B2. Hệ thống lưu tài liệu lên S3 B3. Hệ thống đọc tài liệu từ S3 và tạo ra mảng chuỗi, mỗi phần tử là nội dung của hai trang B4. Hệ thống đọc mảng và chia chunks B5. Hệ thống đọc mảng và trích xuất thực thể hữu ích và quan hệ B6. Hệ thống thông báo thành công và tạo ra hội thoại mới

Bảng 4.9 Mô tả Usecase cho chức năng upload tài liệu

Tên Use Case	Trích xuất văn bản
Mô tả Use Case	Hệ thống trích xuất nội dung từ PDF đã được xử lý
Tác nhân	Hệ thống
Mức độ ưu tiên	Cao
Điều kiện kích hoạt	<ul style="list-style-type: none"> - Quá trình xử lý PDF đã hoàn tất thành công
Điều kiện tiền đề	<ul style="list-style-type: none"> - PDF chứa văn bản có thể trích xuất - Bộ nhớ đủ để xử lý - Dịch vụ trích xuất văn bản đang hoạt động
Ràng buộc	<ul style="list-style-type: none"> - Hỗ trợ tiếng Việt và tiếng Anh - Giữ lại định dạng cơ bản

	<ul style="list-style-type: none"> - Loại bỏ tiêu đề và chân trang - Xử lý các ký tự đặc biệt
Luồng	<p>B1. Người dùng chọn tài liệu</p> <p>B2. Hệ thống lưu tài liệu lên S3</p> <p>B3. Hệ thống đọc tài liệu từ S3 và tạo ra mảng chuỗi, mỗi phần tử gồm nội dung của hai trang</p>

Bảng 4.10 Mô tả Usecase cho chức năng trích xuất văn bản

Tên Use Case	Lưu trữ trên S3
Mô tả Use Case	Hệ thống lưu trữ file PDF lên AWS S3 cloud storage
Tác nhân	Hệ thống
Mức độ ưu tiên	Cao
Điều kiện kích hoạt	<ul style="list-style-type: none"> - Tệp PDF cần được lưu trữ vĩnh viễn
Điều kiện tiền đề	<ul style="list-style-type: none"> - Thông tin xác thực AWS S3 hợp lệ - Kết nối Internet ổn định - Đủ hạn mức lưu trữ - Dịch vụ S3 đang khả dụng
Ràng buộc	<ul style="list-style-type: none"> - Mã hóa tệp trước khi tải lên - Tạo tên tệp duy nhất - Chính sách sao lưu tự động - Quản lý vòng đời dữ liệu - Ghi nhật ký truy cập
Luồng	<p>B1. Người dùng chọn tài liệu</p> <p>B2. Hệ thống lưu tài liệu lên S3</p>

Bảng 4.11 Mô tả Usecase cho chức năng lưu trữ trên S3

Tên Use Case	Tạo chunks văn bản
Mô tả Use Case	Hệ thống chia nhỏ văn bản thành các đoạn nhỏ để lưu vào cơ sở dữ liệu vector Qdrant
Tác nhân	Hệ thống
Mức độ ưu tiên	Cao
Điều kiện kích hoạt	<ul style="list-style-type: none"> - Hệ thống nhận được yêu cầu xử lý tài liệu đã được phân đoạn thành các đoạn văn bản - Có sẵn document_id của tài liệu cần xử lý - Quá trình tải lên và tiền xử lý tài liệu đã hoàn tất
Điều kiện tiền đề	<ul style="list-style-type: none"> - Tài liệu đã được tải lên và xử lý thành các đoạn văn bản - Đã có ID tài liệu (document_id) - Các đối tượng cần thiết đã được khởi tạo - Có kết nối ổn định đến cơ sở dữ liệu Qdrant - Mô hình LLM đã được cấu hình đúng
Ràng buộc	<ul style="list-style-type: none"> - Giữ nguyên ngữ cảnh câu/đoạn văn - Xác thực chất lượng
Luồng	<p>B1. Hệ thống đọc mảng chuỗi, mỗi phần tử là nội dung của hai trang</p> <p>B2. Hệ thống sử dụng prompt và LLM để chia chunk.</p> <p>B3. Hệ thống trả về mảng json, mỗi phần tử là json có các thuộc tính “đoạn 1”, “đoạn 2”, “đoạn 3”...</p> <p>B4. Hệ thống sử dụng model nhúng để nhúng từng chunk</p> <p>B5. Hệ thống tạo collection sử dụng UUID làm tên.</p> <p>B6. Lưu các chunk đã nhúng vào Qdrant</p>

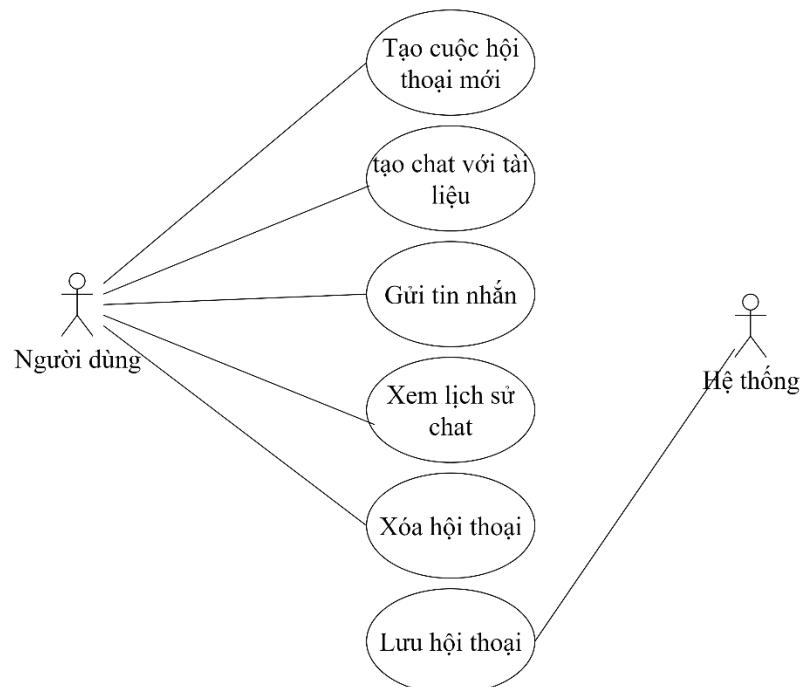
Bảng 4.12 Mô tả Usecase cho chức năng tạo chunks văn bản

Tên Use Case	Tạo thực thể và quan hệ
Mô tả Use Case	Tạo các thực thể hữu ích và quan hệ để lưu vào cơ sở dữ liệu đồ thị Neo4j
Tác nhân	Hệ thống
Mức độ ưu tiên	Cao
Điều kiện kích hoạt	<ul style="list-style-type: none"> - Hệ thống nhận được yêu cầu xử lý tài liệu đã được phân đoạn thành các đoạn văn bản - Có sẵn document_id của tài liệu cần xử lý - Quá trình tải lên và tiền xử lý tài liệu đã hoàn tất
Điều kiện tiền đề	<ul style="list-style-type: none"> - Tài liệu đã được tải lên và xử lý thành các đoạn văn bản. - Đã có ID tài liệu (document_id) - Các đối tượng cần thiết đã được khởi tạo. - Có kết nối ổn định đến cơ sở dữ liệu Neo4j - Mô hình LLM đã được cấu hình đúng
Ràng buộc	<ul style="list-style-type: none"> - Xử lý giới hạn tốc độ API của mô hình LLM (sleep 45 giây sau mỗi 2 lần gọi) - Dữ liệu trích xuất phải được chuyển đổi đúng định dạng JSON - Khi gặp lỗi phải ghi log chi tiết và ném ngoại lệ HTTP 500 - Chức năng cần xử lý hiệu quả với các tài liệu lớn - Đảm bảo dữ liệu được lưu trữ an toàn trong Neo4j - Cấu trúc đồ thị trong Neo4j phải được thiết kế để hỗ trợ truy vấn hiệu quả

Luồng	<p>B1. Hệ thống đọc mảng chuỗi, mỗi phần tử là nội dung của hai trang</p> <p>B2. Hệ thống sử dụng prompt và LLM để trích xuất thực thể hữu ích và quan hệ</p> <p>B3. Hệ thống trả về mảng json, mỗi phần tử là json có thuộc tính là “relationship” là một mảng các json có các thuộc tính “source”, “target”, “relation”, “source_type” và “source_target”, tất cả là chuỗi.</p> <p>B4. Hệ thống sử dụng prompt và LLM để tạo ra tiêu đề cho mỗi hai trang</p> <p>B5. Hệ thống lưu tiêu đề và nội dung tương ứng vào Neo4j</p>
--------------	---

Bảng 4.13 Mô tả Usecase cho chức năng tạo thực thể và quan hệ

Hệ thống chat



Hình 4.12 Lược đồ Usecase cho chức năng chat

Tên Use Case	Tạo cuộc hội thoại mới
Mô tả Use Case	Người dùng khởi tạo cuộc hội thoại mới với chatbot để hỏi về các vấn đề học vụ
Tác nhân	Người dùng
Mức độ ưu tiên	Cao
Điều kiện kích hoạt	<ul style="list-style-type: none"> - Người dùng nhấn "Cuộc hội thoại mới" hoặc gửi tin nhắn đầu tiên
Điều kiện tiền đề	<ul style="list-style-type: none"> - Đã đăng nhập hệ thống - Dịch vụ chat hoạt động - Chưa vượt giới hạn cuộc hội thoại - Mô hình AI sẵn sàng để sử dụng
Ràng buộc	<ul style="list-style-type: none"> - Tối đa 50 cuộc hội thoại cho mỗi người dùng - Tự động tạo tiêu đề cho cuộc hội thoại - Theo dõi metadata cuộc hội thoại - Giao diện cập nhật theo thời gian thực
Luồng	<p>B1. Người dùng chọn chat với tài liệu hoặc chat với sổ tay sinh viên</p> <p>B2. Hệ thống mở giao diện chat mới</p> <p>B3. Người dùng gửi tin nhắn</p> <p>B4. Hệ thống sẽ tạo hội thoại mới và lưu xuống cơ sở dữ liệu</p>

Bảng 4.14 Mô tả Usecase cho chức năng tạo cuộc hội thoại mới

Tên Use Case	Tạo chat với tài liệu
Mô tả Use Case	Người dùng tạo cuộc hội thoại dựa trên tài liệu cụ thể.
Tác nhân	Người dùng

Mức độ ưu tiên	Cao
Điều kiện kích hoạt	<ul style="list-style-type: none"> - Người dùng chọn “Chat with document” và chọn tài liệu pdf.
Điều kiện tiền đề	<ul style="list-style-type: none"> - Đã đăng nhập - Tài liệu đã được xử lý thành công - Embedding của tài liệu đã sẵn sàng
Ràng buộc	<ul style="list-style-type: none"> - Tài liệu cần được xử lý hoàn chỉnh - Quản lý cửa sổ ngữ cảnh - Truy xuất thông tin theo từng tài liệu - Trích dẫn nguồn trong câu trả lời
Luồng	<p>B1. Người dùng chọn chat với tài liệu và chọn tài liệu cần trả lời</p> <p>B2. Hệ thống sẽ tài tài liệu</p> <p>B3. Nếu tải thành công hệ thống sẽ tạo hội thoại mới và lưu xuống cơ sở dữ liệu</p>

Bảng 4.15 Mô tả Usecase cho chức năng tạo chat với tài liệu

Tên Use Case	Gửi tin nhắn
Mô tả Use Case	Người dùng gửi câu hỏi hoặc tin nhắn cho chatbot trong cuộc hội thoại
Tác nhân	Người dùng
Mức độ ưu tiên	Cao
Điều kiện kích hoạt	<ul style="list-style-type: none"> - Người dùng nhập tin nhắn và nhấn gửi hoặc Enter
Điều kiện tiền đề	<ul style="list-style-type: none"> - Đã đăng nhập - Cuộc hội thoại đã tồn tại

	<ul style="list-style-type: none"> - Tin nhắn không được đê trống - Dịch vụ trò chuyện khả dụng - Người dùng chưa bị giới hạn tốc độ gửi tin nhắn
Ràng buộc	<ul style="list-style-type: none"> - Độ dài tối đa mỗi tin nhắn: 2000 ký tự - Giới hạn tốc độ: 10 tin nhắn mỗi phút - Làm sạch dữ liệu đầu vào - Gửi tin nhắn theo thời gian thực
Luồng	<p>B1. Người dùng nhập câu hỏi vào ô chat và nhấn gửi.</p> <p>B2. Hệ thống kiểm tra văn bản (tối đa 2000 ký tự) và giới hạn tốc độ 10 tin nhắn/phút.</p> <p>B3. Hệ thống trả lời câu hỏi và lưu vào cơ sở dữ liệu</p> <p>B4. Hiển thị câu trả lời</p>

Bảng 4.16 Mô tả Usecase cho chức năng gửi tin nhắn

Tên Use Case	Xem lịch sử chat
Mô tả Use Case	Người dùng xem lại các cuộc hội thoại đã có với chatbot
Tác nhân	Người dùng
Mức độ ưu tiên	Trung bình
Điều kiện kích hoạt	<ul style="list-style-type: none"> - Người dùng truy cập thanh bên cuộc hội thoại hoặc trang lịch sử
Điều kiện tiền đề	<ul style="list-style-type: none"> - Đã đăng nhập - Có ít nhất một cuộc hội thoại đã được lưu - Cơ sở dữ liệu truy cập được
Ràng buộc	<ul style="list-style-type: none"> - Phân trang: 20 cuộc hội thoại mỗi trang - Tìm kiếm theo nội dung hoặc ngày tạo

	<ul style="list-style-type: none"> - Sắp xếp theo thời gian gần nhất hoặc mức độ liên quan - Tải tin nhắn theo dạng lazy loading
Luồng	<p>B1. Người dùng truy cập vào thanh bên chứa danh sách các cuộc hội thoại hoặc trang lịch sử chat.</p> <p>B2. Hệ thống hiển thị danh sách các cuộc hội thoại đã được lưu. Giao diện hỗ trợ phân trang (20 cuộc hội thoại mỗi trang) và tải tin nhắn theo kiểu "lazy loading" để tối ưu hiệu suất.</p>

Bảng 4.17 Mô tả Usecase cho chức năng xem lịch sử chat

Tên Use Case	Xóa hội thoại
Mô tả Use Case	Người dùng xóa cuộc hội thoại không cần thiết để dọn dẹp
Tác nhân	Người dùng
Mức độ ưu tiên	Thấp
Điều kiện kích hoạt	<ul style="list-style-type: none"> - Người dùng chọn cuộc hội thoại và xác nhận xóa
Điều kiện tiền đề	<ul style="list-style-type: none"> - Đã đăng nhập - Cuộc hội thoại tồn tại và thuộc về người dùng hiện tại - Người dùng có quyền xóa cuộc hội thoại - Cuộc hội thoại không đang hoạt động (không phải cuộc trò chuyện đang mở)
Ràng buộc	<ul style="list-style-type: none"> - Yêu cầu hộp thoại xác nhận trước khi xóa - Xóa mềm – cho phép khôi phục trong vòng 30 ngày - Xóa kèm toàn bộ tin nhắn liên quan (xóa đệ quy)

