ĐẠI HỌC QUỐC GIA
TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

CỘNG HÒA XÃ HỘI CHỦ NGHĨA
VIỆT NAM
Độc Lập - Tự Do - Hạnh Phúc

ĐỀ CƯƠNG CHI TIẾT

TÊN ĐỀ TÀI TIẾNG VIỆT: Nghiên cứu và phát triển mô hình truy xuất tăng cường tri thức có nhận biết thời gian (Temporal-Aware Graph-RAG) cho tài liệu UIT (UITGraph)

TÊN ĐỀ TÀI TIẾNG ANH: Research and Development of a Temporal-Aware Graph-RAG Retrieval Model for UIT Documents (UITGraph)

Cán bộ hướng dẫn:
ThS Phạm Nguyễn Phúc Toàn
ThS Lưu Thanh Sơn

Thời gian thực hiện: Từ ngày..... đến ngày.....

Sinh viên thực hiện:
Đặng Trần Long - 22520805
Hoàng Bảo Long - 22520807

MỤC LỤC

MỤC LỤC .....
DANH MỤC HÌNH .....
DANH MỤC BẢNG .....
CHƯƠNG 1. Giới thiệu đề tài **1**
1.1 Bối cảnh và Động lực 1
1.2 Vấn đề nghiên cứu 1
1.2.1 Thiếu khả năng suy luận đa bước (multi-hop reasoning) 1
1.2.2 Không nắm bắt được mối quan hệ giữa các thực thể 1
1.2.3 Hiệu suất kém với câu hỏi yêu cầu toàn cục (global query) 1
1.2.4 Thiếu khả năng quản lý tài liệu theo thời gian 2
1.3 Bài toán trung tâm 2
CHƯƠNG 2. Nghiên cứu liên quan **3**
2.1 Tổng quan 3
2.2 Graph-Enhanced RAG 3
2.2.1 Microsoft GraphRAG 3
2.2.2 LightRAG 3
2.2.3 HippoRAG 4
2.3 Agentic Workflow Orchestration với LangGraph 4
2.4 Temporal/Version-aware RAG 4
2.4.1 T-GRAG 4
2.4.2 VersionRAG 5
2.5 Các hệ thống RAG trong miền dữ liệu giáo dục đại học tại Việt Nam . . . 5
2.5.1 URAG – Đại học Bách Khoa TP.HCM 5
2.5.2 REBot – Đại học Cần Thơ 5
2.6 Phân tích so sánh và Lựa chọn giải pháp 6
CHƯƠNG 3. Bài toán **7**
3.1 Đầu vào và đầu ra 7
3.1.1 Đầu vào 7
3.1.2 Đầu ra 7
3.2 Mô hình hóa bài toán 8
3.2.1 Pha lập chỉ mục (offline) 8
3.2.2 Pha truy vấn (online) 8
3.3 Hướng tiếp cận đề xuất 10
CHƯƠNG 4. Mục tiêu, đối tượng và phạm vi nghiên cứu **11**
4.1 Mục tiêu nghiên cứu 11
4.2 Đối tượng nghiên cứu 11

4.3 Phạm vi nghiên cứu 11
CHƯƠNG 5. Phương pháp thực hiện 13
5.1 Tổng quan phương pháp 13
5.2 Thu thập dữ liệu và chuẩn hóa tài liệu 14
5.3 Pha lập chỉ mục ngoại tuyến (Offline Indexing) 14
5.3.1 Điều phối quy trình và quản lý trạng thái bằng LangGraph 15
5.3.2 Xây dựng đồ thị tri thức và kho lưu trữ Hybrid 16
5.3.3 Metadata RAG Subgraph - Trích xuất metadata thời gian thông minh 17
5.3.4 Cơ chế đồng bộ dữ liệu và định danh (Data Synchronization) 19
5.4 Pha truy vấn trực tuyến (Online Retrieval & Generation) 20
5.4.1 Hiểu truy vấn và cơ chế định tuyến ngữ nghĩa 21
5.4.2 Chiến lược Truy xuất kép và Xếp hạng lại 21
5.4.3 Thuật toán xếp hạng đa nhân tố (Temporal & Cohort Scoring) . . 22
5.4.4 Kiểm định chất lượng và Sinh câu trả lời 23
5.5 Triển khai hệ thống và cấu hình vận hành 23
5.5.1 Phân nhóm dịch vụ (Service Orchestration) 24
5.5.2 Thông số cấu hình và Bảo toàn dữ liệu 24
5.5.3 Lựa chọn Model (Model Choice) 25
5.6 Thiết kế thực nghiệm và Phương pháp đánh giá 27
5.6.1 Bộ chỉ số đánh giá 27
5.6.2 Quy trình thực nghiệm 28
CHƯƠNG 6. Kết quả đạt được và Hướng phát triển 30
6.1 Kết quả chức năng (Functional Outcomes) 30
6.2 Các chỉ số kỳ vọng 30

DANH SÁCH HÌNH VẼ

Hình 5.1: Tổng quan kiến trúc hệ thống UITGraph và các lớp xử lý. 13
Hình 5.2: Trình tự xử lý trong luồng lập chỉ mục (Indexing Flow) 15
Hình 5.3: Quy trình xây dựng đồ thị tri thức và các kho lưu trữ phục vụ truy xuất 16
Hình 5.4: Kiến trúc Metadata RAG 17
Hình 5.5: Kiến trúc ba tác tử trong pha truy vấn của UITGraph 20
Hình 5.6: Kiến trúc triển khai hệ thống theo Docker Compose 24

DANH SÁCH BẢNG

Bảng 2.1: So sánh các hướng tiếp cận RAG hiện nay 6

Bảng 5.1: So sánh hiệu quả thực nghiệm các phương pháp trích xuất metadata . . 19
Bảng 5.2: Đặc tả cấu hình mạng và lưu trữ của hệ thống 25

Bảng 6.1: Bảng mục tiêu kỳ vọng 30

CHƯƠNG 1: GIỚI THIỆU ĐỀ TÀI

1.1 Bối cảnh và Động lực

Bối cảnh chuyển đổi số tại Trường Đại học Công nghệ Thông tin - ĐHQG - HCM (UIT) kéo theo nhu cầu cung cấp thông tin chính xác, kịp thời và nhất quán cho các bên liên quan: sinh viên, giảng viên, cố vấn học tập, các phòng/ban chức năng. Thực tế, tri thức vận hành của UIT được phân tán trên nhiều hệ thống: website trường và các đơn vị, cổng thông tin, LMS, email nội bộ, văn bản/quy định, cùng các kênh truyền thông.

Sự phân mảnh này khiến việc truy vấn thông tin thường mất thời gian, khó đồng bộ, khó truy vết nguồn, đồng thời làm tăng tải cho các bộ phận hỗ trợ. Trong bối cảnh đó, cách tiếp cận **Retrieval-Augmented Generation (RAG)** cho phép hệ thống vừa truy xuất tài liệu liên quan, vừa tổng hợp câu trả lời theo ngữ cảnh.

1.2 Vấn đề nghiên cứu

Tuy vậy, RAG truyền thống (dựa chủ yếu vào tìm kiếm tương tự vector) bộc lộ hạn chế khi áp dụng cho môi trường đại học:

1.2.1 Thiếu khả năng suy luận đa bước (multi-hop reasoning)

Các câu hỏi như *"Tôi là sinh viên năm 3 ngành CNTT, muốn xin học bổng và đăng ký thực tập, cần điều kiện gì?"* đòi hỏi kết nối thông tin từ nhiều nguồn khác nhau (quy định học bổng, yêu cầu thực tập, quy trình thực tập, điều kiện theo từng năm học).

1.2.2 Không nắm bắt được mối quan hệ giữa các thực thể

Mối quan hệ giữa khoa/viện, chương trình đào tạo, học phần, giảng viên, và các quy định liên quan thường bị bỏ qua trong các hệ thống chỉ dựa vào biểu diễn vector (vector embeddings).

1.2.3 Hiệu suất kém với câu hỏi yêu cầu toàn cục (global query)

Câu hỏi có dạng *"Có những loại học bổng nào dành cho sinh viên năm 3?"* cần tổng hợp thông tin từ toàn bộ kho tri thức (knowledge base), không chỉ truy xuất các đoạn văn bản tương tự.

1.2.4 Thiếu khả năng quản lý tài liệu theo thời gian

Đây là vấn đề quan trọng nhất trong bối cảnh đại học: Văn bản quy định, quy chế có tính thời gian rõ ràng:

- **Hiệu lực:** Quyết định có ngày bắt đầu và ngày hết hiệu lực
- **Sửa đổi/Bổ sung:** Quy định mới thay thế hoặc sửa đổi quy định cũ
- **Áp dụng theo cohort:** Quy định khác nhau cho từng khóa sinh viên

Ví dụ thực tế:

- Quyết định 108/QĐ-ĐHCNTT (2019) sửa đổi Quyết định 141/QĐ-ĐHCNTT (2018)
- Quy chế đào tạo 2024 thay thế Quy chế 2020
- Học phí năm 2025 chỉ áp dụng cho sinh viên khóa 2025-2032

Các hệ thống RAG hiện tại **không có cơ chế** để:

1. Phân biệt tài liệu còn hiệu lực vs hết hạn
2. Ưu tiên tài liệu mới khi retrieval
3. Truy vết quan hệ sửa đổi giữa các văn bản
4. Lọc tài liệu theo cohort sinh viên

1.3 Bài toán trung tâm

Làm thế nào để xây dựng một hệ thống hỏi–đáp cho tri thức của UIT vừa truy xuất đúng chỗ, vừa tổng hợp có căn cứ, đồng thời:

- Giải thích được nguồn gốc thông tin
- Vận hành hiệu quả (độ trễ, chi phí)
- **Quản lý được tính thời gian của tài liệu** (quy chế, lịch học, học phần, biểu mẫu...)
- **Đảm bảo trả về thông tin còn hiệu lực** và phù hợp với cohort người dùng

Vì vậy, đề tài tập trung nghiên cứu mô hình **UITGraph: Temporal-Aware Graph-RAG cho tài liệu UIT**, kết hợp truy xuất dựa trên cấu trúc đồ thị với cơ chế Temporal Awareness, nhằm nâng cao độ tin cậy của câu trả lời trong môi trường tri thức động.