	<ul style="list-style-type: none"> - Ghi nhận hành động vào nhật ký kiểm tra (audit log)
Luồng	<p>B1. Người dùng truy cập vào thanh bên chứa danh sách các cuộc hội thoại hoặc trang lịch sử chat</p> <p>B2. Người dùng có thể chọn một cuộc hội thoại và ấn vào biểu tượng thùng rác</p> <p>B3. Hệ thống xóa hội thoại khỏi cơ sở dữ liệu</p>

Bảng 4.18 Mô tả Usecase cho chức năng xóa cuộc hội thoại

Tên Use Case	Lưu hội thoại
Mô tả Use Case	Hệ thống lưu trữ tin nhắn của cuộc hội thoại
Tác nhân	Hệ thống
Mức độ ưu tiên	Cao
Điều kiện kích hoạt	<ul style="list-style-type: none"> - Việc trao đổi tin nhắn đã hoàn tất
Điều kiện tiền đề	<ul style="list-style-type: none"> - Kết nối cơ sở dữ liệu khả dụng - Không gian lưu trữ đủ dùng - Dữ liệu đã vượt qua kiểm định hợp lệ - Quyền của người dùng đã được xác minh
Ràng buộc	<ul style="list-style-type: none"> - Tuân thủ nguyên tắc ACID - Dữ liệu được mã hóa khi lưu trữ - Chiến lược sao lưu dữ liệu - Chính sách lưu giữ dữ liệu
Luồng	<p>B1. Người dùng chọn chat với tài liệu hoặc chat với số tay sinh viên</p>

	<p>B2. Nếu chọn chat với số tay sinh viên thì người dùng gửi tin nhắn thì hệ thống sẽ tạo hội thoại mới và lưu xuống cơ sở dữ liệu</p> <p>B3. Nếu chọn chat với tài liệu thì người dùng chọn tài liệu cần trả lời, nếu tài liệu được tải thành công thì hệ thống sẽ tạo hội thoại mới và lưu xuống cơ sở dữ liệu</p>
--	--

Bảng 4.19 Mô tả Usecase cho chức năng lưu cuộc hội thoại

4.9.4. Môi trường triển khai ứng dụng

Sử dụng IntelliJ để chạy BE và Visual Studio Code để chạy FE. Vì mục tiêu để tài là tập trung tạo ra một hệ thống tìm kiếm dữ liệu, do đó chúng em tập trung vào BE hơn là FE.

CHƯƠNG 5. KẾT QUẢ ĐẠT ĐƯỢC

5.1 Nghiên cứu

- Ứng dụng thành công LLMs: Đã nghiên cứu và áp dụng hiệu quả mô hình Gemini-1.5-flash làm mô hình cơ sở và Gemini-2.0-flash cho vai trò tạo sinh văn bản trong hệ thống.
- Làm chủ các kỹ thuật truy xuất tăng cường: Đã nghiên cứu sâu và kết hợp thành công hai phương pháp RAG (truy xuất từ cơ sở dữ liệu vector Qdrant) và GRAG (truy xuất từ cơ sở dữ liệu đồ thị Neo4j) kèm theo cơ chế tự phản ánh.
- Xây dựng kiến trúc Agentic tiên tiến: Đã thiết kế và triển khai một hệ thống Agentic, trong đó Agent sử dụng LLM để tự động lý luận, lựa chọn nguồn truy xuất và tự điều chỉnh hành vi thông qua mô-đun phê bình.
- Ứng dụng Framework và thư viện hiện đại: Đã nghiên cứu và sử dụng thành thạo framework LangChain để đơn giản hóa việc tích hợp LLM và thư viện pyvis để trực quan hóa đồ thị tri thức.

5.2 Sản phẩm

- Xây dựng hệ thống Chatbot hoàn chỉnh: Đã xây dựng thành công một hệ thống chatbot có khả năng tư vấn về học vụ của trường Đại học Nông Lâm, dựa trên nguồn tri thức từ số tay sinh viên 2024.

5.3 Kết quả thực nghiệm

- Chứng minh hiệu quả vượt trội của kiến trúc đề xuất: Thông qua việc đánh giá trên tập dữ liệu gồm 543 cặp câu hỏi-trả lời, kiến trúc đầy đủ (Agent + RAG + GRAG) đã cho thấy hiệu suất vượt trội so với việc chỉ sử dụng RAG hoặc GRAG riêng lẻ.
 - Độ chính xác (Accuracy): Kiến trúc đầy đủ đạt 95.20%, cao hơn đáng kể so với RAG (84.32%) và GRAG (28.41%).

- Tỷ lệ thiếu thông tin (Missing Rate): Kiến trúc đầy đủ có tỷ lệ thiếu thông tin thấp nhất là 3.32%, so với 14.58% của RAG và 54.43% của GRAG.
- Độ tương đồng Cosine trung bình: Điểm tương đồng trung bình của kiến trúc đầy đủ là 0.88, cao hơn RAG (0.84) và vượt trội so với GRAG (0.31), cho thấy sự ổn định và chất lượng của câu trả lời.
- Hạn chế hiện tượng ảo giác (Hallucination): Kiến trúc đầy đủ duy trì được tỷ lệ ảo giác ở mức rất thấp (1.48%), chứng tỏ khả năng bám sát vào ngữ cảnh được cung cấp. Các biểu đồ phân phối cũng cho thấy các câu trả lời của kiến trúc này tập trung chủ yếu ở mức điểm tương đồng cao (trên 0.8), trong khi các kiến trúc khác có sự phân tán lớn hơn về phía điểm thấp.

CHƯƠNG 6. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

6.1 Kết luận

Đề tài "Xây dựng chatbot tư vấn học vụ Nông Lâm kết hợp giữa hệ thống GRAG và RAG" đã hoàn thành các mục tiêu đề ra. Chúng em đã nghiên cứu và triển khai thành công một kiến trúc chatbot tiên tiến, kết hợp linh hoạt giữa việc truy xuất văn bản phẳng (RAG) và truy xuất đồ thị tri thức (GRAG).

Điểm nổi bật của đề tài là việc xây dựng một hệ thống Agentic thông minh, trong đó Agent sử dụng mô hình Gemini để tự động phân tích câu hỏi, lựa chọn nguồn truy xuất phù hợp và tự sửa lỗi thông qua mô-đun phê bình. Cách tiếp cận này đã giải quyết được những hạn chế của các hệ thống RAG và GRAG truyền thống khi hoạt động riêng lẻ.

6.2 Hướng phát triển

- Có thể cải tiến prompt của Agent và Commentor để có cơ chế tự sửa lỗi tốt hơn.
- Cần tìm một chiến lược chia chunk và trích xuất thực thể hữu ích, quan hệ chính xác và đỡ tốn chi phí API.
- Nâng cấp mỗi cuộc hội thoại có nhiều tài liệu kiến thức.

- Nghiên cứu và sử dụng các LLM khác tốt hơn trong nhúng tài liệu, trả lời câu hỏi và các thành phần dùng LLM khác.

CHƯƠNG 7. TÀI LIỆU THAM KHẢO

- [1] Shailja Gupta, Rajesh Ranjan, Surya Narayan Singh, "A Comprehensive Survey of Retrieval-Augmented Generation (RAG): Evolution, Current Landscape and Future Directions," 2024. [Online]. Available: <https://arxiv.org/abs/2410.12837>.
- [2] Meng-Chieh Lee, Qi Zhu, Costas Mavromatis, Zhen Han, Soji Adeshina, Vassilis N. Ioannidis, Huzefa Rangwala, Christos Faloutsos, "Agent-G: An Agentic Framework for Graph Retrieval Augmented Generation," 2024. [Online]. Available: <https://openreview.net/forum?id=g2C947jjjQ>.
- [3] Sundar Pichai, Demis Hassabis, "Introducing Gemini: our largest and most capable AI model," 2023. [Online]. Available: <https://blog.google/technology/ai/google-gemini-ai>.
- [4] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, Yuan Cao, "ReAct: Synergizing Reasoning and Acting in Language Models," 2022. [Online]. Available: <https://arxiv.org/abs/2210.03629>.
- [5] Trương Nguyễn Yên Nhi, "Tát tần tật về Chatbot (Phần 1)," 2024. [Trực tuyến]. Available: <https://ailab.siu.edu.vn/article/47/tat-tan-tat-ve-chatbot-phan-1>.
- [6] "Chatbot là gì?," AWS, [Trực tuyến]. Available: <https://aws.amazon.com/vi/what-is/chatbot>.
- [7] Văn Công Vũ, Lê Thị Ngọc Hoa, "GIẢI PHÁP ỦNG DỤNG CÔNG NGHỆ CHATBOT TRONG ĐÀO TẠO, BỒI DƯỠNG LÝ LUẬN CHÍNH TRỊ TRỰC TUYẾN TRONG THỜI ĐẠI CÁCH MẠNG CÔNG NGHIỆP 4.0," TẠP CHÍ KHOA HỌC YERSIN – CHUYÊN ĐỀ KHOA HỌC & CÔNG

- NGHỆ, 2020. [Trực tuyến]. Available: <https://vjol.info.vn/index.php/YERSIN/article/download/61945/52066/>.
- [8] Pham Nam, "Tổng quan về Chatbot," 2021. [Trực tuyến]. Available: <https://viblo.asia/p/tong-quan-ve-chatbot-yMnKMByaZ7P>.
- [9] "Chatbot: Lợi ích, hạn chế và các nền tảng phổ biến nhất 2024," 2024. [Online]. Available: <https://careerviet.vn/vi/talentcommunity/wiki-career/top-15-phan-mem-chatbot-mien-phi-tot-nhat-duoc-ua-chuong-hien-nay.35A52314.html>.
- [10] Ivan Belcic, "What is RAG (retrieval augmented generation)?," 2024. [Online]. Available: <https://www.ibm.com/think/topics/retrieval-augmented-generation>.
- [11] Nils Reimers, Iryna Gurevych, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks," 2019.
- [12] "Indexing," Qdrant, 2025. [Online]. Available: <https://qdrant.tech/documentation/concepts/indexing>.
- [13] "Query planning," Qdrant, 2025. [Online]. Available: <https://qdrant.tech/documentation/concepts/search>.
- [14] "Hybrid and Multi-Stage Queries," Qdrant, 2025. [Online]. Available: <https://qdrant.tech/documentation/concepts/hybrid-queries>.
- [15] Sharmila Ananthasayanam, "7 Chunking Strategies in RAG You Need To Know," 2025. [Online]. Available: <https://www.f22labs.com/blogs/7-chunking-strategies-in-rag-you-need-to-know>.
- [16] Long Nguyễn Thành, "Tôi Ưu Hóa RAG: Khám Phá 5 Chiến Lược Chunking Hiệu Quả Bạn Cần Biết," 2025. [Trực tuyến]. Available: <https://viblo.asia/p/toi-uu-hoa-rag-kham-pha-5-chien-luoc-chunking-hieu-qua-ban-can-biet-EvbLbPGW4nk>.
- [17] Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, Furu Wei Microsoft Corporation, "Multilingual E5 Text Embeddings: A Technical Report".

- [18] Xin Zhang^{1,2}, Yanzhao Zhang¹, Dingkun Long¹, Wen Xie¹, Ziqi Dai¹, Jialong Tang¹, Huan Lin¹ Baosong Yang¹, Pengjun Xie¹, Fei Huang¹, Meishan Zhang*, Wenjie Li², Min Zhang¹ Tongyi Lab, Alibaba Group, 2The Hong Kong Polytechnic University, "mGTE:Generalized Long-Context Text Representation and Reranking Models for Multilingual Text Retrieval".
- [19] Nils Reimers, Iryna Gurevych, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks".
- [20] Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, Matei Zaharia. , "ColBERTv2: Effective and Efficient Retrieval via Lightweight Late Interaction".
- [21] "Generative AI - Ground LLMs with Knowledge Graphs," Neo4j, [Online]. Available: <https://neo4j.com/generativeai>.
- [22] "NeoConverse - Graph Database Search with Natural Language," Neo4j, [Online]. Available: <https://neo4j.com/labs/genai-ecosystem/neoconverse>.
- [23] "A Vietnamese Multitask Language Understanding Benchmark Suite for Large Language Models," Zalo, 2025. [Online]. Available: <https://vmlu.ai/>.
- [24] Cuc Thi Bui, Nguyen Truong Son, Truong Van Trang, Lam Viet Phung, Pham Nhut Huy, Hoang Anh Le, Quoc Huu Van, Phong Nguyen-Thuan Do, Van Le Tran Truc, Duc Thanh Chau, Le-Minh Nguyen, "VMLU Benchmarks: A comprehensive benchmark toolkit for Vietnamese," 2025.
- [25] Shirley Wu, Shiyu Zhao, Michihiro Yasunaga, Kexin Huang, Kaidi Cao, Qian Huang, Vassilis N. Ioannidis, Karthik Subbian, James Zou, Jure Leskovec, Department of Computer Science, Stanford University Amazon, "STaRK: Benchmarking LLM Retrieval on Textual and Relational Knowledge Bases," 2024. [Online]. Available: <https://arxiv.org/abs/2404.13207>.
- [26] N. Y. X. H. L. Y. R. M. F. W. Liang Wang, "Multilingual E5 Text Embeddings: A Technical Report".

- [27] Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, Furu Wei Microsoft Corporation, "Multilingual E5 Text Embeddings: A Technical Report," p. <https://arxiv.org/pdf/2402.05672.pdf>.
- [28] Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, Furu Wei Microsoft Corporation, "Multilingual E5 Text Embeddings: A Technical Report," p. <https://arxiv.org/pdf/2402.05672.pdf>.
- [29] Nils Reimers, Iryna Gurevych, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks".

PHỤ LỤC

1. Chi tiết các bước thực hiện

1.1. Trích xuất dữ liệu

Hệ thống nhận một file pdf, lưu trữ lên S3 và đọc từ S3. Mỗi hai trang sẽ được gộp thành một đoạn. Kết quả là mảng chuỗi gồm các nội dung của pdf.



```
def upload_to_s3(file_content: bytes, file_name: str, document_id: str) -> dict:
    """Upload a file to S3"""
    instances = _get_global_instances()
    s3_client = instances['s3_client']

    if not s3_client:
        raise HTTPException(
            status_code=status.HTTP_500_INTERNAL_SERVER_ERROR,
            detail="S3 client not initialized"
        )

    try:
        # Create S3 key with document_id to avoid duplicates
        s3_key = f"documents/{document_id}/{file_name}"

        # Upload file to S3
        s3_client.put_object(
            Bucket=S3_BUCKET_NAME,
            Key=s3_key,
            Body=file_content,
            ContentType='application/pdf'
        )

        s3_url = f"s3://{S3_BUCKET_NAME}/{s3_key}"

        return {
            "s3_key": s3_key,
            "s3_url": s3_url
        }
    except Exception as e:
        print(f"Error uploading to S3: {str(e)}")
        raise HTTPException(
            status_code=status.HTTP_500_INTERNAL_SERVER_ERROR,
            detail=f"Failed to upload to S3: {str(e)}"
        )
    
```

Phương thức lưu tài liệu lên S3



```
def read_chunks(self):
    all_pages_text = []

    response = self.s3.get_object(Bucket=self.bucket_name, Key=self.key)
    file_stream = BytesIO(response['Body'].read())

    try:
        with pdfplumber.open(file_stream) as pdf:
            num_pages = len(pdf.pages)

            for i in range(0, num_pages, 2):
                group_text = []

                for j in range(i, min(i + 2, num_pages)):
                    page = pdf.pages[j]
                    text = page.extract_text()
                    if text:
                        group_text.append(text)

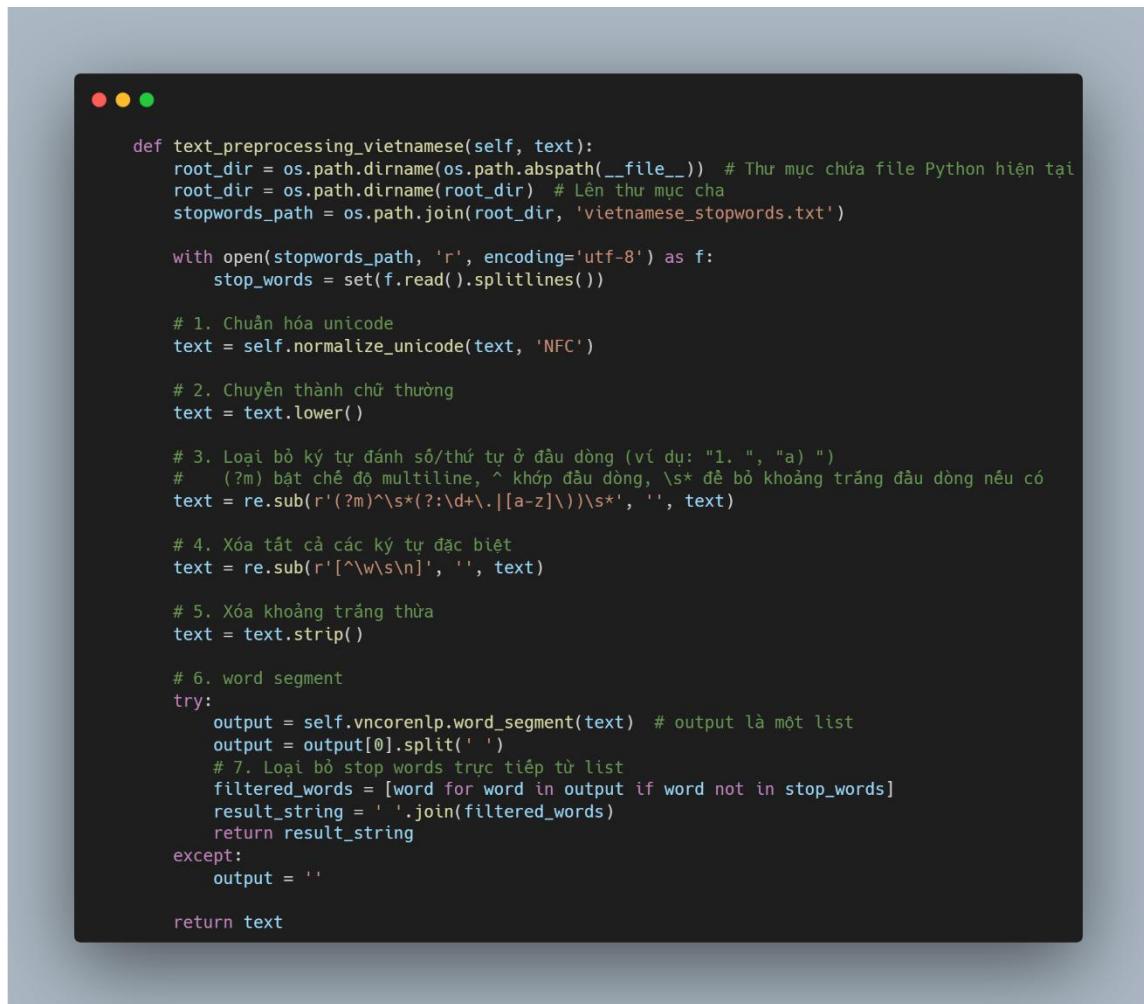
                if group_text:
                    all_pages_text.append('\n'.join(group_text))
    except Exception as e:
        print(f'Lỗi khi xử lý file {self.pdf_path}: {e}')

    return all_pages_text
```

Phương thức trích xuất nội dung file pdf

1.2. Tiền xử lý dữ liệu

Khi đã có mảng chuỗi các đoạn văn được trích xuất từ file pdf. Chúng em sẽ tiền xử lý dữ liệu cho các nội dung



```
def text_preprocessing_vietnamese(self, text):
    root_dir = os.path.dirname(os.path.abspath(__file__)) # Thư mục chứa file Python hiện tại
    root_dir = os.path.dirname(root_dir) # Lên thư mục cha
    stopwords_path = os.path.join(root_dir, 'vietnamese_stopwords.txt')

    with open(stopwords_path, 'r', encoding='utf-8') as f:
        stop_words = set(f.read().splitlines())

    # 1. Chuẩn hóa unicode
    text = self.normalize_unicode(text, 'NFC')

    # 2. Chuyển thành chữ thường
    text = text.lower()

    # 3. Loại bỏ ký tự đánh số/thứ tự ở đầu dòng (ví dụ: "1.", "a")
    #     (?m) bắt chế độ multiline, ^ khớp đầu dòng, \s* để bỏ khoảng trắng đầu dòng nếu có
    text = re.sub(r'(?m)^s*(?:\d+\.|[a-z])\s*', '', text)

    # 4. Xóa tất cả các ký tự đặc biệt
    text = re.sub(r'[^w\s\n]', '', text)

    # 5. Xóa khoảng trắng thừa
    text = text.strip()

    # 6. word segment
    try:
        output = self.vncorenlp.word_segment(text) # output là một list
        output = output[0].split(' ')
        # 7. Loại bỏ stop words trực tiếp từ list
        filtered_words = [word for word in output if word not in stop_words]
        result_string = ' '.join(filtered_words)
        return result_string
    except:
        output = ''
    return text
```

Phương thức tiền xử lý dữ liệu

Bước 1: Chuẩn hóa các ký tự tiếng Việt. Ví dụ: á → á để đảm bảo thống nhất.

Bước 2: Chuyển tất cả sang chữ thường.

Bước 3: Loại bỏ đánh số câu và ký hiệu liệt kê. Ví dụ như a), b), c) hoặc 1), 2), 3)...

Bước 4: Loại bỏ các ký tự đặc biệt như !, @, #, \$, %...

Bước 5: Xóa khoảng trắng đầu cuối.

Bước 6: Tách từ sử dụng thư viện VnCoreNLP.

“học sinh học sinh học” thành “học_sinh_học_sinh_học”

Bước 7: Loại bỏ các stopwords trong tiếng việt bằng cách đọc file

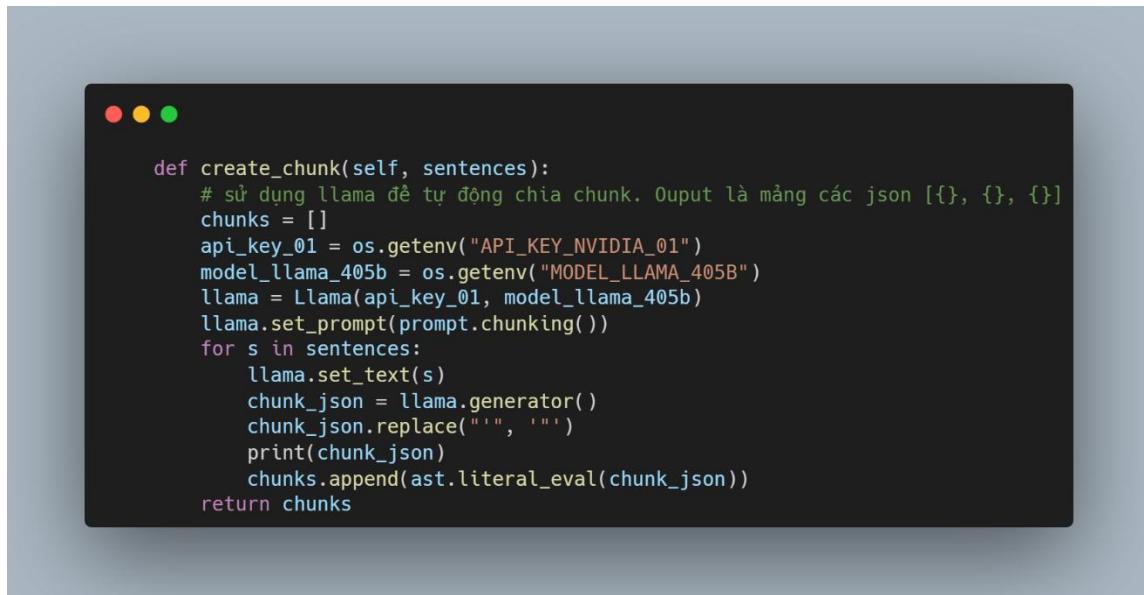
vietnamese_stopwords.txt, file này chứa tất cả các stopword trong tiếng việt như là “và”, “thì”, “bị”, “là”... Các từ này không có ý nghĩa nhiều trong một câu điều này sẽ làm cho câu nhúng không được tập trung ý nghĩa.

1.3. Thêm dữ liệu vào Qdrant

1.3.1. Tạo chunks

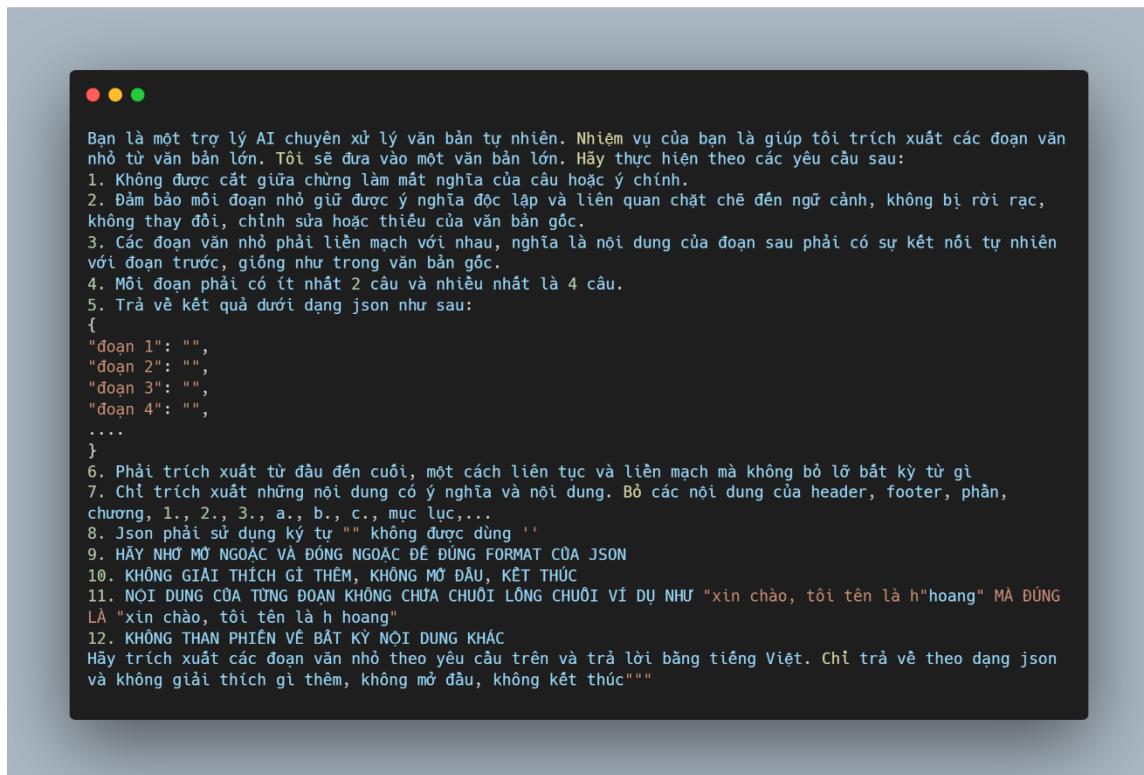
Từ mảng đã được tiền xử lý dữ liệu, chúng em sẽ sử dụng LLM meta/llama-3.1-405b-instruct để tạo chunk.

Lý do sử dụng mô hình meta/llama-3.1-405b-instruct là miễn phí API nhưng sẽ bị giới hạn tốc độ do Nvidia hỗ trợ



```
def create_chunk(self, sentences):
    # sử dụng llama để tự động chia chunk. Output là mảng các json [{}], {}, {}]
    chunks = []
    api_key_01 = os.getenv("API_KEY_NVIDIA_01")
    model_llama_405b = os.getenv("MODEL_LLAMA_405B")
    llama = Llama(api_key_01, model_llama_405b)
    llama.set_prompt(prompt.chunking())
    for s in sentences:
        llama.set_text(s)
        chunk_json = llama.generator()
        chunk_json.replace("'''", ' ')
        print(chunk_json)
        chunks.append(ast.literal_eval(chunk_json))
    return chunks
```

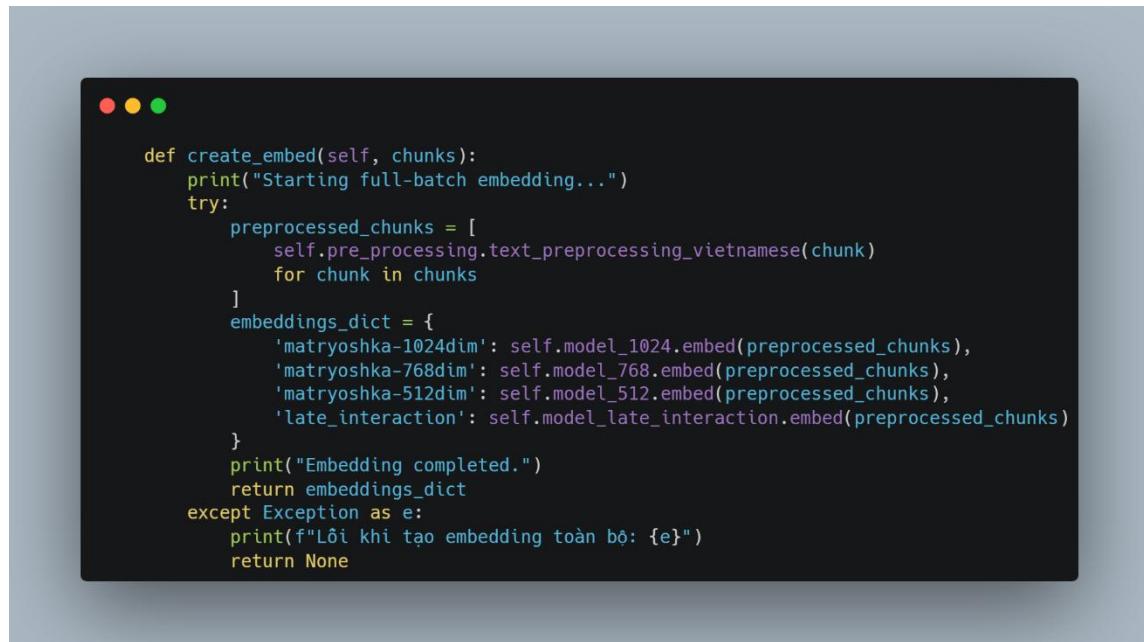
Phương thức tự động tạo chunk



Prompt sử dụng để tạo chunk

1.3.2. Nhúng

Sau khi đã có mảng chunk thì tiếp tục lặp qua và nhúng ở các chiều 1024, 768, 512 và vector late interaction