CHƯƠNG 2: NGHIÊN CỨU LIÊN QUAN

2.1 Tổng quan

Retrieval-Augmented Generation (RAG) được nghiên cứu rộng rãi trong các hệ thống hỏi–đáp dựa trên tài liệu, đặc biệt trong bối cảnh giáo dục và quản trị tri thức tổ chức [1]. Nhóm công trình liên quan có thể quy về ba mạch kỹ thuật:

1. RAG tăng cường bởi đồ thị (Graph-enhanced RAG) để khai thác quan hệ thực thể và hỗ trợ truy vấn toàn cục;
2. Quy trình tác tử (agentic) để điều phối pipeline nhiều bước, kiểm soát trạng thái và chất lượng;
3. Temporal/Version-aware RAG để xử lý tri thức biến đổi theo thời gian.

Tuy nhiên, khi áp dụng vào miền dữ liệu học vụ, mục tiêu không chỉ là truy xuất "đúng chủ đề" mà phải truy xuất "đúng hiệu lực, đúng đối tượng áp dụng", đồng thời truy vết được văn bản sửa đổi/bổ sung – đây là ràng buộc mà nhiều hướng tiếp cận hiện tại chưa đáp ứng đầy đủ.

2.2 Graph-Enhanced RAG

2.2.1 Microsoft GraphRAG

Microsoft GraphRAG [2] tích hợp đồ thị tri thức vào pipeline RAG bằng cách trích xuất thực thể–quan hệ, sau đó tổ chức tri thức theo các cộng đồng (communities) bằng community detection để hỗ trợ truy vấn toàn cục (global queries) cần tổng hợp từ nhiều vùng tri thức. Độ bao phủ của đồ thị sẽ phụ thuộc mạnh vào độ dài ngữ cảnh của Large Language Model (LLM) được sử dụng.

Mặc dù mạnh mẽ về cấu trúc, Microsoft GraphRAG tốn rất nhiều chi phí để vận hành và sử dụng, điểm yếu của Microsoft GraphRAG còn nằm trong miền dữ liệu học vụ nằm ở pha lập chỉ mục/truy vấn, đồng thời không xử lý trực tiếp các ràng buộc hiệu lực văn bản, quan hệ sửa đổi và phạm vi áp dụng theo khóa sinh viên.

2.2.2 LightRAG

LightRAG [3] hướng tới giảm chi phí của GraphRAG bằng kiến trúc truy xuất kép (dual-level retrieval): truy xuất mức thấp theo entities/relationships phục vụ truy vấn cụ thể và truy xuất mức cao theo community summaries phục vụ truy vấn tổng quan. Cách thiết kế này giúp giảm tải token và cải thiện khả năng tổng hợp ngữ cảnh.

Thách thức lớn nhất ngăn cản LightRAG ứng dụng vào bối cảnh này là thiếu lớp suy luận theo "hiệu lực" và "đối tượng áp dụng", nên vẫn có nguy cơ ưu tiên các đoạn đúng ngữ nghĩa nhưng sai phiên bản hoặc sai thời điểm.

2.2.3 HippoRAG

HippoRAG [4] mô phỏng cơ chế bộ nhớ dài hạn và dùng Personalized PageRank (PPR) để xếp hạng tri thức theo cấu trúc liên kết, từ đó tăng khả năng kết nối ngữ cảnh và gợi ý multi-hop. Dù vậy, rào cản chính của HippoRAG nằm ở độ phức tạp khi triển khai thực tế, đồng thời thiếu vắng cơ chế chỉ ra rõ cách ràng buộc hiệu lực/phiên bản để tránh trả lời dựa trên văn bản đã hết hiệu lực hoặc đã bị sửa đổi.

2.3 Agentic Workflow Orchestration với LangGraph

Các framework điều phối như LangGraph [5–7] hỗ trợ xây dựng pipeline dạng đồ thị trạng thái (stateful), cho phép rẽ nhánh theo điều kiện, tái sử dụng trạng thái truy vấn và kiểm soát nhiều bước xử lý (hiểu truy vấn → truy xuất → kiểm định chất lượng → tổng hợp). Về mặt thuật toán, điểm mạnh của cách tiếp cận này là:

(i) Chuẩn hóa trạng thái (schema) để theo dõi biến trung gian;

(ii) Định tuyến có điều kiện dựa trên ngưỡng confidence/quality;

(iii) Chèn checkpoint để có thể kiểm soát/human-in-the-loop khi cần.

Song, bản thân orchestration chỉ giải quyết bài toán quy trình mà bỏ ngỏ bài toán ngữ nghĩa thời gian. Nếu tầng truy xuất bên dưới không có ràng buộc temporal, hệ vẫn có thể tổng hợp từ ngữ cảnh sai phiên bản.

2.4 Temporal/Version-aware RAG

2.4.1 T-GRAG

T-GRAG [8] xử lý tri thức biến đổi theo thời gian bằng đồ thị tri thức có đánh dấu thời gian (time-stamped knowledge graph) và cơ chế phân rã truy vấn theo thời gian (temporal query decomposition), sau đó truy xuất theo nhiều lớp để trả lời câu hỏi nhạy thời gian. Dù đã tiếp cận khía cạnh thời gian, T-GRAG vẫn chưa thể giải quyết trọn vẹn bài toán, mô hình hóa "tiến hóa tri thức" chưa tương đương với quản lý "khoảng hiệu lực" (`valid_from`/`valid_until`) và "quan hệ sửa đổi/bổ sung" vốn phổ biến trong văn bản quy định.

2.4.2 VersionRAG

VersionRAG [9] tập trung vào tài liệu có chuỗi phiên bản rõ ràng (v1→v2→v3), xây dựng version graph phân cấp và phát hiện thay đổi ngầm định để định tuyến truy xuất theo phiên bản phù hợp. Điểm hạn chế của hướng tiếp cận này là sự phụ thuộc cứng nhắc vào chuỗi phiên bản (version chain), trong khi dữ liệu quy định thực tế đòi hỏi mô hình hóa trực tiếp hiệu lực và quan hệ amendment thay vì chỉ dựa vào chuỗi version.

2.5 Các hệ thống RAG trong miền dữ liệu giáo dục đại học tại Việt Nam

Trong miền dữ liệu giáo dục đại học, các hệ thống hỏi đáp (QA Systems) đang chuyển dịch mạnh mẽ từ các chatbot dựa trên quy tắc (rule-based) sang các mô hình sinh văn bản có truy xuất (RAG). Tại Việt Nam, một số nghiên cứu tiêu biểu dưới đây đã bước đầu giải quyết bài toán tư vấn tuyển sinh và quy chế đào tạo.

Tuy nhiên, điểm hạn chế chung của các hệ thống hiện có là việc phụ thuộc quá lớn vào tìm kiếm tương đồng vector (Vector Similarity Search) đơn thuần. Cách tiếp cận này thường gặp khó khăn khi đối mặt với hai thách thức đặc thù của văn bản hành chính nhà trường:

1. **Sự phân mảnh thông tin:** Câu trả lời thường nằm rải rác ở nhiều văn bản khác nhau (ví dụ: điều kiện học bổng nằm ở Quyết định A, nhưng mức tiền lại nằm ở Thông báo B), đòi hỏi khả năng suy luận đa bước.

2. **Tính biến động theo thời gian:** Các hệ thống RAG cơ bản thường truy xuất dựa trên mức độ liên quan về từ khóa mà bỏ qua trạng thái hiệu lực (validity) của văn bản, dẫn đến việc cung cấp thông tin từ các quy chế đã hết hạn hoặc bị thay thế.

2.5.1 URAG – Đại học Bách Khoa TP.HCM

URAG [10] triển khai hệ thống hybrid RAG kết hợp rule-based FAQs với retrieval-based generation để hỗ trợ tư vấn tuyển sinh tại ĐHBK TP.HCM. Hệ thống ưu tiên trả lời từ cơ sở FAQ cho các câu hỏi phổ biến/nhạy cảm, sau đó mới chuyển sang truy xuất tài liệu và sinh câu trả lời qua LLM. Tuy nhiên, URAG chưa tích hợp đồ thị tri thức và thiếu cơ chế quản lý hiệu lực theo thời gian; do đó khó hỗ trợ truy vấn multi-hop và khó đảm bảo trả lời đúng khi quy định thay đổi.

2.5.2 REBot – Đại học Cần Thơ

REBot [11] áp dụng framework CatRAG (Category-aware RAG) kết hợp dense retrieval với graph reasoning thông qua đồ thị tri thức phân cấp theo danh mục, được

làm giàu bằng đặc trưng ngữ nghĩa (semantic features). Hệ thống sử dụng intent classifier nhẹ để định tuyến truy vấn đến module truy xuất phù hợp và đạt F1-score 98.89% trên tập dữ liệu của Đại học Cần Thơ. Tuy nhiên, REBot chưa giải quyết trực tiếp bài toán quản lý hiệu lực theo thời gian và quan hệ sửa đổi văn bản, nên độ tin cậy có thể giảm khi quy định thay đổi.

⇒ Sự phụ thuộc thuần túy vào tìm kiếm vector khiến các hệ thống này dễ gặp lỗi ảo giác khi đối mặt với các văn bản sửa đổi có nội dung ngữ nghĩa tương đồng cao nhưng mang hiệu lực pháp lý đối lập.

2.6 Phân tích so sánh và Lựa chọn giải pháp

Để làm rõ vị trí và tính mới của giải pháp đề xuất, chúng tôi thực hiện so sánh các mô hình RAG tiêu biểu dựa trên các tiêu chí kỹ thuật nâng cao:

- **Cấu trúc tri thức (Knowledge Structure):** Cách hệ thống tổ chức dữ liệu (Vector phẳng hay Đồ thị có cấu trúc).
- **Suy luận đa bước (Multi-hop Reasoning):** Khả năng kết nối thông tin từ các thực thể gián tiếp để trả lời câu hỏi phức tạp.
- **Chiến lược xử lý siêu dữ liệu (Metadata Strategy):** Phương pháp hệ thống sử dụng để trích xuất và chuẩn hóa các thông tin thời gian (ngày hiệu lực, văn bản sửa đổi).

Bảng 2.1. So sánh các hướng tiếp cận RAG hiện nay

Feature	**NaiveRAG** (URAG, REBot)	**GraphRAG** (Microsoft)	**T-GRAG**	**UITGraph** (Ours)
Cấu trúc tri thức	Vector Store	Đồ thị tri thức	Đồ thị có nhãn thời gian	Đồ thị + Temporal Metadata
Multi-hop Reasoning	Không có	Cao	Trung bình	Cao (Nhờ LightRAG)
Xử lý thời gian (Temporal)	Không có	Không có	Chỉ tịnh tiến	Validity & Amendment
Cơ chế Metadata	Không xác định	Trích xuất thực thể (Entity)	Regex/ Rule-based	RAG-based Subgraph
Chi phí vận hành	Thấp	Rất cao	Trung Bình	Tối ưu

⇒ Trong khi GraphRAG mạnh về suy luận nhưng chi phí quá cao, và các giải pháp Naive RAG thiếu độ chính xác về ngữ cảnh, UITGraph lựa chọn cách tiếp cận cân bằng: sử dụng LightRAG [3] cho suy luận cấu trúc và tích hợp module chuyên biệt Metadata RAG để xử lý bài toán thời gian với chi phí thấp.

CHƯƠNG 3: BÀI TOÁN

Trong môi trường đại học, thông tin học vụ và hành chính thường tồn tại dưới dạng nhiều văn bản được công bố phân tán trên nhiều kênh thông tin. Đặc điểm quan trọng của nhóm tài liệu này là tính thay đổi theo thời gian: văn bản có thể có ngày hiệu lực/hết hiệu lực, có thể được sửa đổi hoặc bổ sung bởi văn bản khác, và có thể chỉ áp dụng cho một nhóm đối tượng nhất định (ví dụ theo khóa sinh viên). Bài toán đặt ra là xây dựng một hệ thống hỏi–đáp dựa trên tài liệu UIT có khả năng:

(i) Truy xuất đúng nguồn,

(ii) Tổng hợp câu trả lời có căn cứ (kèm trích dẫn),

(iii) Bảo đảm tính đúng đắn theo thời gian và đối tượng áp dụng tại thời điểm người dùng đặt câu hỏi.

3.1 Đầu vào và đầu ra

3.1.1 Đầu vào

Hệ thống nhận vào bộ ba (D, q, c) tại thời điểm truy vấn t_{query}.

- **Tập tài liệu** $D = \{d_1, d_2, ..., d_n\}$. Mỗi tài liệu d_i bao gồm nội dung sau xử lý (ví dụ: văn bản/markdown trích xuất từ web hoặc PDF), biểu diễn embedding để truy xuất, và siêu dữ liệu thời gian.
- **Biểu diễn hình thức:** $d_i = \langle Content_i, Embedding_i, Meta_i \rangle$.
- Trong đó, $Meta_i$ là tập thuộc tính dùng để quản lý hiệu lực và quan hệ văn bản, gồm tối thiểu: {`valid_from`, `valid_until`, `document_number`, `student_cohorts`, `amends_documents`, `amended_by`, `temporal_extraction_method`, `temporal_confidence`, `indexed_at`, `is_archived`, `archived_at`, `archive_reason`}.
- **Truy vấn** q: Câu hỏi ngôn ngữ tự nhiên của người dùng.
- **Ngữ cảnh** c: Thông tin truy vấn kèm thuộc tính khóa sinh viên K_c (tương ứng năm khóa cần áp dụng trong truy vấn).

3.1.2 Đầu ra

Hệ thống trả về bộ kết quả gồm câu trả lời và tập bằng chứng:

- **Câu trả lời** A: Văn bản tiếng Việt được sinh bởi mô hình ngôn ngữ dựa trên ngữ cảnh truy xuất.

- **Tập bằng chứng** $S \subset D$: Tập các đoạn trích (chunks) được chọn làm căn cứ cho A, kèm tham chiếu nguồn.
- **Cảnh báo hiệu lực** W: Thông báo cho người dùng nếu bằng chứng thuộc trạng thái đã hết hiệu lực, sắp hết hiệu lực, bị sửa đổi, hoặc bị lưu trữ (archived).

3.2 Mô hình hóa bài toán

Bài toán được mô hình hóa thành hai pha xử lý: pha lập chỉ mục (offline) và pha truy vấn - xếp hạng (online).

3.2.1 Pha lập chỉ mục (offline)

Mục tiêu của pha offline là ánh xạ tập tài liệu thô D thành cơ sở tri thức phục vụ truy xuất gồm: đồ thị tri thức, kho vector và kho metadata thời gian. Pha này cần đáp ứng các yêu cầu sau:

- **Xử lý nội dung đa dạng:** Hệ thống cần xử lý được nhiều nguồn dữ liệu khác nhau (web, PDF, HTML) và chuẩn hóa về định dạng thống nhất để phục vụ phân đoạn và trích xuất tri thức. Với tài liệu PDF phức tạp (chứa bảng biểu, tiêu đề phân cấp), cần bảo toàn cấu trúc layout để giữ nguyên ngữ cảnh phân cấp.
- **Trích xuất metadata thời gian với độ tin cậy cao:** Hệ thống cần tự động trích xuất các trường metadata quan trọng (`valid_from`, `valid_until`, `document_number`, `student_cohorts`) với độ chính xác cao ($\geq 90\%$). Cơ chế trích xuất phải có khả năng xử lý các quan hệ thời gian ẩn (ví dụ: "có hiệu lực sau 45 ngày kể từ ngày ký") mà các phương pháp khớp mẫu đơn giản không thể xử lý.
- **Liên kết quan hệ sửa đổi:** Hệ thống cần tự động nhận diện và thiết lập quan hệ giữa các văn bản khi tài liệu d_i sửa đổi/bổ sung văn bản khác (`amends_documents`), và ngược lại (`amended_by`), để hỗ trợ truy vết lịch sử thay đổi quy định.
- **Xây dựng cấu trúc tri thức đa chiều:** Tích hợp ba lớp lưu trữ bổ trợ nhau: đồ thị tri thức (hỗ trợ suy luận quan hệ), vector embeddings (tìm kiếm ngữ nghĩa), và cơ sở dữ liệu quan hệ (quản lý metadata thời gian với khả năng lọc chính xác).

3.2.2 Pha truy vấn (online)

Trong pha online, hệ thống truy xuất một tập ứng viên từ D và thực hiện xếp hạng dựa trên điểm ngữ nghĩa, điểm thời gian và hệ số tăng cường theo khóa sinh viên.

Bài toán lựa chọn tập bằng chứng tối ưu S^* được mô hình hóa như sau:

$$S^* = \arg\max_{S \subset D} Score(S \mid q, t_{query}, c) \tag{3.1}$$

Trong đó, điểm tổng hợp cho mỗi tài liệu ứng viên d_i được tính theo công thức:

$$final\_score(d_i) = [(1 - w) \cdot semantic\_score + w \cdot temporal\_score] \times cohort\_boost \tag{3.2}$$

Với w là trọng số cân bằng giữa ngữ nghĩa và thời gian (mặc định $w = 0.3$, có thể cấu hình). Các thành phần được mô tả như sau:

Điểm ngữ nghĩa (semantic_score)

Điểm tương đồng giữa truy vấn q và tài liệu ứng viên, được lấy từ tín hiệu truy xuất theo vector và xếp hạng theo đồ thị (Vector Similarity + Graph Ranking).

Hệ số tăng cường theo khóa sinh viên (cohort_boost)

Hệ số nhân ưu tiên tài liệu có danh sách khóa áp dụng phù hợp với khóa sinh viên trích từ ngữ cảnh c.

- Nếu thiếu khóa truy vấn hoặc tài liệu không có danh sách khóa: $cohort\_boost = 1.0$.
- Khớp chính xác ($query\_cohort \in document\_cohorts$): $cohort\_boost = 1.5$.
- Khớp gần (trong khoảng ± 3 năm so với một khóa bất kỳ của tài liệu): $cohort\_boost = 1.2$.
- Các trường hợp còn lại: $cohort\_boost = 1.0$.

Điểm thời gian (temporal_score)

Điểm này nằm trong khoảng $[0, 1]$, phản ánh đồng thời trạng thái hiệu lực và độ mới của tài liệu. Điểm thời gian được tính theo hai bước:

(i) Tính điểm cơ sở theo hàm `calculate_temporal_score`,

(ii) Hiệu chỉnh bởi các hệ số phạt chất lượng.

Bước 1 (Temporal base): $temporal\_base = calculate\_temporal\_score(d_i, t_{query})$. Hàm `calculate_temporal_score` được mô tả theo các trường hợp:

- Nếu $is\_archived = True$: $temporal\_base = 0.0$.
- Nếu tài liệu đã hết hiệu lực tại t_{query}:
  - Nếu $days\_expired > 365$: $temporal\_base = 0.1$.
  - Ngược lại ($0 < days\_expired \leq 365$): $temporal\_base = 0.5 - \frac{days\_expired}{365} \times 0.4$.
- Nếu tài liệu có thuộc tính `amended_by` (đã bị văn bản khác sửa đổi/bổ sung): $temporal\_base = 0.3$.

- Nếu tài liệu còn hiệu lực (valid) thì áp dụng suy giảm theo thời gian lập chỉ mục `indexed_at` với $days\_old$:
  - Nếu $days\_old \leq 30$: $temporal\_base = 1.0$.
  - Nếu $30 < days\_old \leq 365$: $temporal\_base = 0.9 - \frac{days\_old - 30}{365} \times 0.2$ (giảm về 0.7).
  - Nếu $days\_old > 365$: $temporal\_base = \max(0.5, 0.7 - \frac{days\_old - 365}{365} \times 0.2)$.

Bước 2 (Quality penalties): $temporal\_score = temporal\_base \times penalty\_factor$, trong đó:

- Nếu tài liệu đã hết hiệu lực (expired): $penalty\_factor = 0.5$.
- Nếu tài liệu sắp hết hiệu lực trong < 30 ngày (expiring soon): $penalty\_factor = 0.8$.
- Các trường hợp còn lại: $penalty\_factor = 1.0$.

Cuối cùng, hệ thống sắp xếp các ứng viên theo $final\_score$ giảm dần và chọn các tài liệu có điểm cao nhất làm tập bằng chứng trả về.

3.3 Hướng tiếp cận đề xuất

Để giải quyết bài toán đã mô hình hóa ở trên, đề tài đề xuất hệ thống **UITGraph: Temporal-Aware Graph-RAG** với ba trụ cột kỹ thuật chính:

1. **Graph-enhanced Retrieval (LightRAG):** Sử dụng kiến trúc truy xuất kép (dual-level retrieval) kết hợp entity-level và community-level để hỗ trợ cả truy vấn cụ thể (specific queries) lẫn truy vấn toàn cục (global queries), đồng thời giảm chi phí so với GraphRAG truyền thống.