Phương thức nhúng văn bản

1.3.3. Thêm vào Qdrant

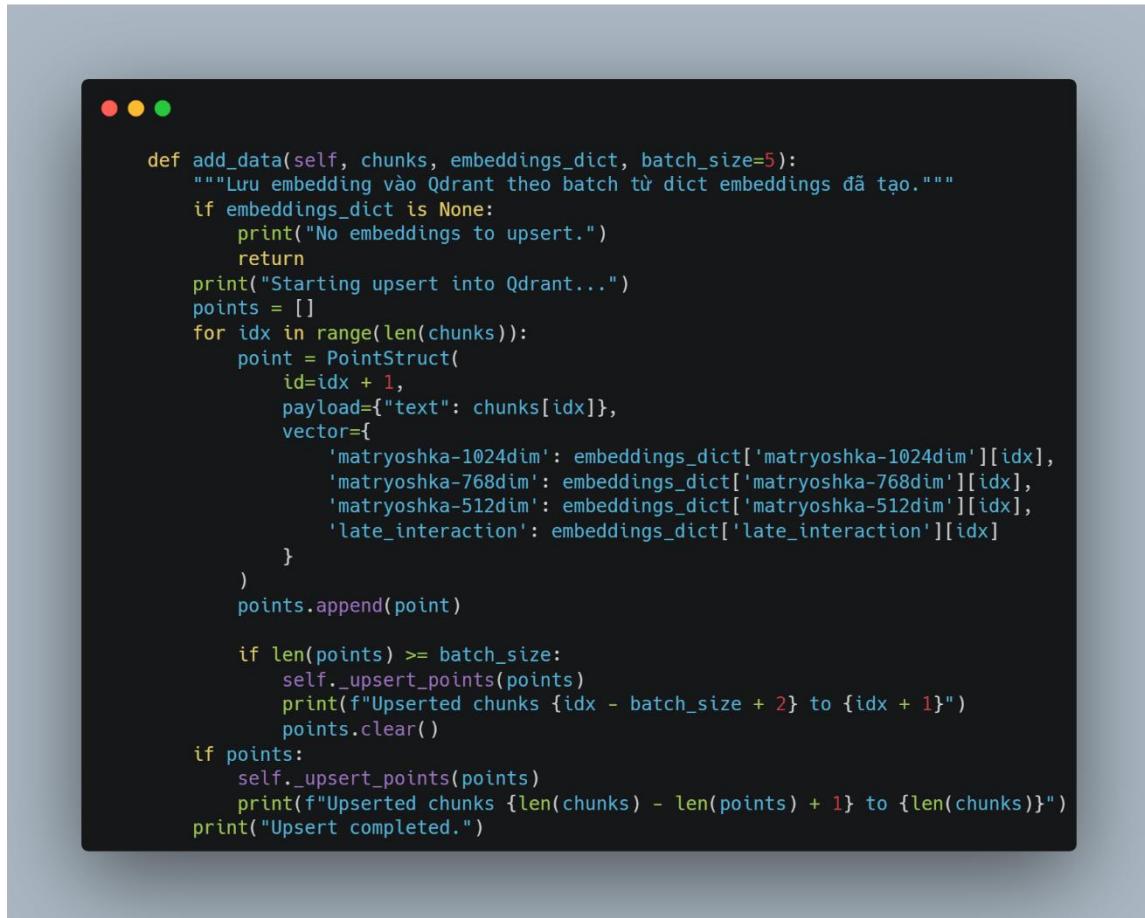
Trước khi lưu cần tạo collection trong Qdrant. Cần kiểm tra collection đã tồn tại chưa sau đó tạo mới.



```
def create_collection(self):
    if self.client.collection_exists(collection_name=self.collection_name):
        print("Collection đã tồn tại")
        return
    hnsw_config = {
        "m": 16,
        "ef_construct": 1000,
        "full_scan_threshold": 10000
    }
    self.client.create_collection(
        collection_name=self.collection_name,
        vectors_config={
            'matryoshka-1024dim': models.VectorParams(
                size=1024,
                distance=models.Distance.COSINE,
                datatype=models.Datatype.FLOAT32
            ),
            'matryoshka-768dim': models.VectorParams(
                size=768,
                distance=models.Distance.COSINE,
                datatype=models.Datatype.FLOAT32
            ),
            'matryoshka-512dim': models.VectorParams(
                size=512,
                distance=models.Distance.COSINE,
                datatype=models.Datatype.FLOAT32
            ),
            'late_interaction': models.VectorParams(
                size=128,
                distance=models.Distance.COSINE,
                multivector_config=models.MultiVectorConfig(
                    comparator=models.MultiVectorComparator.MAX_SIM
                )
            )
        },
        quantization_config=models.ScalarQuantization(
            scalar=models.ScalarQuantizationConfig(
                type=models.ScalarType.INT8,
                quantile=0.99,
                always_ram=True,
            ),
        ),
        hnsw_config=hnsw_config
    )
    print("create collection success")
    return self.client
```

Phương thức tạo collection cho Qdrant

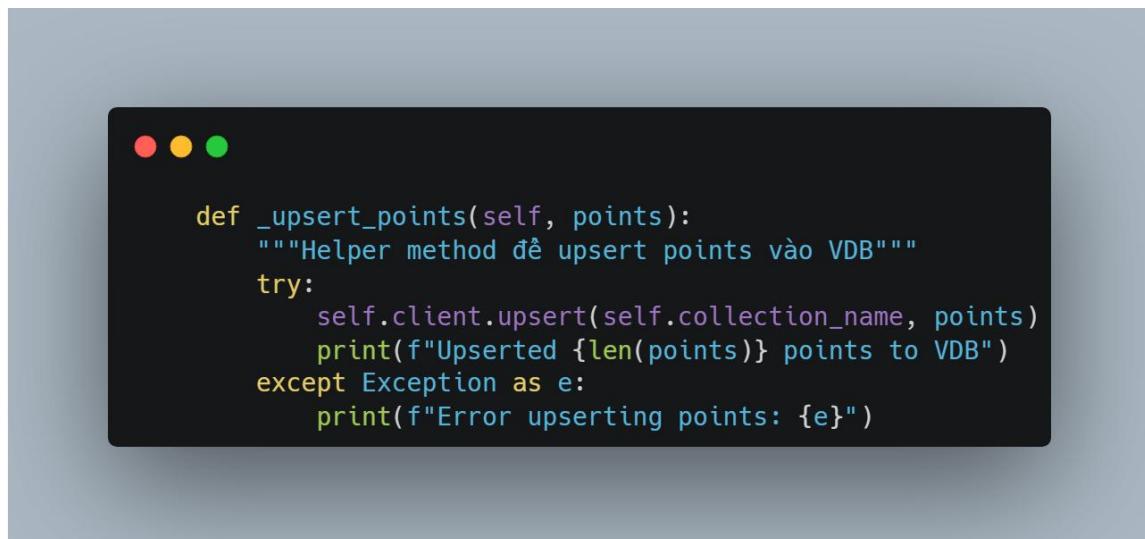
Lặp qua từng nhúng và lưu vào Qdrant theo batch



```
def add_data(self, chunks, embeddings_dict, batch_size=5):
    """Lưu embedding vào Qdrant theo batch từ dict embeddings đã tạo."""
    if embeddings_dict is None:
        print("No embeddings to upsert.")
        return
    print("Starting upsert into Qdrant...")
    points = []
    for idx in range(len(chunks)):
        point = PointStruct(
            id=idx + 1,
            payload={"text": chunks[idx]},
            vector={
                'matryoshka-1024dim': embeddings_dict['matryoshka-1024dim'][idx],
                'matryoshka-768dim': embeddings_dict['matryoshka-768dim'][idx],
                'matryoshka-512dim': embeddings_dict['matryoshka-512dim'][idx],
                'late_interaction': embeddings_dict['late_interaction'][idx]
            }
        )
        points.append(point)

    if len(points) >= batch_size:
        self._upsert_points(points)
        print(f"Upserted chunks {idx - batch_size + 2} to {idx + 1}")
        points.clear()
    if points:
        self._upsert_points(points)
        print(f"Upserted chunks {len(chunks) - len(points) + 1} to {len(chunks)}")
    print("Upsert completed.")
```

Phương thức lưu vào Qdrant theo batch



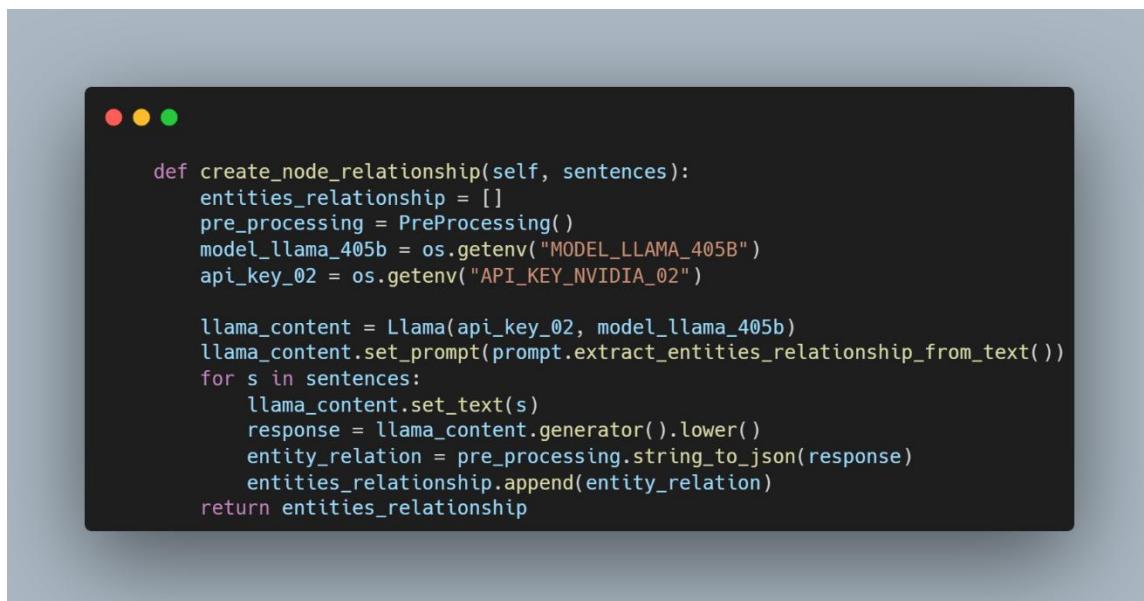
```
def _upsert_points(self, points):
    """Helper method để upsert points vào VDB"""
    try:
        self.client.upsert(self.collection_name, points)
        print(f"Upserted {len(points)} points to VDB")
    except Exception as e:
        print(f"Error upserting points: {e}")
```

Phương thức helper

1.4. Thêm dữ liệu vào Neo4j

1.4.1. Tạo thực thể hữu ích và mối quan hệ

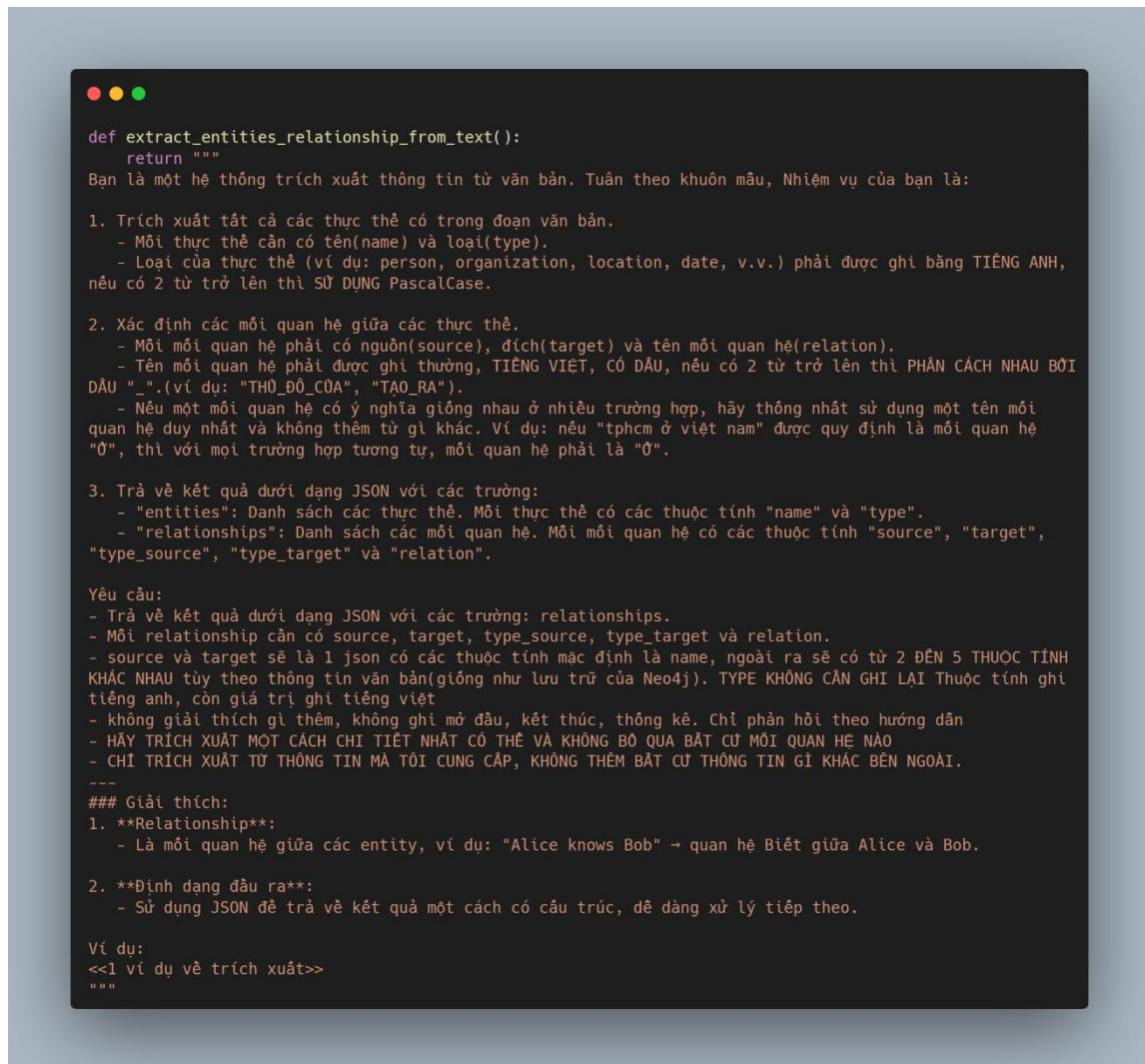
Tiếp tục lặp qua các đoạn văn bản đã được tiền xử lý và sử dụng LLM meta/llama-3.1-405b-instruct để tạo thực thể hữu ích và quan hệ.



```
def create_node_relationship(self, sentences):
    entities_relationship = []
    pre_processing = PreProcessing()
    model_llama_405b = os.getenv("MODEL_LLAMA_405B")
    api_key_02 = os.getenv("API_KEY_NVIDIA_02")

    llama_content = Llama(api_key_02, model_llama_405b)
    llama_content.set_prompt(prompt.extract_entities_relationship_from_text())
    for s in sentences:
        llama_content.set_text(s)
        response = llama_content.generator().lower()
        entity_relation = pre_processing.string_to_json(response)
        entities_relationship.append(entity_relation)
    return entities_relationship
```