2. **Metadata RAG Subgraph:** Đề xuất cơ chế trích xuất metadata thời gian dựa trên RAG chuyên biệt (specialized mini-RAG), coi việc xác định siêu dữ liệu thời gian là một bài toán Question Answering ngược vào chính tài liệu đó. Cách tiếp cận này vượt trội hơn Regex hoặc LLM Prompting đơn thuần trong việc xử lý các quan hệ thời gian phức tạp.

3. **Multi-Agent Orchestration (LangGraph):** Áp dụng kiến trúc đa tác tử có trạng thái với ba agent chuyên biệt (Query Understanding, Quality Assessment, Response Generation), cho phép định tuyến linh hoạt và cơ chế fallback để đảm bảo tính an toàn của câu trả lời.

Chi tiết kỹ thuật về kiến trúc hệ thống, quy trình lập chỉ mục, thuật toán xếp hạng và triển khai được trình bày đầy đủ tại Chương 5.

CHƯƠNG 4: MỤC TIÊU, ĐỐI TƯỢNG VÀ PHẠM VI NGHIÊN CỨU

4.1 Mục tiêu nghiên cứu

Mục tiêu của đề tài là xây dựng khung giải pháp **Temporal-Aware Graph-RAG (UITGraph)** cho bài toán hỏi - đáp trên tài liệu UIT, nhằm bảo đảm truy xuất đúng nguồn và đúng hiệu lực văn bản tại thời điểm truy vấn.

Các mục tiêu kỹ thuật của hệ thống UITGraph gồm:

- Biểu diễn tri thức dạng đồ thị và truy xuất theo quan hệ để hỗ trợ truy vấn toàn cục và suy luận đa bước.
- Điều phối quy trình hỏi - đáp theo mô hình tác tử có trạng thái, rẽ nhánh theo ngưỡng độ tin cậy và chất lượng ngữ cảnh.
- Quản lý tài liệu theo thời gian ở mức chi tiết: trích xuất khoảng hiệu lực, nhận diện quan hệ sửa đổi/bổ sung và hỗ trợ lọc theo khóa sinh viên (cohort).
- Xây dựng và đánh giá thực nghiệm hệ thống UITGraph về độ chính xác (Accuracy), khả năng truy hồi (Recall) và thời gian phản hồi (Latency) so với các phương pháp RAG cơ bản.

4.2 Đối tượng nghiên cứu

Đối tượng nghiên cứu là quy trình truy xuất - xếp hạng - tổng hợp tri thức trên tập tài liệu UIT, trong đó trọng tâm gồm:

(i) Cơ chế xây dựng và khai thác cấu trúc đồ thị tri thức (LightRAG),

(ii) Cơ chế điều phối tác tử (LangGraph) cho pipeline nhiều bước, và

(iii) Cơ chế mô hình hóa - khai thác metadata thời gian để đảm bảo tính đúng đắn theo hiệu lực văn bản.

4.3 Phạm vi nghiên cứu

Phạm vi triển khai tập trung vào các nguồn tài liệu chính thống phục vụ tra cứu học vụ/hành chính (website, văn bản PDF/HTML, thông báo), được thu thập bằng crawler và/hoặc tải lên thủ công. Hệ thống hướng đến trả lời tiếng Việt, có trích dẫn nguồn và cảnh báo khi văn bản liên quan đã hết hiệu lực hoặc đã bị sửa đổi/bổ sung.

Các nội dung ngoài phạm vi gồm:

(i) Tri thức ngoài tập tài liệu đã thu thập;

(ii) Xử lý thông tin cá nhân nhạy cảm;

(iii) Cập nhật thời gian thực từ các nguồn không kiểm soát.

CHƯƠNG 5: PHƯƠNG PHÁP THỰC HIỆN

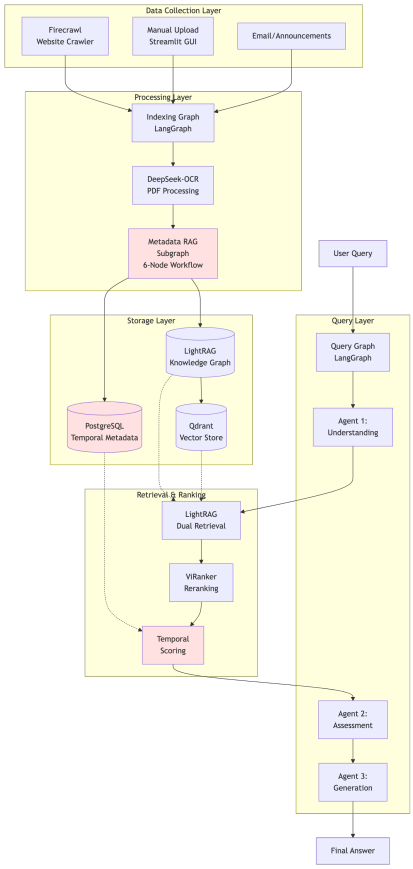
5.1 Tổng quan phương pháp

Hệ thống UITGraph được xây dựng và triển khai theo phương pháp tiếp cận hai pha (two-phase approach), đảm bảo sự tách biệt giữa quá trình xử lý dữ liệu chuyên sâu và quá trình phản hồi thời gian thực:

1. **Pha lập chỉ mục ngoại tuyến (Offline Indexing):** Tập trung vào việc chuẩn hóa nguồn dữ liệu đa dạng, trích xuất tri thức cốt lõi và metadata thời gian, từ đó xây dựng cơ sở tri thức đa chiều.

2. **Pha truy vấn trực tuyến (Online Retrieval & Generation):** Thực hiện hiểu câu hỏi người dùng, truy xuất thông tin dựa trên xếp hạng ngữ nghĩa – thời gian – ngữ cảnh sinh viên, và tổng hợp câu trả lời có trích dẫn xác thực.

Toàn bộ quy trình được điều phối bởi LangGraph dưới dạng một quy trình làm việc có trạng thái (stateful workflow). Kiến trúc này cho phép hệ thống duy trì ngữ cảnh xử lý, định tuyến linh hoạt dựa trên độ tin cậy của dữ liệu và tự động điều hướng khi gặp lỗi (fallback mechanisms).



Hình 5.1. Tổng quan kiến trúc hệ thống UITGraph và các lớp xử lý

5.2 Thu thập dữ liệu và chuẩn hóa tài liệu

Hệ thống thiết lập quy trình tiếp nhận tài liệu đa kênh để đảm bảo độ bao phủ thông tin:

- **Thu thập tự động:** Sử dụng dịch vụ Firecrawl [12] để quét và trích xuất dữ liệu từ các trang web công khai của UIT.
- **Tải lên thủ công:** Cho phép quản trị viên cập nhật tài liệu nội bộ qua giao diện Streamlit.
- **Kênh thông báo:** Tích hợp nguồn dữ liệu từ email và các thông báo hành chính.

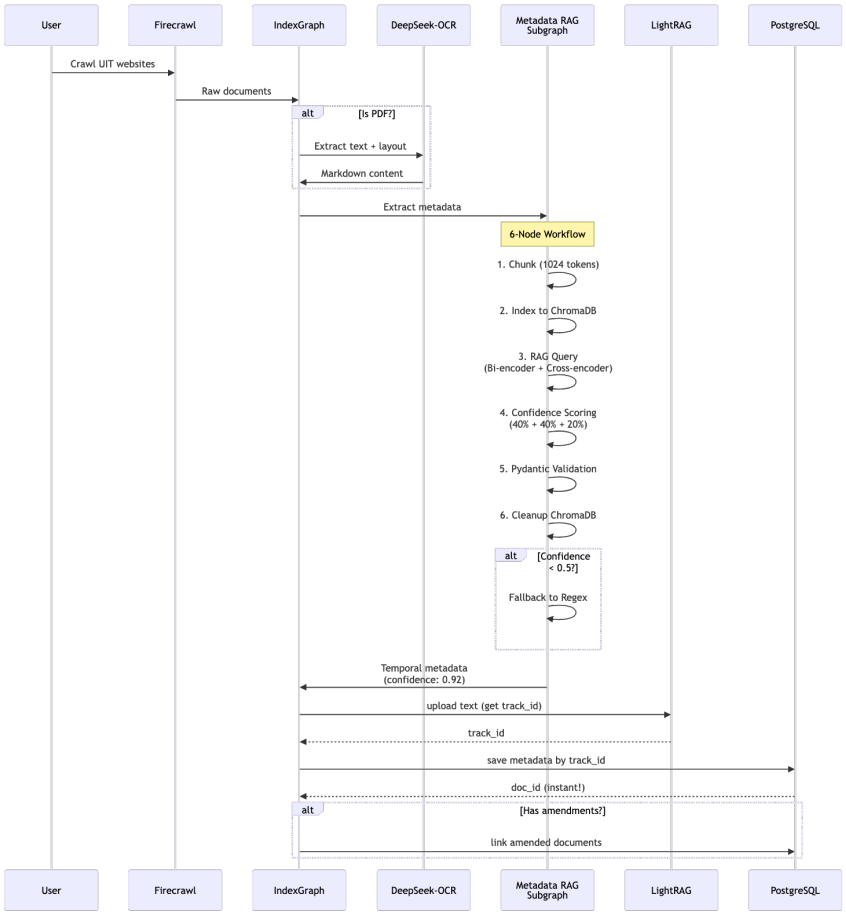
Dữ liệu thô sau khi thu thập sẽ được chuẩn hóa về định dạng Markdown trước khi đưa vào pha lập chỉ mục. Chiến lược xử lý được chia theo định dạng tệp:

- **Tài liệu HTML/văn bản:** Trích xuất nội dung và loại bỏ nhiễu trình bày để thu được văn bản phục vụ phân đoạn và trích xuất tri thức.
- **Đối với tài liệu PDF phức tạp** (chứa bảng biểu, tiêu đề phân cấp), hệ thống tích hợp mô hình DeepSeek-OCR nhằm bảo toàn cấu trúc layout. Đầu ra markdown từ OCR giúp giữ nguyên ngữ cảnh phân cấp (hierarchy context), tạo tiền đề quan trọng cho việc cắt đoạn (chunking) chính xác ở bước sau.

5.3 Pha lập chỉ mục ngoại tuyến (Offline Indexing)

Mục tiêu của pha này là chuyển đổi tài liệu thô thành ba cấu phần lưu trữ phục vụ cho các chiến lược truy vấn khác nhau:

1. **Vector Embeddings (Qdrant) [13]:** Phục vụ tìm kiếm tương đồng ngữ nghĩa.
2. **Knowledge Graph (NetworkX):** Phục vụ suy luận đa bước và truy vấn cấu trúc.
3. **Relational Metadata (PostgreSQL) [14]:** Đóng vai trò "Source of Truth" để kiểm soát tính hiệu lực và vòng đời tài liệu.



Hình 5.2. Trình tự xử lý trong luồng lập chỉ mục (Indexing Flow)

5.3.1 Điều phối quy trình và quản lý trạng thái bằng LangGraph

Toàn bộ quy trình lập chỉ mục được mô hình hóa thành một **Indexing Graph** – một workflow có trạng thái (stateful workflow) được quản lý bởi framework LangGraph [5]. Quy trình này thực hiện tuần tự các bước: khởi tạo danh sách tệp, kiểm tra định dạng, phân tích nội dung (tích hợp DeepSeek-OCR cho PDF), trích xuất metadata thời gian, và cuối cùng là tích hợp vào cơ sở tri thức LightRAG.

Khác với các luồng xử lý tuyến tính truyền thống, kiến trúc này duy trì một trạng thái xuyên suốt vòng đời xử lý. Cấu trúc này lưu trữ ba nhóm thông tin trọng yếu:

1. **Ngữ cảnh xử lý:** Bao gồm đường dẫn nguồn, loại tệp, và cấu hình phiên làm việc.

2. **Dữ liệu trung gian:** Chứa nội dung Markdown sau chuẩn hóa, kết quả OCR, và các metadata thời gian trích xuất được. Việc lưu trữ này cho phép các node xử lý sau có thể truy xuất lại kết quả của node trước đó mà không cần tính toán lại.

3. **Cơ chế kiểm soát lỗi:** Hệ thống ghi nhận trạng thái lỗi cục bộ (theo từng tệp) thay vì ngắt toàn bộ quy trình. Điều này cho phép cơ chế tự động thử lại hoặc bỏ qua tệp lỗi để đảm bảo tính ổn định của toàn hệ thống.

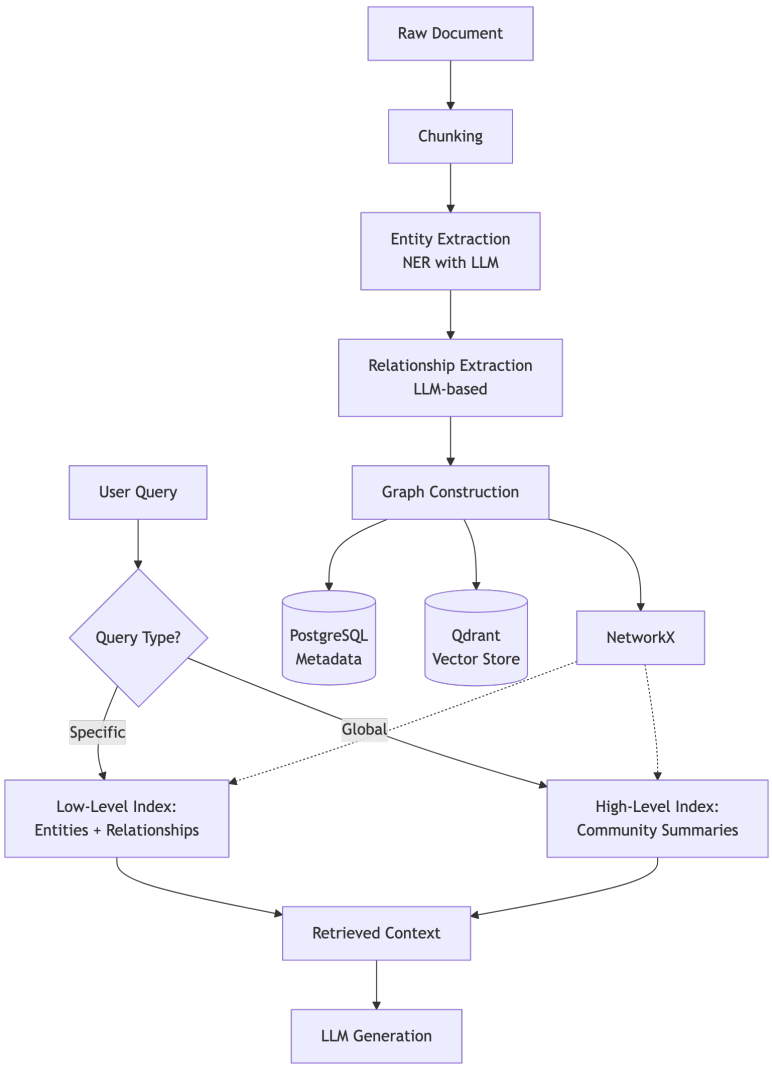
5.3.2 Xây dựng đồ thị tri thức và kho lưu trữ Hybrid

Tại bước xử lý trung tâm, tài liệu sau khi phân đoạn (chunking) sẽ trải qua quy trình trích xuất thông tin hai bước sử dụng LLM:

1. **Trích xuất thực thể (NER):** Nhận diện các đối tượng đặc thù trong miền dữ liệu đại học.
2. **Trích xuất quan hệ (Relation Extraction):** Xác định mối liên kết ngữ nghĩa giữa các thực thể.

Hệ thống áp dụng mô hình **Hybrid Storage** để tối ưu hóa việc lưu trữ:

- **NetworkX:** Lưu trữ đồ thị tri thức, biểu diễn các quan hệ cấu trúc như prerequisites (tiên quyết), belongs_to (thuộc về).
- **Qdrant:** Lưu trữ vector embeddings cho các đoạn văn bản và thực thể.
- **PostgreSQL:** Lưu trữ metadata thời gian chuẩn hóa, đảm bảo khả năng lọc chính xác (Exact Filtering) mà các cơ sở dữ liệu vector thường gặp hạn chế.



Hình 5.3. Quy trình xây dựng đồ thị tri thức và các kho lưu trữ phục vụ truy xuất

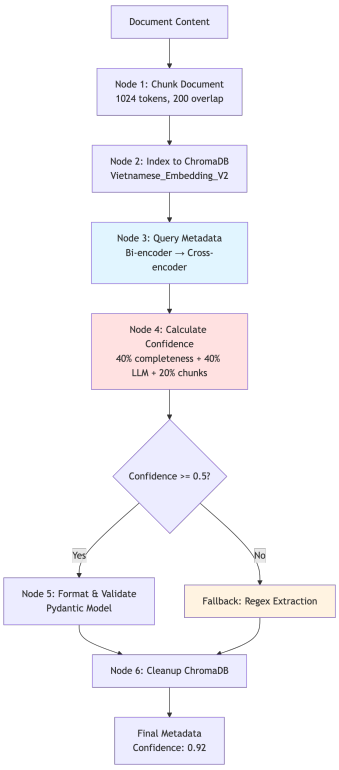
Hệ thống lược đồ được định nghĩa bao gồm:
Thực thể (Nodes):

- *Academic:* Course, Program, Department, Faculty.
- *Administrative:* Regulation, Procedure, Form.
- *Financial:* Scholarship, Tuition, Fee.
- *Personnel:* Lecturer, Staff, Advisor.
- *Temporal:* AcademicYear, Semester, Cohort.

Quan hệ (Edges): Tiêu biểu như issued_by (ban hành bởi), applies_to (áp dụng cho khóa), và đặc biệt là các quan hệ kiểm soát phiên bản văn bản như amends (sửa đổi), supersedes (thay thế).

5.3.3 Metadata RAG Subgraph - Trích xuất metadata thời gian thông minh

Thay vì sử dụng các phương pháp trích xuất truyền thống (Regex cứng nhắc hay LLM Prompting đơn lẻ) vốn dễ gặp lỗi ảo giác với văn bản pháp quy phức tạp, hệ thống UITGraph áp dụng kiến trúc **Metadata RAG Subgraph** – một quy trình RAG chuyên biệt (mini-RAG workflow) coi việc xác định siêu dữ liệu thời gian là một bài toán Hỏi-Đáp (Question Answering) ngược vào chính tài liệu đó.



Hình 5.4. Kiến trúc Metadata RAG

Quy trình xử lý 6 bước

Bước 1 - Phân mảnh ngữ cảnh lớn (Chunking):

Tài liệu được chia thành các đoạn lớn (1024 tokens với 200 overlap) để giữ trọn vẹn ngữ cảnh các điều khoản quy định. Khác với chunking thông thường (512 tokens), chiến lược này đảm bảo các thông tin metadata không bị cắt ngang.

Bước 2 - Lập chỉ mục tạm thời (In-Memory Indexing):

Khởi tạo ChromaDB vector store tạm thời chỉ tồn tại trong quá trình xử lý tài liệu đó. Sử dụng Vietnamese_Embedding_v2 (1024-dim) để tối ưu cho tiếng Việt.

Bước 3 - Truy vấn Metadata (Two-Stage Retrieval):

Giai đoạn 1 (Bi-Encoder): Sử dụng Vietnamese_Embedding_v2 để quét nhanh top-50 đoạn văn bản liên quan. Queries: "Ngày hiệu lực của văn bản này?", "Văn bản áp dụng cho khóa nào?", "Số hiệu văn bản?"

Giai đoạn 2 (Cross-Encoder): Sử dụng ViRanker [15] (namdp-ptit/ViRanker) để xếp hạng lại. Chọn ra top-5 đoạn chính xác nhất dựa trên semantic similarity score.

Bước 4 - Tính toán độ tin cậy trích xuất (Confidence Scoring):

Sau khi LLM thực hiện trích xuất thông tin từ các chunk ngữ cảnh được retrieve, hệ thống tính toán điểm tin cậy tổng hợp C_{total} dựa trên công thức trọng số:

$$C_{total} = 0.4 \times S_{completeness} + 0.4 \times S_{LLM} + 0.2 \times S_{quality} \tag{5.1}$$

Trong đó:

- $S_{completeness}$: Tỷ lệ trường bắt buộc được trích xuất thành công. Hệ thống yêu cầu 3 trường cốt lõi: `document_number`, `valid_from`, và `cohort_years`. Điểm này tính bằng số trường có giá trị hợp lệ chia cho tổng số trường bắt buộc (3).