Phương thức tạo thực thể và quan hệ



```
def extract_entities_relationship_from_text():
    return """
Bạn là một hệ thống trích xuất thông tin từ văn bản. Tuân theo khuôn mẫu, Nhiệm vụ của bạn là:
1. Trích xuất tất cả các thực thể có trong đoạn văn bản.
- Mỗi thực thể cần có tên(name) và loại(type).
- Loại của thực thể (ví dụ: person, organization, location, date, v.v.) phải được ghi bằng TIẾNG ANH,
nếu có 2 từ trở lên thì SỬ DỤNG PascalCase.

2. Xác định các mối quan hệ giữa các thực thể.
- Mỗi mối quan hệ phải có nguồn(source), đích(target) và tên mối quan hệ(relation).
- Tên mối quan hệ phải được ghi thường, TIẾNG VIỆT, CÓ DẤU, nếu có 2 từ trở lên thì PHÂN CÁCH NHAU BỞI
DẤU "_".(ví dụ: "THÔ ĐÔ CỦA", "TẠO_RA").
- Nếu một mối quan hệ có ý nghĩa giống nhau ở nhiều trường hợp, hãy thông nhất sử dụng một tên mối
quan hệ duy nhất và không thêm từ gì khác. Ví dụ: nếu "tphcm ở việt nam" được quy định là mối quan hệ
"Ở", thì với mọi trường hợp tương tự, mối quan hệ phải là "Ở".
3. Trả về kết quả dưới dạng JSON với các trường:
- "entities": Danh sách các thực thể. Mỗi thực thể có các thuộc tính "name" và "type".
- "relationships": Danh sách các mối quan hệ. Mỗi mối quan hệ có các thuộc tính "source", "target",
"type_source", "type_target" và "relation".
Yêu cầu:
- Trả về kết quả dưới dạng JSON với các trường: relationships.
- Mỗi relationship cần có source, target, type_source, type_target và relation.
- source và target sẽ là 1 json có các thuộc tính mặc định là name, ngoài ra sẽ có từ 2 ĐẾN 5 THUỘC TÍNH
KHÁC NHAU tùy theo thông tin văn bản(giống như lưu trữ của Neo4j). TYPE KHÔNG CẦN GHI LẠI Thuộc tính ghi
tiếng anh, còn giá trị ghi tiếng việt
- Không giải thích gì thêm, không ghi mở đầu, kết thúc, thông kê. Chỉ phản hồi theo hướng dẫn
- HÃY TRÍCH XUẤT MỘT CÁCH CHI TIẾT NHẤT CÓ THỂ VÀ KHÔNG BỎ QUA BẤT CỨ MỐI QUAN HỆ NÀO
- CHỈ TRÍCH XUẤT TỪ THÔNG TIN MÀ TÔI CUNG CẤP, KHÔNG THÊM BẤT CỨ THÔNG TIN GIÁ KHÁC BÊN NGOÀI.
---
### Giải thích:
1. **Relationship**:
- Là mối quan hệ giữa các entity, ví dụ: "Alice knows Bob" → quan hệ Biết giữa Alice và Bob.

2. **Định dạng đầu ra**:
- Sử dụng JSON để trả về kết quả một cách có cấu trúc, dễ dàng xử lý tiếp theo.

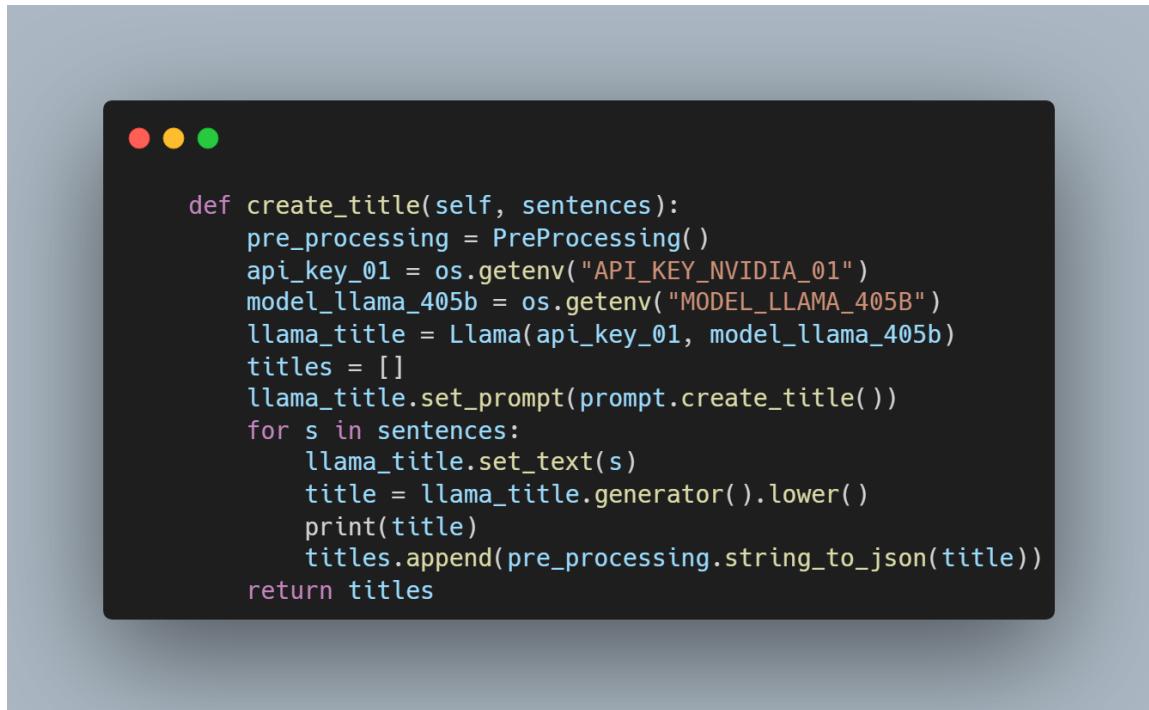
Ví dụ:
<<1 ví dụ về trích xuất>>
"""

```

Prompt tạo thực thể và quan hệ

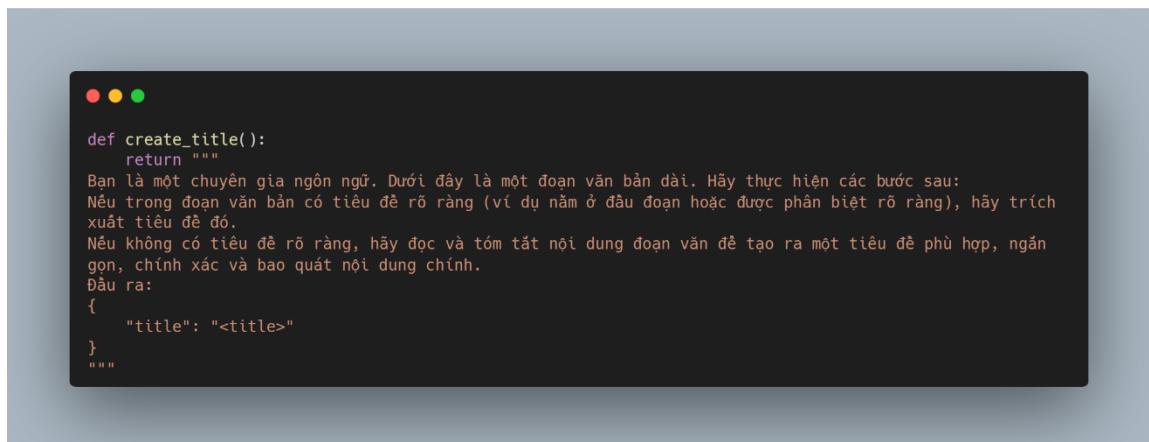
1.4.2. Tạo tiêu đề

Tiếp tục lặp qua các đoạn văn bản đã được tiền xử lý và sử dụng LLM meta/llama-3.1-405b-instruct để tạo tiêu đề.



```
def create_title(self, sentences):
    pre_processing = PreProcessing()
    api_key_01 = os.getenv("API_KEY_NVIDIA_01")
    model_llama_405b = os.getenv("MODEL_LLAMA_405B")
    llama_title = Llama(api_key_01, model_llama_405b)
    titles = []
    llama_title.set_prompt(prompt.create_title())
    for s in sentences:
        llama_title.set_text(s)
        title = llama_title.generator().lower()
        print(title)
        titles.append(pre_processing.string_to_json(title))
    return titles
```

Phương thức tạo tiêu đề



```
def create_title():
    return ""
Bạn là một chuyên gia ngôn ngữ. Dưới đây là một đoạn văn bản dài. Hãy thực hiện các bước sau:
Nếu trong đoạn văn bản có tiêu đề rõ ràng (ví dụ năm ở đầu đoạn hoặc được phân biệt rõ ràng), hãy trích
xuất tiêu đề đó.
Nếu không có tiêu đề rõ ràng, hãy đọc và tóm tắt nội dung đoạn văn để tạo ra một tiêu đề phù hợp, ngắn
gọn, chính xác và bao quát nội dung chính.
Đầu ra:
{
    "title": "<title>"
}
"""
```

Prompt tạo tiêu đề

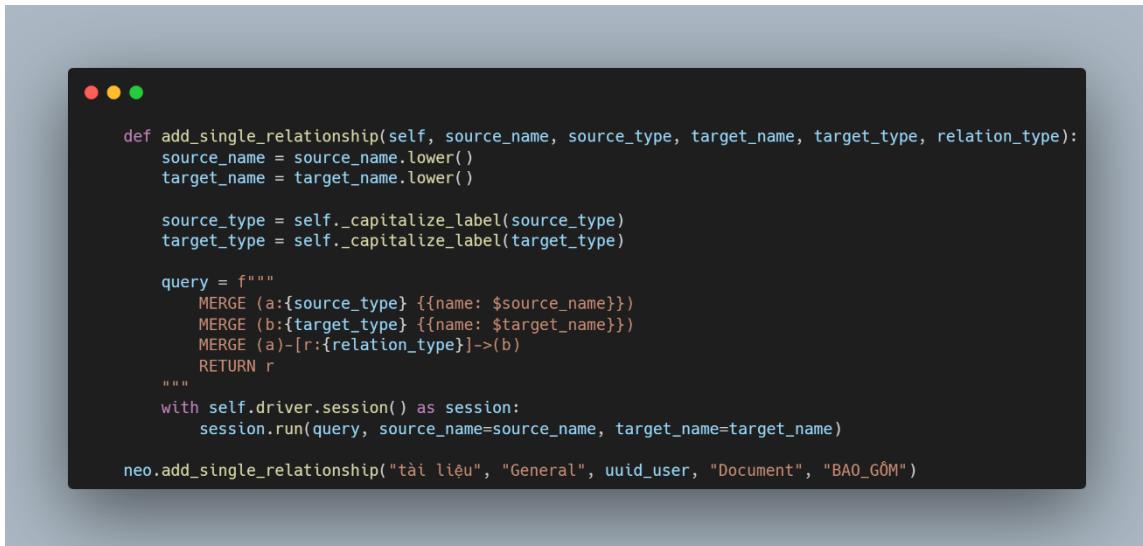
1.4.3. Lưu vào Neo4j

Đối với mỗi tiêu đề sẽ có các thực thể và quan hệ tương ứng, chúng em sẽ tiến hành lặp qua và lưu vào Neo4j.

Đầu tiên cần tạo mối quan hệ giữa node cao nhất và id của tài liệu

Tiếp theo là tạo mối quan hệ giữa id tài liệu và các tiêu đề

Tiếp theo là tạo mối quan hệ giữa các tiêu đề và thực thể, quan hệ tương ứng



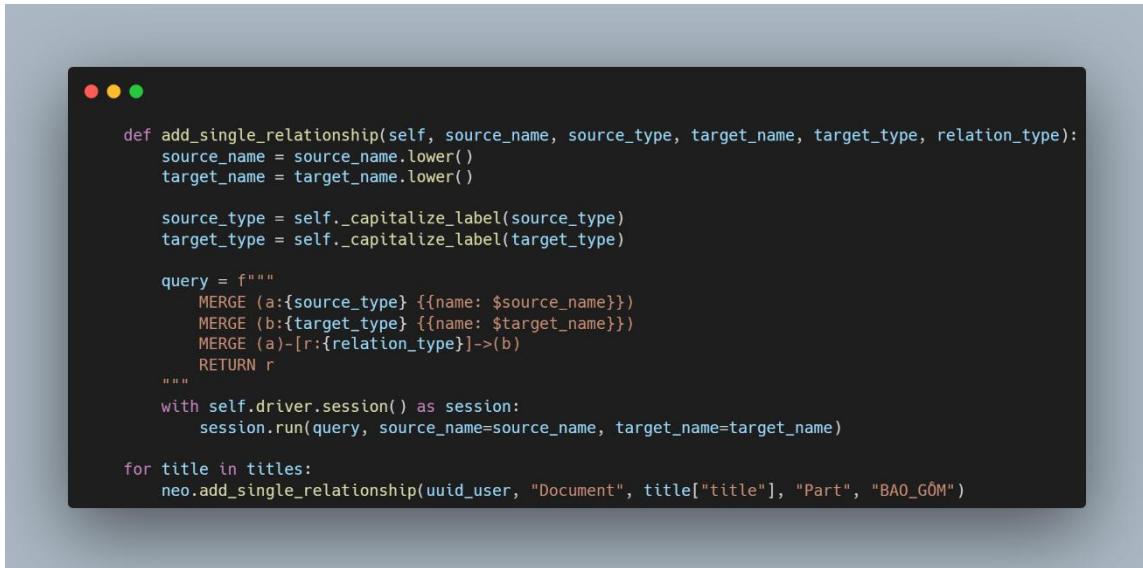
```
def add_single_relationship(self, source_name, source_type, target_name, target_type, relation_type):
    source_name = source_name.lower()
    target_name = target_name.lower()

    source_type = self._capitalize_label(source_type)
    target_type = self._capitalize_label(target_type)

    query = f"""
        MERGE (a:{source_type} {{name: $source_name}})
        MERGE (b:{target_type} {{name: $target_name}})
        MERGE (a)-[r:{relation_type}]->(b)
        RETURN r
    """
    with self.driver.session() as session:
        session.run(query, source_name=source_name, target_name=target_name)

neo.add_single_relationship("tài liệu", "General", "uuid_user", "Document", "BAO_GÔM")
```