- S_{LLM}: Tỷ lệ trường không rỗng (non-NULL ratio), phản ánh mức độ LLM có thể trích xuất được giá trị từ ngữ cảnh. Tính bằng $1.0 - \frac{\text{số trường NULL}}{\text{tổng số trường}}$, trong đó hệ thống kiểm tra 5 trường: `document_number`, `valid_from`, `valid_until`, `cohort_years`, `amends_documents`. Điểm cao cho thấy LLM không trả về giá trị "NULL" hoặc rỗng.

- $S_{quality}$: Tỷ lệ loại chunk được tìm thấy cho các truy vấn metadata (coverage ratio). Hệ thống thực hiện 4 truy vấn độc lập (cho `document_number`, `valid_from`, `cohort_years`, `amends_documents`) và tính tỷ lệ truy vấn có ít nhất một chunk được retrieve thành công từ vector store tạm thời.

Nếu quá trình trích xuất hoàn tất thành công, metadata được lưu trực tiếp vào PostgreSQL. Trong trường hợp gặp lỗi hoặc thất bại, hệ thống tự động kích hoạt cơ chế dự phòng (fallback) sử dụng Regex để trích xuất các trường cơ bản từ văn bản. Thực nghiệm cho thấy phương pháp RAG-based đạt điểm tin cậy trung bình **0.92** trên tập tài liệu kiểm tra, cải thiện đáng kể so với phương pháp Regex thuần túy (0.5-0.6).

Bước 5 - Kiểm tra và Định dạng (Pydantic Validation):

- Chuẩn hóa định dạng ngày tháng (YYYY-MM-DD)

- Chuẩn hóa số hiệu văn bản (e.g., "803/QĐ-ĐHCNTT")
- Validate cohort_years là danh sách số nguyên

Bước 6 - Dọn dẹp (Cleanup):

- Xóa ChromaDB collection tạm thời để giải phóng tài nguyên
- Đảm bảo không memory leak trong quá trình batch processing

So sánh hiệu quả

Bảng 5.1. So sánh hiệu quả thực nghiệm các phương pháp trích xuất metadata

Phương pháp	**Confidence**	**Tốc độ**	**Accuracy**	**Chi phí LLM**
Metadata RAG Subgraph	0.92	2-3s	92%	Trung bình
Regex-only (Fallback)	0.5-0.6	0.2s	50-60%	Không
LLM Prompting đơn lẻ	0.6-0.7	1.5s	60-70%	Cao

Kết luận:

- Metadata RAG Subgraph cải thiện 83% so với Regex-only (+0.42 absolute, từ 0.5 lên 0.92)
- Đảm bảo khả năng xử lý các quan hệ thời gian ẩn (e.g., "có hiệu lực sau 45 ngày kể từ ngày ký")
- Fallback mechanism đảm bảo hệ thống luôn trả về kết quả

5.3.4 Cơ chế đồng bộ dữ liệu và định danh (Data Synchronization)

Trong kiến trúc tích hợp với LightRAG [3], quá trình nạp và xử lý tài liệu diễn ra theo cơ chế bất đồng bộ (asynchronous processing). Khi nội dung được gửi đi, hệ thống chỉ nhận lại ngay lập tức một mã theo dõi (`track_id`) thay vì định danh tài liệu (`doc_id`) chính thức.

Để giải quyết vấn đề độ trễ đồng bộ (sync latency) thường gặp ở cơ chế kiểm tra lặp, hệ thống áp dụng quy trình tra cứu ngược. Quy trình thực hiện như sau:

1. Hệ thống nhận `track_id` ngay khi gửi yêu cầu lập chỉ mục.
2. Thay vì chờ đợi phản hồi từ API, hệ thống thực hiện truy vấn trực tiếp vào kho metadata PostgreSQL nội bộ của LightRAG.
3. Thực hiện ánh xạ `track_id` sang `doc_id` thực tế ngay khi bản ghi được khởi tạo.

4. Cập nhật metadata thời gian (temporal metadata) vào đúng bản ghi tài liệu thông qua `doc_id` vừa tìm được.

Giải pháp kỹ thuật này giúp loại bỏ hoàn toàn thời gian chờ chết (dead time) của cơ chế Polling, giảm thiểu tải cho API và ngăn chặn hiệu quả các lỗi timeout (quá thời gian chờ) khi thực hiện lập chỉ mục theo lô lớn (batch indexing).

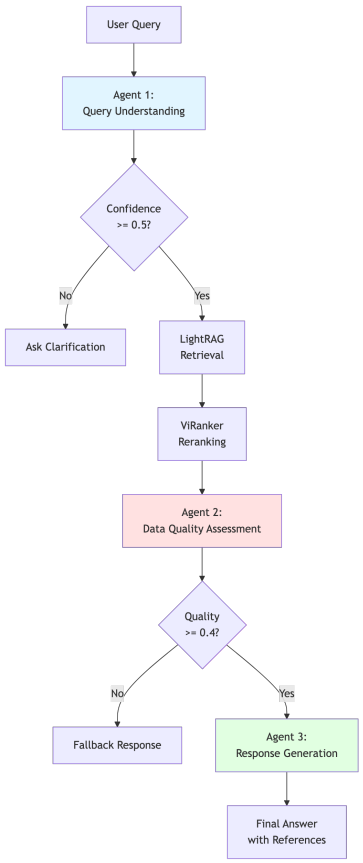
5.4 Pha truy vấn trực tuyến (Online Retrieval & Generation)

Pha truy vấn đóng vai trò là động cơ thực thi thời gian thực, chịu trách nhiệm tiếp nhận câu hỏi tự nhiên từ người dùng và sinh ra câu trả lời chính xác dựa trên bằng chứng.

Kiến trúc xử lý được tổ chức theo mô hình **Multi-Agent System** [6], bao gồm ba tác tử chuyên biệt:

1. **Agent 1 (Query Understanding):** Hiểu ý định và định tuyến.
2. **Agent 2 (Quality Assessment):** Đánh giá độ tin cậy của dữ liệu.
3. **Agent 3 (Response Generation):** Tổng hợp và sinh câu trả lời.

Quy trình này không chạy tuyến tính mà bao gồm các checkpoints để quyết định luồng đi (fallback hoặc tiếp tục), đảm bảo tính an toàn của câu trả lời.



Hình 5.5. Kiến trúc ba tác tử trong pha truy vấn của UITGraph

5.4.1 Hiểu truy vấn và cơ chế định tuyến ngữ nghĩa

Hệ thống quản lý quy trình truy vấn thông qua một **Query Graph** có trạng thái (stateful), duy trì ngữ cảnh xuyên suốt từ lúc nhận câu hỏi đến khi sinh câu trả lời.

Quản lý trạng thái

Cấu trúc trạng thái lưu trữ các nhóm thông tin:

- **Đầu vào:** Câu hỏi tự nhiên của người dùng.
- **Phân tích (Agent 1):** Ý định (Intention), thực thể (Entities), độ tin cậy (Confidence).
- **Truy xuất:** Danh sách ứng viên (chunks) và quan hệ đồ thị.
- **Đánh giá:** Điểm chất lượng, mức độ bao phủ thông tin.

Cơ chế định tuyến

Workflow áp dụng cơ chế rẽ nhánh có điều kiện tại hai điểm kiểm soát:

1. **Sau phân tích ý định:** Nếu độ tin cậy của mô hình hiểu ngôn ngữ $Confidence < 0.5$ hoặc phát hiện thiếu thông tin ngữ cảnh, hệ thống chuyển sang trạng thái "Hỏi để làm rõ" (Clarification) thay vì truy xuất mù quáng.
2. **Sau đánh giá chất lượng:** Nếu dữ liệu truy xuất không đạt ngưỡng chất lượng ($Quality < 0.4$), hệ thống kích hoạt cơ chế Fallback (Trả lời an toàn) để tránh hiện tượng ảo giác (hallucination).

Tác tử hiểu truy vấn (Agent 1)

Agent 1 đóng vai trò bộ lọc đầu vào, sử dụng LLM để trích xuất cấu trúc dữ liệu gồm: Ý định truy vấn, Danh sách thực thể chính, và câu hỏi làm rõ (nếu cần thiết).

5.4.2 Chiến lược Truy xuất kép và Xếp hạng lại

Sau khi xác định ý định, hệ thống thực hiện chiến lược **Truy xuất kép (Dual Retrieval)** trên nền tảng LightRAG [3]:

- Kết hợp tìm kiếm chi tiết (Low-level) dựa trên thực thể/đoạn văn.
- Kết hợp tìm kiếm tổng quát (High-level) dựa trên tóm tắt cộng đồng.

Tập ứng viên thu được (Top-K = 60) được đưa qua mô hình ViRanker [15] để tính điểm tương đồng ngữ nghĩa ($S_{semantic}$), tạo tiền đề cho thuật toán xếp hạng đa nhân tố.

5.4.3 Thuật toán xếp hạng đa nhân tố (Temporal & Cohort Scoring)

Đây là thành phần cốt lõi của UITGraph, đảm bảo kết quả trả về không chỉ đúng nội dung mà còn đúng thời điểm và đối tượng áp dụng.

Điểm số thời gian cơ sở (Temporal Base Score)

Hệ thống định nghĩa hàm tính điểm $S_{base}(d)$ dựa trên vòng đời và trạng thái sửa đổi của tài liệu d:

$$S_{base}(d) = \begin{cases} 0.0 & \text{nếu } d \text{ là tài liệu lưu trữ} \\ 0.1 & \text{nếu } d \text{ hết hạn } > 365 \text{ ngày} \\ 0.5 - \frac{t_{expired}}{365} \times 0.4 & \text{nếu } 0 < t_{expired} \leq 365 \\ 0.3 & \text{nếu } d \text{ bị sửa đổi} \\ f_{recency}(t_{age}) & \text{trường hợp khác} \end{cases} \tag{5.2}$$

Trong đó, hàm suy giảm theo độ mới $f_{recency}$ dựa trên tuổi đời tài liệu t_{age} (số ngày từ khi ban hành):

$$f_{recency}(t_{age}) = \begin{cases} 1.0 & \text{nếu } t_{age} \leq 30 \\ 0.9 - \frac{t_{age}-30}{365} \times 0.2 & \text{nếu } 30 < t_{age} \leq 365 \\ \max\left(0.5, 0.7 - \frac{t_{age}-365}{365} \times 0.2\right) & \text{nếu } t_{age} > 365 \end{cases} \tag{5.3}$$

Hệ số phạt rủi ro (Risk Penalty)

Hệ số phạt $P(d)$ được áp dụng để giảm thiểu rủi ro sử dụng thông tin lỗi thời:

- $P(d) = 0.5$: Tài liệu đã hết hạn.
- $P(d) = 0.8$: Tài liệu sắp hết hạn (trong vòng 30 ngày).
- Điểm thời gian thực tế: $S_{temporal}(d) = S_{base}(d) \times P(d)$.