Phương thức tạo mối quan hệ giữa node cao nhất và id của tài liệu



```
def add_single_relationship(self, source_name, source_type, target_name, target_type, relation_type):
    source_name = source_name.lower()
    target_name = target_name.lower()

    source_type = self._capitalize_label(source_type)
    target_type = self._capitalize_label(target_type)

    query = f"""
        MERGE (a:{source_type} {{name: $source_name}})
        MERGE (b:{target_type} {{name: $target_name}})
        MERGE (a)-[r:{relation_type}]->(b)
        RETURN r
    """
    with self.driver.session() as session:
        session.run(query, source_name=source_name, target_name=target_name)

    for title in titles:
        neo.add_single_relationship(uuid_user, "Document", title["title"], "Part", "BAO_GÔM")
```

Phương thức tạo mối quan hệ giữa id tài liệu và các tiêu đề



```
def import_relationships(self, content, part, type_Part):
    relationships = content["relationships"]
    with self.driver.session() as session:
        for rel in relationships:
            source = rel["source"]
            target = rel["target"]
            relation = rel["relation"].upper().replace(" ", "_").replace("-", "_")
            print(f'{source} {relation} {target}')
            type_source = self._capitalize_label(rel["type_source"])
            type_target = self._capitalize_label(rel["type_target"])
            type_Part_cap = self._capitalize_label(type_Part)

            session.execute_write(
                self._create_relation_with_Part,
                source, type_source,
                target, type_target,
                relation,
                part, type_Part_cap
            )
    for r, title in zip(entities_relationship, titles):
        neo.import_relationships(r, title["title"], "Part")
```

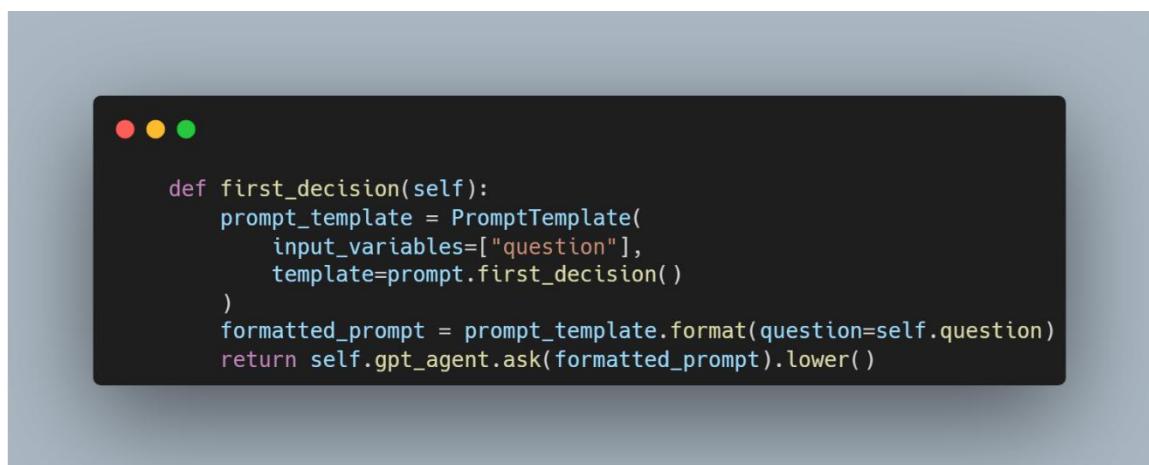
Phương thức tạo mối quan hệ giữa các tiêu đề và thực thể, quan hệ tương ứng

1.5. Xây dựng mô-đun phê bình

1.5.1. Agent

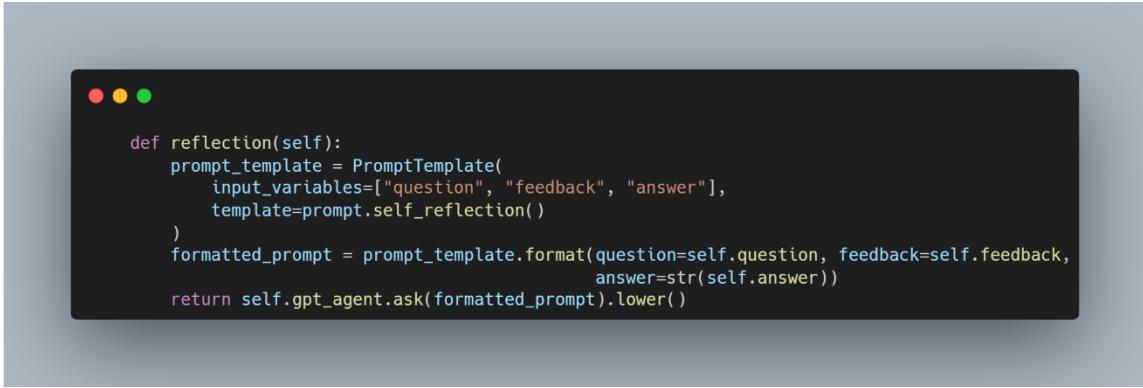
Đây là thành phần quyết định chọn nguồn truy xuất và có cơ chế tự sửa lỗi.

Khi chạy lần đầu tiên sẽ không có feedback và chỉ dự đoán nguồn truy xuất thông qua câu hỏi. Sau khi chạy lần thứ hai sẽ dự đoán nguồn dựa trên feedback



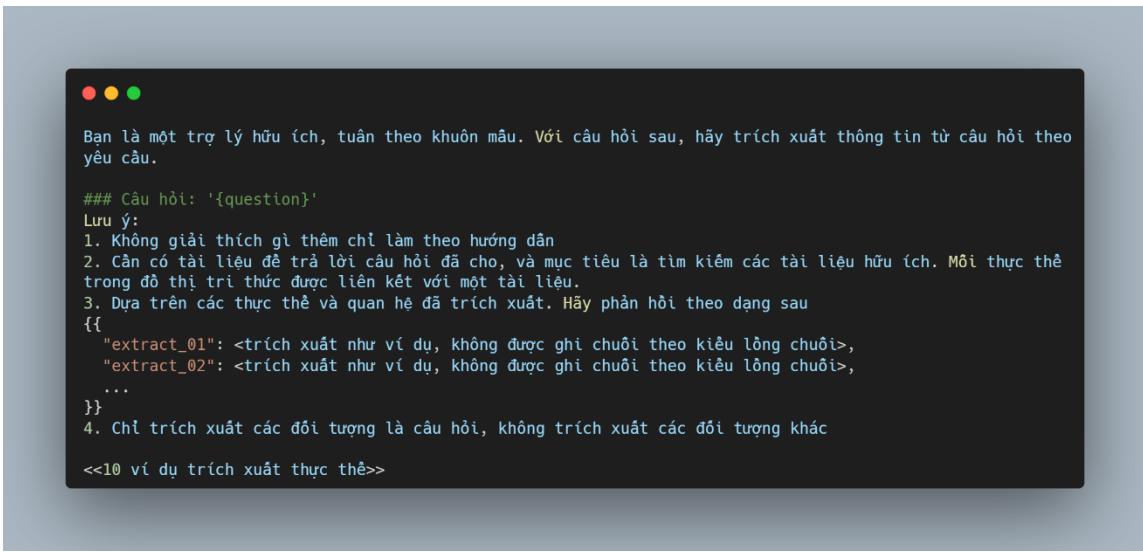
```
def first_decision(self):
    prompt_template = PromptTemplate(
        input_variables=["question"],
        template=prompt.first_decision()
    )
    formatted_prompt = prompt_template.format(question=self.question)
    return self.gpt_agent.ask(formatted_prompt).lower()
```

Phương thức trích xuất thực thể



```
def reflection(self):
    prompt_template = PromptTemplate(
        input_variables=["question", "feedback", "answer"],
        template=prompt.self_reflection()
    )
    formatted_prompt = prompt_template.format(question=self.question, feedback=self.feedback,
                                                answer=str(self.answer))
    return self.gpt_agent.ask(formatted_prompt).lower()
```

Phương thức tự sửa lỗi

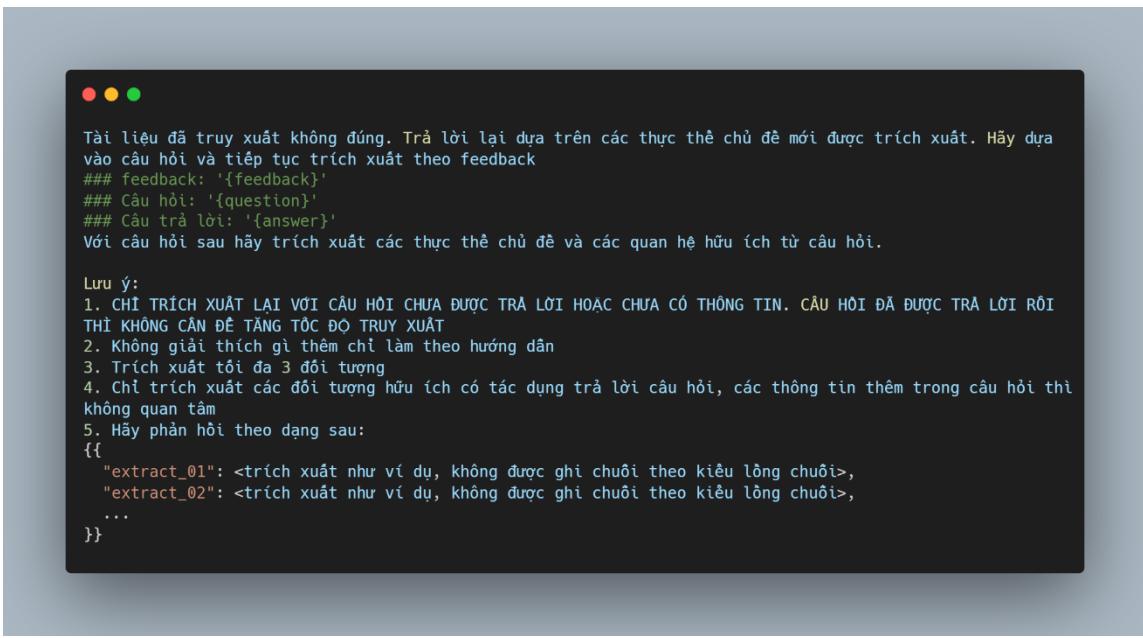


Bạn là một trợ lý hữu ích, tuân theo khuôn mẫu. Với câu hỏi sau, hãy trích xuất thông tin từ câu hỏi theo yêu cầu.

```
### Câu hỏi: '{question}'
Lưu ý:
1. Không giải thích gì thêm chỉ làm theo hướng dẫn
2. Cần có tài liệu để trả lời câu hỏi đã cho, và mục tiêu là tìm kiếm các tài liệu hữu ích. Mỗi thực thể trong đồ thị trí thức được liên kết với một tài liệu.
3. Dựa trên các thực thể và quan hệ đã trích xuất. Hãy phản hồi theo dạng sau
{{ "extract_01": <trích xuất như ví dụ, không được ghi chuỗi theo kiểu lồng chuỗi>,
  "extract_02": <trích xuất như ví dụ, không được ghi chuỗi theo kiểu lồng chuỗi>,
  ...
}}
4. Chỉ trích xuất các đối tượng là câu hỏi, không trích xuất các đối tượng khác

<<10 ví dụ trích xuất thực thể>>
```

Prompt trích xuất thực thể



Tài liệu đã truy xuất không đúng. Trả lời lại dựa trên các thực thể chủ đề mới được trích xuất. Hãy dựa vào câu hỏi và tiếp tục trích xuất theo feedback

```
### feedback: '{feedback}'
### Câu hỏi: '{question}'
### Câu trả lời: '{answer}'
Với câu hỏi sau hãy trích xuất các thực thể chủ đề và các quan hệ hữu ích từ câu hỏi.
```

Lưu ý:

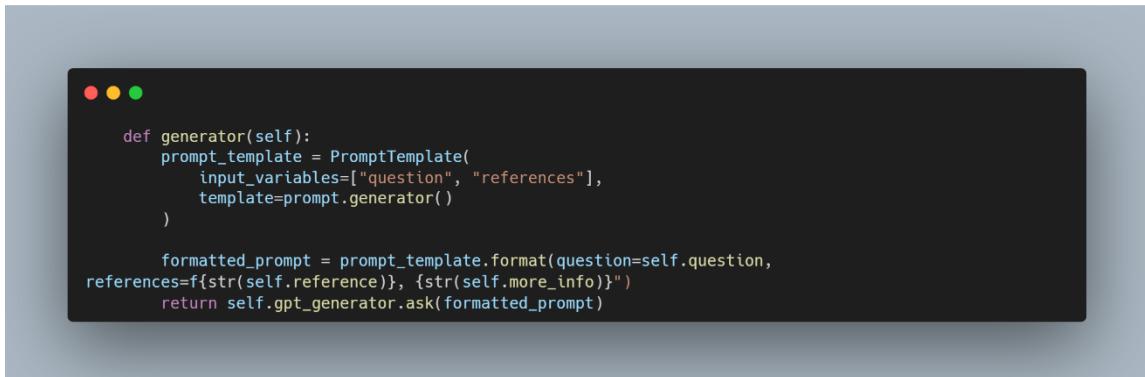
- CHỈ TRÍCH XUẤT LẠI VỚI CÂU HỎI CHƯA ĐƯỢC TRẢ LỜI HOẶC CHƯA CÓ THÔNG TIN. CÂU HỎI ĐÃ ĐƯỢC TRẢ LỜI RỒI THÌ KHÔNG CẦN ĐỀ TĂNG TỐC ĐỘ TRUY XUẤT
- Không giải thích gì thêm chỉ làm theo hướng dẫn
- Trích xuất tối đa 3 đối tượng
- Chỉ trích xuất các đối tượng hữu ích có tác dụng trả lời câu hỏi, các thông tin thêm trong câu hỏi thì không quan tâm
- Hãy phản hồi theo dạng sau:

```
{{ "extract_01": <trích xuất như ví dụ, không được ghi chuỗi theo kiểu lồng chuỗi>,
  "extract_02": <trích xuất như ví dụ, không được ghi chuỗi theo kiểu lồng chuỗi>,
  ...
}}
```

Prompt tự sửa lỗi

1.5.2. Generator

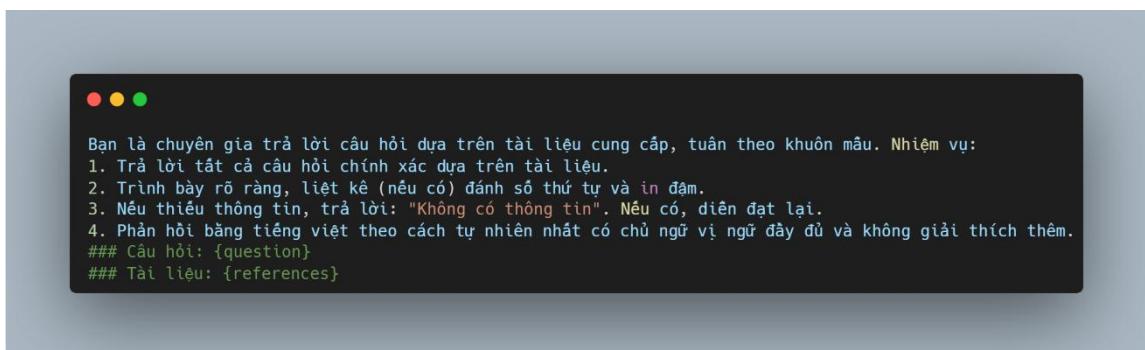
Đây là thành phần tạo câu trả lời từ tài liệu đã được truy xuất



```
def generator(self):
    prompt_template = PromptTemplate(
        input_variables=["question", "references"],
        template=prompt.generator()
    )

    formatted_prompt = prompt_template.format(question=self.question,
                                                references=f"{{str(self.reference)}}", {{str(self.more_info)}}")
    return self.gpt_generator.ask(formatted_prompt)
```

Phương thức sinh câu trả lời



Bạn là chuyên gia trả lời câu hỏi dựa trên tài liệu cung cấp, tuân theo khuôn mẫu. Nhiệm vụ:

1. Trả lời tất cả câu hỏi chính xác dựa trên tài liệu.
2. Trình bày rõ ràng, liệt kê (nếu có) đánh số thứ tự và in đậm.
3. Nếu thiếu thông tin, trả lời: "Không có thông tin". Nếu có, diên dạt lại.
4. Phản hồi bằng tiếng viết theo cách tự nhiên nhất có chủ ngữ vị ngữ đầy đủ và không giải thích thêm.

```
### Câu hỏi: {question}
### Tài liệu: {references}
```

Prompt sinh câu trả lời

1.5.3. Validator

Đây là thành phần nhiệm vụ nhị phân đưa ra kết quả “yes” hoặc “no”.

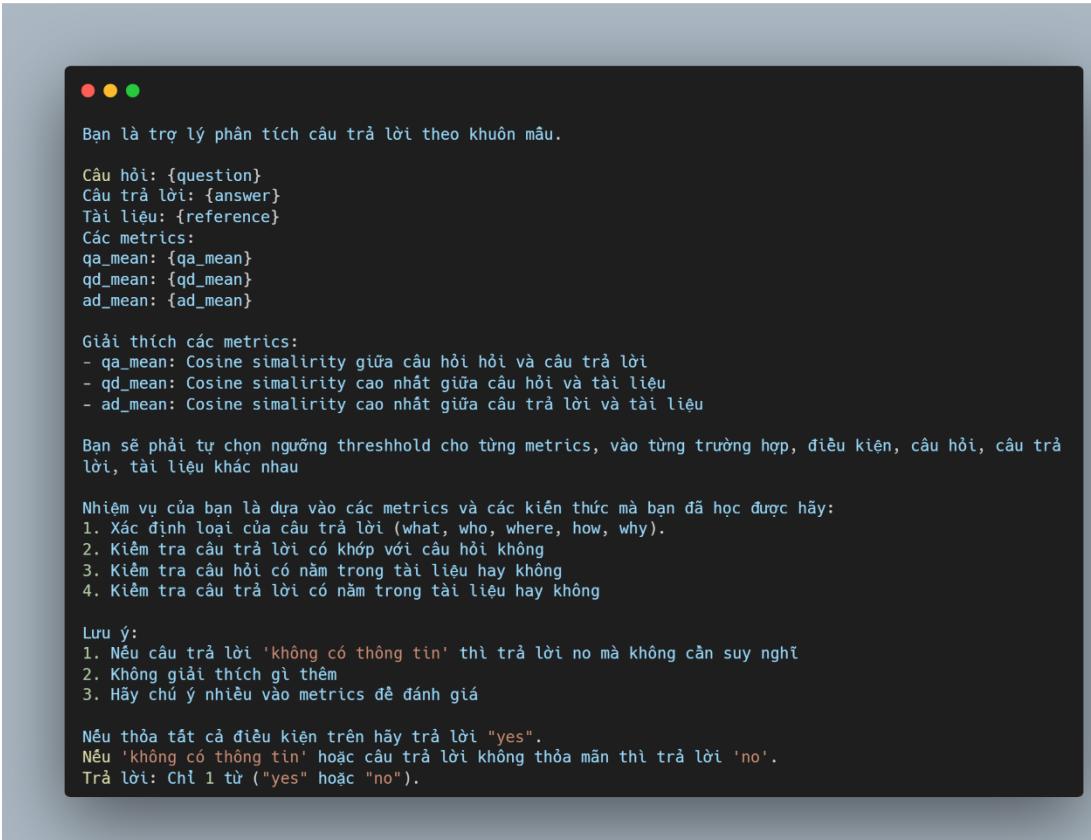


```
def valid(self):
    val_512 = self.validator_512.evaluate(self.question, self.answer, self.reference_final)
    val_768 = self.validator_768.evaluate(self.question, self.answer, self.reference_final)
    val_1024 = self.validator_1024.evaluate(self.question, self.answer, self.reference_final)
    qa_mean = (val_512['QA_similarity'] + val_768['QA_similarity'] + val_1024['QA_similarity']) / 3
    qd_mean = (val_512['Q_in_D_max'] + val_768['Q_in_D_max'] + val_1024['Q_in_D_max']) / 3
    ad_mean = (val_512['A_in_D_max'] + val_768['A_in_D_max'] + val_1024['A_in_D_max']) / 3

    prompt_template = PromptTemplate(
        input_variables=["question", "answer", "qa_mean", "reference", "qd_mean", "ad_mean"],
        template=prompt.valid_stsv()
    )
    formatted_prompt = prompt_template.format(question=self.question, answer=self.answer,
                                                reference=self.reference, qa_mean=qa_mean, qd_mean=qd_mean, ad_mean=ad_mean)

    return self.gpt_valid.ask(formatted_prompt).lower()
```

Phương thức xác nhận câu trả lời



```
Bạn là trợ lý phân tích câu trả lời theo khuôn mẫu.

Câu hỏi: {question}
Câu trả lời: {answer}
Tài liệu: {reference}
Các metrics:
qa_mean: {qa_mean}
qd_mean: {qd_mean}
ad_mean: {ad_mean}

Giải thích các metrics:
- qa_mean: Cosine similarity giữa câu hỏi hỏi và câu trả lời
- qd_mean: Cosine similarity cao nhất giữa câu hỏi và tài liệu
- ad_mean: Cosine similarity cao nhất giữa câu trả lời và tài liệu

Bạn sẽ phải tự chọn ngưỡng threshhold cho từng metrics, vào từng trường hợp, điều kiện, câu hỏi, câu trả lời, tài liệu khác nhau

Nhiệm vụ của bạn là dựa vào các metrics và các kiến thức mà bạn đã học được hãy:
1. Xác định loại của câu trả lời (what, who, where, how, why).
2. Kiểm tra câu trả lời có khớp với câu hỏi không
3. Kiểm tra câu hỏi có nằm trong tài liệu hay không
4. Kiểm tra câu trả lời có nằm trong tài liệu hay không

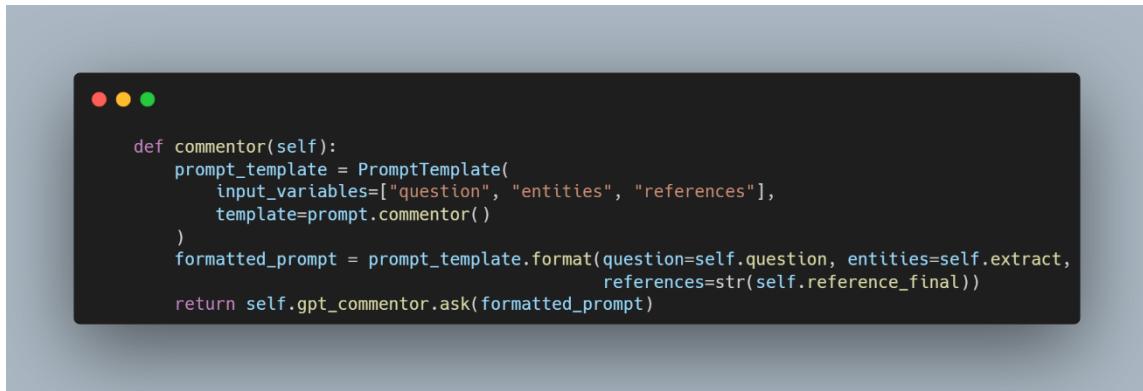
Lưu ý:
1. Nếu câu trả lời 'không có thông tin' thì trả lời no mà không cần suy nghĩ
2. Không giải thích gì thêm
3. Hãy chú ý nhiều vào metrics để đánh giá

Nếu thỏa tất cả điều kiện trên hãy trả lời "yes".
Nếu 'không có thông tin' hoặc câu trả lời không thỏa mãn thì trả lời 'no'.
Trả lời: Chỉ 1 từ ("yes" hoặc "no").
```

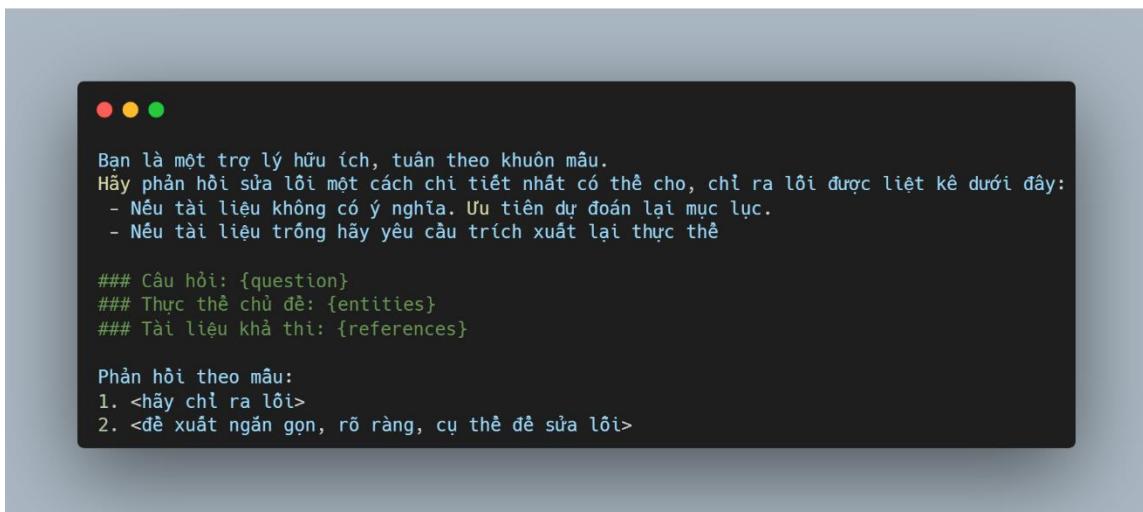
Prompt xác nhận câu trả lời

1.5.4. Comentor

Đây là thành phần sửa lỗi và phản hồi về cho Agent.



Phương thức sửa lỗi



Prompt sửa lỗi

2. Ứng dụng

2.1. Lược đồ cơ sở dữ liệu hệ thống

Bảng users

Tên	Kiểu dữ liệu	Mô tả
id	int	Khóa chính, không được null
username	varchar(50)	Tên người dùng, không được null
email	varchar(100)	Email, không được null
password	varchar(255)	Mật khẩu, không được null
created_at	timestamp	Thời gian tạo, không được null
updated_at	timestamp	Thời gian cập nhật
password_reset_token	varchar(255)	Token đặt lại mật khẩu
password_reset_token_expiry	datetime	Thời gian hết hạn token

Bảng user_documents

Tên	Kiểu dữ liệu	Mô tả
id	int	Khóa chính, không được null
user_id	int	ID người dùng, không được null
conversation_id	varchar(36)	ID cuộc trò chuyện, không được null
document_id	varchar(255)	ID tài liệu, không được null
filename	varchar(255)	Tên tệp, không được null
file_size	bigint	Kích thước tệp
status	varchar(50)	Trạng thái
s3_key	varchar(255)	Khóa S3
s3_url	varchar(255)	URL S3
created_at	timestamp	Thời gian tạo, không được null
updated_at	timestamp	Thời gian cập nhật

Bảng messages

Tên	Kiểu dữ liệu	Mô tả
id	int	Khóa chính, không được null
conversation_id	varchar(36)	ID cuộc trò chuyện, không được null
content	text	Nội dung, không được null
type	varchar(10)	Loại, không được null
created_at	timestamp	Thời gian tạo, không được null

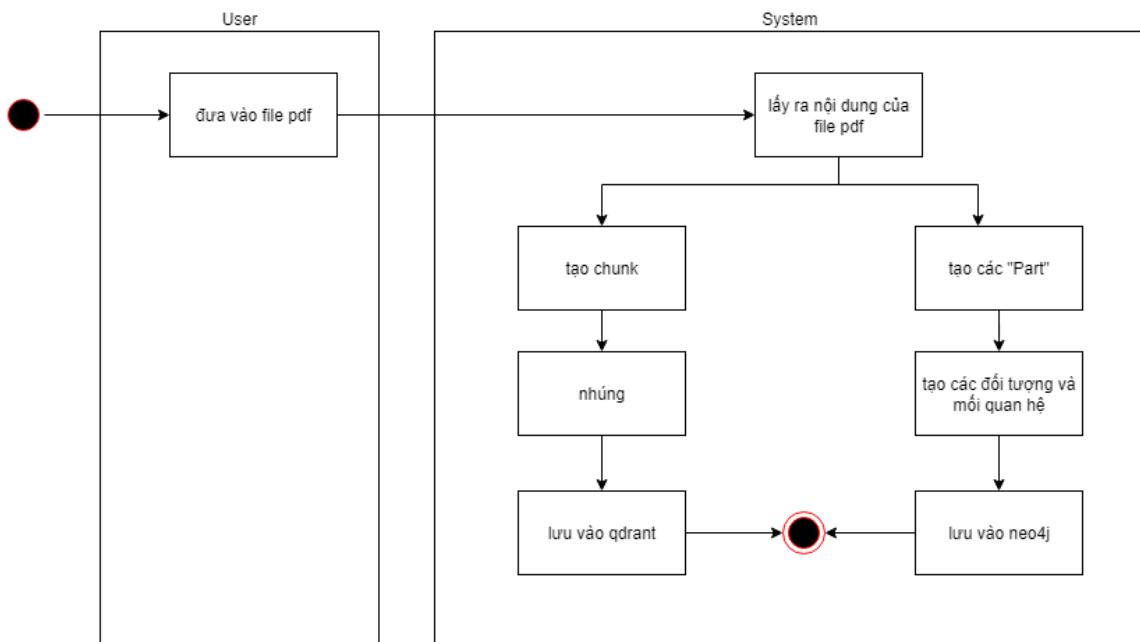
Bảng conversations

Tên	Kiểu dữ liệu	Mô tả
id	varchar(36)	Khóa chính, không được null
title	varchar(255)	Tiêu đề, không được null
user_id	int	ID người dùng, không được null
created_at	timestamp	Thời gian tạo, không được null
updated_at	timestamp	Thời gian cập nhật