Hệ số khuếch đại theo khóa sinh viên (Cohort Boost)

Nếu truy vấn chứa thông tin khóa (ví dụ: "K2024"), hệ số $C(d)$ được áp dụng:

- $C(d) = 1.5$: Khớp chính xác khóa áp dụng.
- $C(d) = 1.2$: Khớp gần (± 3 năm).
- $C(d) = 1.0$: Không có thông tin khóa.

Công thức xếp hạng tổng hợp

Điểm cuối cùng của tài liệu là tổ hợp tuyến tính giữa ngữ nghĩa và thời gian, được khuếch đại bởi hệ số khóa:

$$Score_{final}(d) = [(1 - w) \cdot S_{semantic}(d) + w \cdot S_{temporal}(d)] \times C(d) \tag{5.4}$$

(Với $w = 0.3$ là trọng số mặc định ưu tiên ngữ nghĩa)

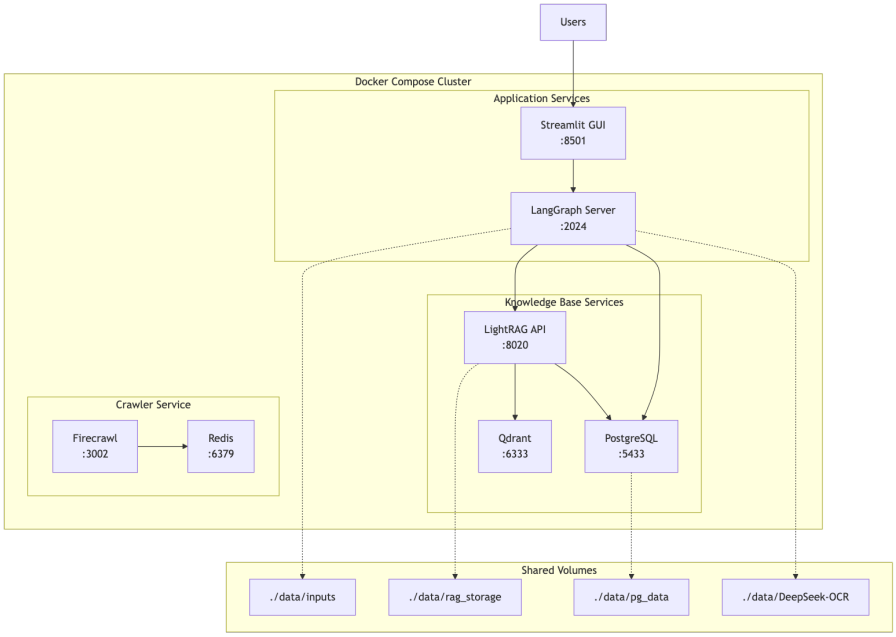
5.4.4 Kiểm định chất lượng và Sinh câu trả lời

Quy trình cuối cùng được thực hiện bởi hai tác tử chuyên biệt để đảm bảo độ tin cậy:

1. **Agent 2 (Đánh giá):** Kiểm tra Top-10 tài liệu sau xếp hạng về độ bao phủ thông tin. Nếu dữ liệu không đủ để trả lời câu hỏi, Agent kích hoạt cờ fallback.
2. **Agent 3 (Sinh đáp án):** Tổng hợp câu trả lời từ các bằng chứng đã chọn. Đặc biệt, Agent 3 có khả năng:
   - Tự động gắn liên kết tham chiếu đến văn bản gốc.
   - Chèn cảnh báo hiệu lực (Warning Note) nếu câu trả lời trích dẫn từ các văn bản sắp hết hạn hoặc đã có sửa đổi, giúp người dùng nhận thức được rủi ro áp dụng.

5.5 Triển khai hệ thống và cấu hình vận hành

Hệ thống UITGraph được đóng gói và triển khai theo kiến trúc vi dịch vụ (Microservices Architecture) trên nền tảng container hóa Docker. Việc điều phối các dịch vụ được thực hiện thông qua Docker Compose, đảm bảo tính nhất quán của môi trường chạy (runtime environment), khả năng mở rộng (scalability) và dễ dàng tái lập (reproducibility) trên các hạ tầng khác nhau.



Hình 5.6. Kiến trúc triển khai hệ thống theo Docker Compose

5.5.1 Phân nhóm dịch vụ (Service Orchestration)

Dựa trên sơ đồ kiến trúc, hệ thống được chia thành ba cụm dịch vụ logic, hoạt động độc lập nhưng liên kết chặt chẽ qua mạng nội bộ Docker (Bridge Network):

1. **Application Services (Tầng ứng dụng):**
   - *Streamlit GUI:* Cổng giao tiếp người dùng, cung cấp giao diện trực quan cho việc upload tài liệu và chat.
   - *LangGraph Server:* Đóng vai trò "nhạc trưởng", điều phối luồng xử lý trạng thái giữa các tác tử.
2. **Knowledge Base Services (Tầng tri thức & lưu trữ):**
   - *LightRAG API:* Cung cấp các endpoint truy xuất dữ liệu RAG.
   - *Qdrant:* Cơ sở dữ liệu Vector lưu trữ embeddings.
   - *PostgreSQL:* Lưu trữ metadata quan hệ và dữ liệu thời gian.
3. **Crawler Service (Tầng thu thập):**
   - *Firecrawl:* Dịch vụ thu thập dữ liệu web tự động.
   - *Redis:* Hàng đợi thông điệp và cache hỗ trợ cho Crawler.

5.5.2 Thông số cấu hình và Bảo toàn dữ liệu

Để đảm bảo an toàn dữ liệu và tách biệt môi trường, hệ thống sử dụng cơ chế Shared Volumes. Các cấu hình cổng và đường dẫn lưu trữ được quy hoạch chi tiết như sau:

Bảng 5.2. Đặc tả cấu hình mạng và lưu trữ của hệ thống

Nhóm	**Tên Service**	**Port**	**Vai trò chính**
Application	Streamlit GUI	8501:8501	Giao diện người dùng cuối
	LangGraph API	2024:80	API điều phối luồng xử lý
Knowledge	LightRAG API	8020:8020	Backend xử lý truy vấn RAG
	Qdrant	6333:6333	Vector Database Interface
	PostgreSQL	5433:5432	Relational Database (Metadata)
Crawler	Firecrawl	3002:3002	Web Crawler Service
	Redis	6379:6379	Cache & Message Queue

Cấu hình volumes mapping:

- `./data/inputs`: Chứa tài liệu thô đầu vào (PDF, Markdown).
- `./data/rag_storage`: Lưu trữ chỉ mục cục bộ của LightRAG (Entity/Relation store).
- `./data/pg_data`: Bảo toàn dữ liệu của PostgreSQL (tránh mất dữ liệu khi restart container).
- `./data/DeepSeek-OCR`: Lưu trữ mô hình OCR và cache kết quả xử lý hình ảnh.

5.5.3 Lựa chọn Model (Model Choice)

Hệ thống UITGraph sử dụng bộ model đa tầng được lựa chọn cẩn trọng để cân bằng giữa hiệu năng, chi phí và độ chính xác cho miền dữ liệu học vụ tiếng Việt.

Large Language Model (LLM)

- **Model:** Qwen3-4B-Instruct [16]
- **Lý do lựa chọn:**
  - *Lightweight và hiệu quả:* Với chỉ 4B parameters, Qwen3-4B đạt hiệu năng ấn tượng trên các tác vụ instruction-following và information extraction, trong khi vẫn đảm bảo tốc độ inference nhanh và yêu cầu tài nguyên thấp.
  - *Hỗ trợ tiếng Việt:* Model được pre-train trên corpus đa ngôn ngữ chất lượng cao bao gồm tiếng Việt, giúp hiểu ngữ cảnh văn bản hành chính và pháp quy tốt hơn các mô hình chỉ tập trung tiếng Anh.
  - *Chi phí triển khai thấp:* Kích thước nhỏ gọn cho phép deploy dễ dàng trên hardware phổ thông hoặc cloud với chi phí thấp, phù hợp với môi trường giáo dục có ngân sách hạn chế.
  - *Latency thấp:* Inference speed nhanh đảm bảo response time tốt cho các tác vụ real-time như query understanding và entity extraction, cải thiện trải nghiệm người dùng.

  - *Context window đầy đủ:* Hỗ trợ context length đủ lớn để xử lý các đoạn văn bản quy định và tài liệu học vụ thông thường mà không cần phân đoạn quá nhỏ.

- **Use case:** Entity extraction (LightRAG), Metadata extraction (Temporal RAG Subgraph), Query understanding (Agent 1), Response generation (Agent 3).

Embedding Model

- **Model:** AITeamVN/Vietnamese_Embedding_v2
- **Dimension:** 1024-dim
- **Lý do lựa chọn:**
  - *Chuyên biệt hóa tiếng Việt:* Model được fine-tune đặc biệt cho văn bản tiếng Việt, vượt trội so với các multilingual embedding models (như BGE-M3 [17]) trên semantic similarity tasks cho tiếng Việt.
  - *Dimension phù hợp:* 1024-dim cân bằng giữa khả năng biểu diễn ngữ nghĩa và hiệu quả lưu trữ/truy xuất trong vector database.
  - *Domain alignment:* Model thể hiện hiệu năng tốt với văn bản chuyên ngành (giáo dục, hành chính) so với các general-purpose embeddings.
- **Use case:** Indexing documents (chunk embedding), Query retrieval (semantic search trong Qdrant), Metadata RAG Subgraph (chunk retrieval).

Reranker Model

- **Model:** namdp-ptit/ViRanker [15]
- **Architecture:** Cross-encoder dựa trên PhoBERT backbone
- **Lý do lựa chọn:**
  - *Cross-encoder superiority:* Khác với bi-encoder (embedding model), cross-encoder có khả năng capture interaction giữa query và document tốt hơn, dẫn đến độ chính xác cao hơn trong reranking.
  - *Vietnamese-specific:* Model được train đặc biệt cho văn bản tiếng Việt, đạt 85%+ accuracy trên văn bản giáo dục tiếng Việt.
  - *Two-stage efficiency:* Sử dụng trong giai đoạn 2 của retrieval (sau khi vector search), cho phép cân bằng giữa tốc độ (bi-encoder) và độ chính xác (cross-encoder).
- **Use case:** Reranking top-k chunks trong Metadata RAG Subgraph, Quality assessment trong multi-factor ranking.