Bảng contact messages

Tên	Kiểu dữ liệu	Mô tả
id	int	Khóa chính, không được null
name	varchar(100)	Tên, không được null
email	varchar(100)	Email, không được null
subject	varchar(255)	Chủ đề, không được null
message	text	Nội dung tin nhắn, không được null
created_at	timestamp	Thời gian tạo, không được null

2.2. Chức năng tạo tri thức từ tài liệu bên ngoài



Biểu đồ activity cho chức năng tạo tài liệu

```

Function process_file_upload(file, user_id, db):
    If file is missing or not a PDF:
        Raise 400 Error

    Generate unique document_id
    Read file content asynchronously

    If content is empty:
        Raise 400 Error

    Call upload_to_s3(content, filename, document_id)
    -> Return s3_key, s3_url

    Create document record in DB (status = "processing")

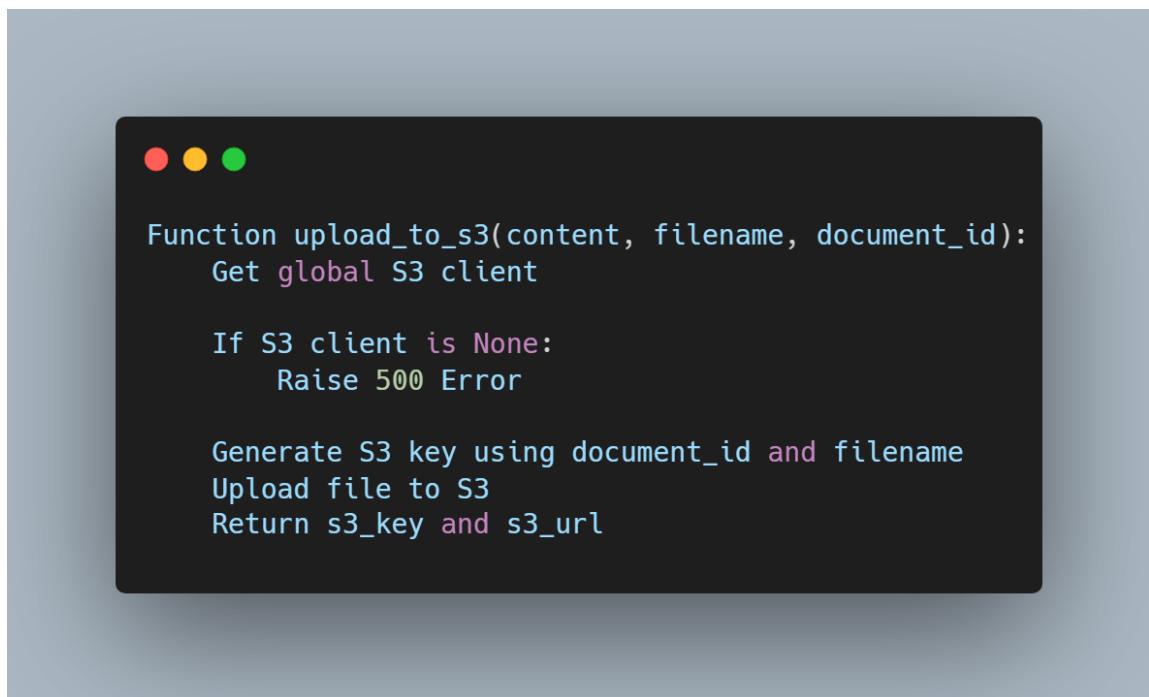
    Try:
        Call process_pdf(s3_url, s3_key, document_id) with 20 min timeout

        If success:
            Update document status to "completed"
        If timeout or error:
            Update document status to "failed"

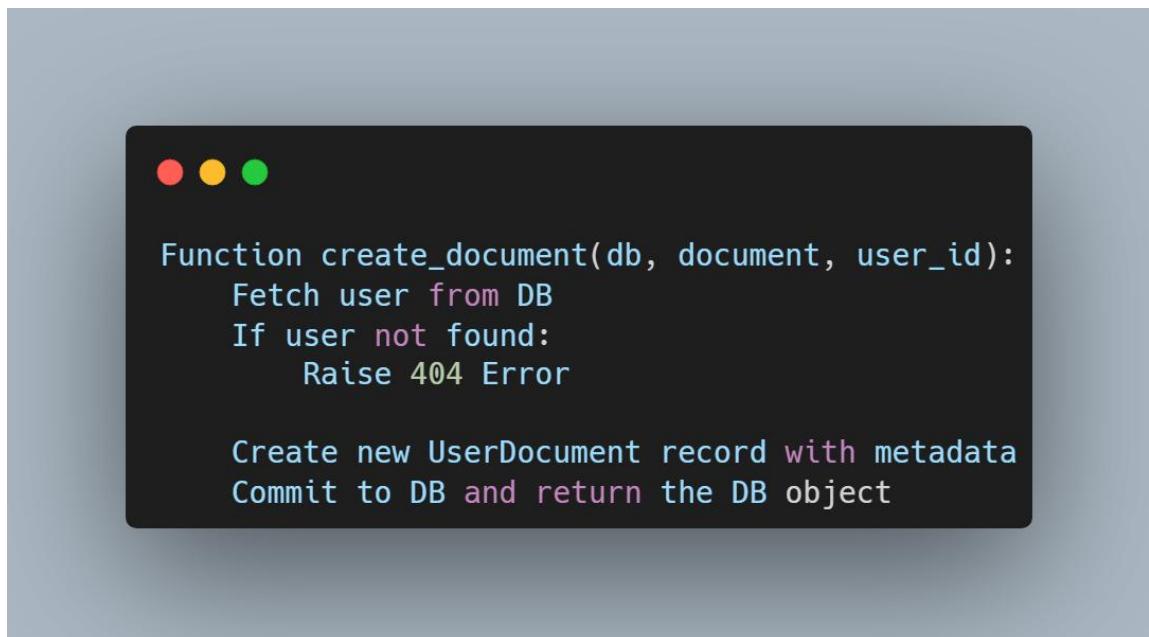
    Return metadata (document_id, filename, s3_url, final status)

```

Mã giả chức năng xử lý tài liệu mới



Mã giả chức năng lưu tài liệu lên S3



Mã giả chức năng lưu tài liệu vào cơ sở dữ liệu

```
Function process_pdf(s3_url, s3_key, document_id):
    Set PDF instance with path, bucket, key
    Call pdf.read_chunks() to extract raw text (sentences)

    Call _add_data_to_qdrant(sentences, document_id)
    Call _add_data_to_neo4j(sentences, document_id)
```

Mã giả chức năng trích xuất dữ liệu từ pdf

```
Function _add_data_to_qdrant(sentences, document_id):
    Set up llama_chunks instance with chunking prompt

    For each sentence:
        Set text in llama_chunks
        Try:
            Generate JSON chunk
            Append chunk to chunks list

    Extract all paragraph content from chunks

    Set Qdrant collection name = document_id
    Create collection in Qdrant

    Generate embeddings for paragraphs
    Add data to Qdrant
```

Mã giả chức năng lưu tài liệu vào Qdrant

```
Function _add_data_to_neo4j(sentences, document_id):
    Set up llama_title with title prompt

    For each sentence:
        Generate title (lowercase)
        Convert to JSON format
        Add to title list
        (Sleep 45s every 2 sentences)

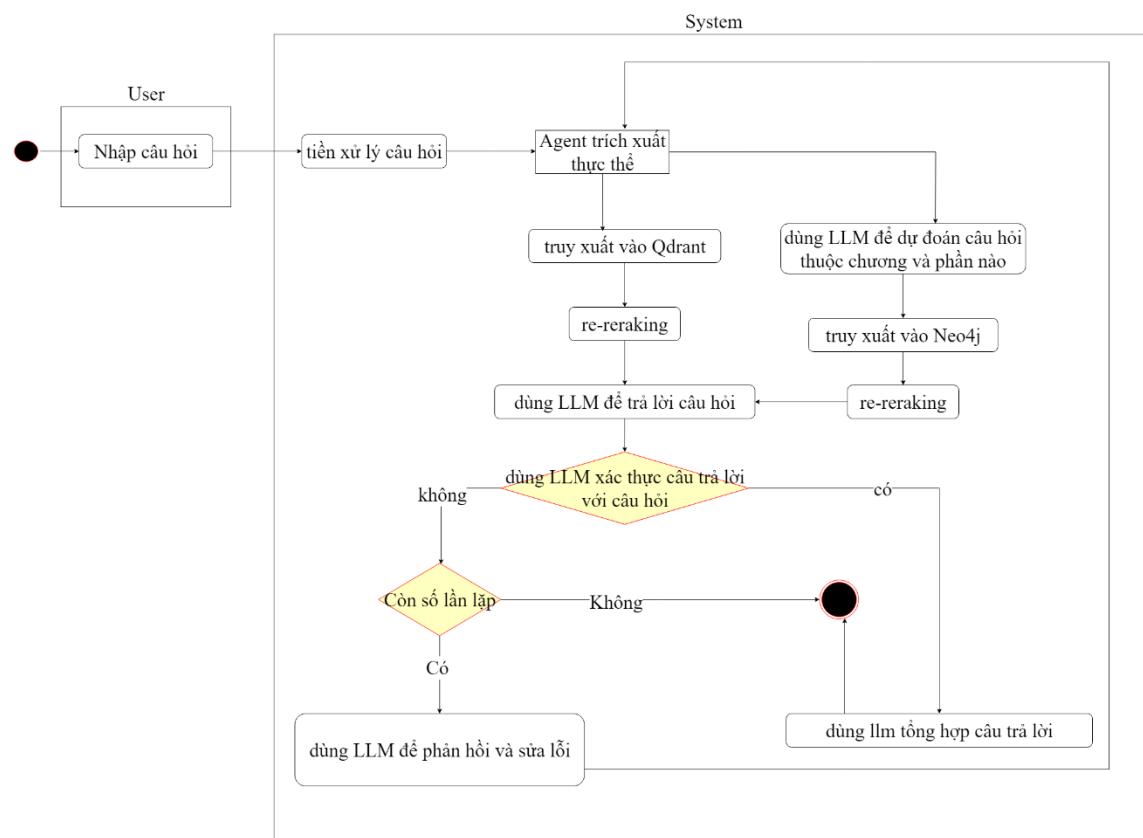
    Set up llama_content with entity extraction prompt

    For each sentence:
        Generate entities/relationships
        Convert to JSON
        Add to entity_relationship list

    In Neo4j:
        Connect "General" node to document node (BAO_GÔM)
        For each title:
            Connect document node to title node (BAO_GÔM)
        For each title + relationship:
            Import relationships into graph
```

Mã giả cho chức năng lưu tài liệu vào Neo4j

2.3. Chức năng hỏi đáp số tay sinh viên



Biểu đồ hỏi đáp số tay sinh viên



```
FUNCTION answer_s2s_stsv():
    Initialize:
        references_final ← ""
        extract ← ""
        feedback ← ""
        answer_final ← ""

    PRINT "question:", question

    FOR step t from 0 to T-1:
        PRINT "Step", t, "initial feedback:", feedback
        references_final ← ""

        IF feedback is empty THEN
            agent_response ← call first_decision()
        ELSE
            agent_response ← call reflection()

        extract ← parse agent_response to dictionary (JSON)

        PRINT "extract:", extract

        FOR each attribute, action IN extract:
            PRINT "attr:", attribute
            PRINT "action:", action

            IF action CONTAINS "graph":
                references ← call retrieval_graph_stsv()
            ELSE:
                references ← call retrieval_text()

            PRINT "Available references:", references
            references_final ← references_final + string(references)

        formatted_prompt ← fill prompt.generator_stsv() with question and references_final
        generated_answer ← call gemini_generator with formatted_prompt
        answer_final ← answer_final + "Step " + t + ": " + generated_answer

        PRINT "Answer:", answer_final

        formatted_valid_prompt ← fill prompt.valid_stsv() with question and answer_final
        validator ← call gemini_valid with formatted_valid_prompt, convert to lowercase

        PRINT "valid:", validator

        IF validator CONTAINS "yes":
            RETURN call summary_answer()

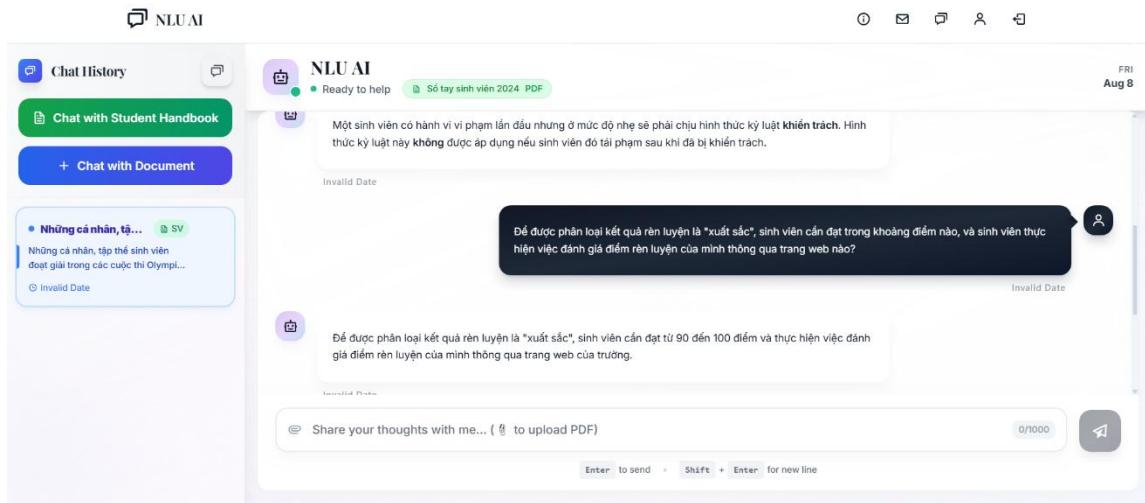
        formatted_commentor_prompt ← fill prompt.commentor_stsv() with question, extract,
        references_final
        feedback ← call gemini_commentor with formatted_commentor_prompt

    RETURN call summary_answer()

FUNCTION summary_answer():
    formatted_summary_prompt ← fill prompt.summary_answer_user() with question and answer_final
    RETURN call gemini_agent with formatted_summary_prompt
```

Mã giả hỏi đáp số tay sinh viên

2.4. Màn hình ứng dụng website



Màn hình hỏi đáp số tay sinh viên