OCR Model

- **Model:** DeepSeek-OCR
- **Lý do lựa chọn:**
  - *Layout preservation:* Model có khả năng bảo toàn cấu trúc layout của tài liệu PDF phức tạp (bảng biểu, tiêu đề phân cấp), quan trọng để giữ nguyên hierarchy context cho chunking.
  - *Markdown output:* Đầu ra dạng markdown giúp dễ dàng parse và maintain semantic structure của tài liệu.
  - *Multilingual support:* Hỗ trợ tốt cả tiếng Việt và tiếng Anh, phù hợp với tài liệu học vụ thường chứa cả hai ngôn ngữ.
- **Use case:** Pre-processing PDF documents trước khi indexing.

5.6 Thiết kế thực nghiệm và Phương pháp đánh giá

Để kiểm chứng tính hiệu quả của kiến trúc UITGraph, quá trình đánh giá được thiết kế tập trung vào bốn thành phần cốt lõi:

1. **Khả năng nhận thức thời gian:** Độ chính xác trong việc trích xuất metadata và quan hệ sửa đổi.
2. **Hiệu năng truy xuất:** Chất lượng của thuật toán xếp hạng đa nhân tố (Ngữ nghĩa – Thời gian – Khóa sinh viên).
3. **Độ an toàn của hệ thống:** Độ tin cậy của cơ chế định tuyến và khả năng kích hoạt Fallback khi thiếu thông tin.
4. **Hiệu suất vận hành:** Tốc độ lập chỉ mục và độ trễ của cơ chế đồng bộ dữ liệu.

5.6.1 Bộ chỉ số đánh giá

Temporal Extraction

Đánh giá mức độ khớp giữa dữ liệu trích xuất và nhãn thủ công trên các trường: `valid_from`, `valid_until`, `cohort_scope`.

- **Precision (Độ chính xác):** Tỷ lệ thông tin thời gian trích xuất đúng trên tổng số thông tin trích xuất.
- **Accuracy (Độ đúng):** Tỷ lệ tài liệu được xác định đúng hoàn toàn trạng thái sửa đổi (`amends`/`amended_by`).

Query Pipeline

Sử dụng các chỉ số phổ biến trong Information Retrieval:

- **Hit Rate @ K:** Tỷ lệ câu hỏi có ít nhất một văn bản liên quan xuất hiện trong Top-K kết quả.
- **Temporal Accuracy @ K:** Chỉ số tùy biến, đo lường tỷ lệ các văn bản trong Top-K có tính hiệu lực (còn hạn/đúng khóa) so với yêu cầu của câu hỏi.

Routing & Safety

Sử dụng ma trận nhầm lẫn để đánh giá quyết định của Agent:

- **True Positive:** Hệ thống trả lời khi có đủ thông tin.
- **True Negative:** Hệ thống từ chối (Fallback/Clarify) khi không đủ thông tin hoặc câu hỏi không rõ ràng.
- **False Positive (Nguy hiểm):** Hệ thống cố trả lời (Hallucination) khi dữ liệu thiếu hoặc sai.

System Performance

- **Indexing Latency:** Thời gian trung bình để xử lý một lô tài liệu (Batch processing time).
- **Sync Lag:** Độ trễ giữa lúc nhận `track_id` và lúc hoàn tất ghi metadata (`doc_id` mapping).

5.6.2 Quy trình thực nghiệm

Quá trình đánh giá được thực hiện theo kịch bản bốn giai đoạn:
Giai đoạn 1: Chuẩn bị dữ liệu

- Xây dựng bộ dữ liệu kiểm thử (Golden Dataset) bao gồm các cặp câu hỏi-câu trả lời (QA pairs) và các tài liệu quy chế có tính chất phức tạp về thời gian (sửa đổi, thay thế).
- Gán nhãn thủ công (Manual labeling) cho các ý định truy vấn và trạng thái fallback để làm cơ sở tham chiếu (Ground Truth).

Giai đoạn 2: Đánh giá thành phần

- Thực thi pipeline lập chỉ mục trên tập dữ liệu mẫu.
- Thống kê tỷ lệ lỗi của module DeepSeek-OCR trên các tài liệu PDF bảng biểu.

- Đo lường độ chính xác của Temporal Extraction Agent khi sử dụng chiến lược Regex đơn thuần so với chiến lược kết hợp LLM.

Giai đoạn 3: Đánh giá End-to-End và cơ chế an toàn

- Chạy tập truy vấn kiểm thử qua toàn bộ hệ thống.
- Phân tích ma trận nhầm lẫn để đánh giá ngưỡng tin cậy (Confidence Threshold). Mục tiêu là tối thiểu hóa tỷ lệ False Positive (trả lời sai/bịa đặt), chấp nhận tăng tỷ lệ Fallback để đảm bảo tính chính xác học thuật.

Giai đoạn 4: Nghiên cứu so sánh

Để chứng minh hiệu quả của thuật toán xếp hạng đề xuất, thực hiện so sánh kết quả Top-1 giữa hai cấu hình:

1. **Baseline:** Chỉ sử dụng tìm kiếm ngữ nghĩa (Semantic Search thuần túy).
2. **UITGraph:** Sử dụng Semantic Search kết hợp Temporal Scoring và Cohort Boost.

Mục tiêu: Phân tích các trường hợp sai (Error Analysis) của Baseline mà UITGraph khắc phục được, đặc biệt là các lỗi liên quan đến văn bản cũ/hết hạn.

CHƯƠNG 6: KẾT QUẢ ĐẠT ĐƯỢC VÀ HƯỚNG PHÁT TRIỂN

6.1 Kết quả chức năng (Functional Outcomes)

Hệ thống hướng tới việc hoàn thiện ba nhóm năng lực cốt lõi:

- **Xây dựng Hybrid Knowledge Base:** Hợp nhất thành công ba luồng dữ liệu: Đồ thị tri thức (cấu trúc), Vector embeddings (ngữ nghĩa) và Metadata (thời gian), tạo nền tảng cho truy vấn đa chiều.
- **Cơ chế truy xuất nhận thức ngữ cảnh:** Hiện thực hóa thuật toán xếp hạng Cohort-aware, đảm bảo thông tin được trích xuất chính xác theo khóa sinh viên, loại bỏ nhiễu từ các văn bản không còn hiệu lực.
- **Sinh câu trả lời an toàn và minh bạch:** Hệ thống có khả năng tự động gắn trích dẫn nguồn và phát cảnh báo rủi ro đối với các văn bản sắp hết hạn hoặc đã có sửa đổi.

6.2 Các chỉ số kỳ vọng

Dựa trên thiết kế thực nghiệm, hệ thống đặt ra các mốc hiệu năng mục tiêu để kiểm chứng tính khả thi và độ chính xác so với các phương pháp truyền thống.

Bảng 6.1. Bảng mục tiêu kỳ vọng

Thành phần	**Tiêu chí đánh giá**	**Target**
Temporal Extraction	Độ chính xác trích xuất metadata	$\geq 92\%$
Query Routing	Độ chính xác phân loại Intent/Fallback	$\geq 90\%$
Response Quality	Độ chính xác về tính thời gian	$\geq 90\%$
Reranking	Khả năng cải thiện Top-1 so với Baseline	$\geq 85\%$
Indexing Performance	Độ trễ đồng bộ Metadata (Sync Lag)	Thấp (Real-time)

TÀI LIỆU THAM KHẢO

[1] P. Lewis, E. Perez, A. Piktus, *et al.*, "Retrieval-augmented generation for knowledge-intensive nlp tasks," *arXiv preprint arXiv:2005.11401*, 2020.

[2] D. Edge, H. Trinh, N. Cheng, *et al.*, "From local to global: A graph rag approach to query-focused summarization," *arXiv preprint arXiv:2404.16130*, 2024.

[3] Z. Guo, L. Xia, Y. Yu, T. Ao, and C. Huang, "Lightrag: Simple and fast retrieval-augmented generation," *arXiv preprint arXiv:2410.05779*, 2024. Accepted at EMNLP 2025.

[4] B. Jiménez Gutiérrez, Y. Shu, Y. Gu, M. Yasunaga, and Y. Su, "Hipporag: Neurobiologically inspired long-term memory for large language models," *arXiv preprint arXiv:2405.14831*, 2024. Accepted at NeurIPS 2024.

[5] LangChain, "Langgraph overview," 2026.

[6] LangChain, "Multi-agent collaboration," 2026.

[7] LangChain, "Langgraph graph api," 2026.

[8] D. Li, Y. Niu, Y. Ai, X. Zou, B. Qi, and J. Liu, "T-grag: A dynamic graphrag framework for resolving temporal conflicts and redundancy in knowledge retrieval," *arXiv preprint arXiv:2508.01680*, 2025. To appear in MM '25, October 27–31, 2025, Dublin, Ireland.

[9] D. Huwiler, K. Stockinger, and J. Fürst, "Versionrag: Version-aware retrieval-augmented generation for evolving documents," *arXiv preprint arXiv:2510.08109*, 2025.

[10] L. Nguyen and T. Quan, "Urag: Implementing a unified hybrid rag for precise answers in university admission chatbots," *arXiv preprint arXiv:2501.16276*, 2025.

[11] T. Ma, T.-T. La, L.-T. Le Huu, M.-N. Nguyen, and K.-V. Pham Luu, "Rebot: From rag to catrag with semantic enrichment and graph routing," *arXiv preprint arXiv:2510.01800*, 2025.

[12] Firecrawl, "Firecrawl documentation," 2026.

[13] Qdrant, "Qdrant documentation," 2026.

[14] The PostgreSQL Global Development Group, "Postgresql documentation," 2026.

[15] P.-N. Dang, K.-L. Nguyen, and T.-H. Pham, "Viranker: A bge-m3 & block-wise parallel transformer cross-encoder for vietnamese reranking," *arXiv preprint arXiv:2509.09131*, 2025.

[16] A. Yang *et al.*, "Qwen3 technical report," *arXiv preprint arXiv:2505.09388*, 2025. Submitted 14 May 2025.

[17] X. Xiao, X. Ye, B. Wang, *et al.*, "Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation," *arXiv preprint arXiv:2402.03216*, 2024.