ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH

BÁO CÁO
ĐỒ ÁN CHUYÊN NGÀNH

Hệ thống Truy xuất Thông tin Đa phương thức từ Âm thanh Kết hợp Large Language Models

---

Ngành: KHOA HỌC MÁY TÍNH

HỘI ĐỒNG: KHOA HỌC MÁY TÍNH

GVHD: VÕ THANH HÙNG

—o0o—

SVTH: VÕ LÝ ĐẮC DUY 2252125

TP. HỒ CHÍ MINH, THÁNG 12/2025

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH

BÁO CÁO
ĐỒ ÁN CHUYÊN NGÀNH

Hệ thống Truy xuất Thông tin Đa phương thức từ Âm thanh Kết hợp Large Language Models

---

Ngành: KHOA HỌC MÁY TÍNH

HỘI ĐỒNG: KHOA HỌC MÁY TÍNH

GVHD: VÕ THANH HÙNG

—o0o—

SVTH: VÕ LÝ ĐẮC DUY 2252125

TP. HỒ CHÍ MINH, THÁNG 12/2025

Lời cam đoan

Em xin khẳng định đây là sản phẩm nghiên cứu của riêng em, được hoàn thành với sự hướng dẫn khoa học của thầy Võ Thanh Hùng. Mọi dữ liệu, hình ảnh và kết quả trình bày trong báo cáo đều phản ánh chính xác quá trình thực hiện, không qua bất kỳ sự chỉnh sửa hay ngụy tạo nào. Những ý tưởng và thông tin được kế thừa từ các nghiên cứu trước đã được ghi nhận rõ ràng trong danh mục tài liệu tham khảo. Em hoàn toàn chịu trách nhiệm về tính chính xác của những điều đã nêu.

TP. Hồ Chí Minh, tháng 12 năm 2025

Sinh viên thực hiện

Võ Lý Đắc Duy

Lời cảm ơn

Trước tiên, em muốn bày tỏ sự tri ân đến thầy Võ Thanh Hùng vì những chỉ dẫn tận tâm và sự kiên nhẫn trong việc đồng hành cùng em suốt thời gian qua. Nhờ có thầy mà em có thể biến những ý tưởng ban đầu thành một sản phẩm hoàn chỉnh. Em cũng cảm ơn công sức của các giảng viên tại Khoa Khoa học và Kỹ thuật Máy tính, những người đã xây dựng cho em một nền móng kiến thức vững chắc về lập trình, thuật toán và trí tuệ nhân tạo. Không có những bài giảng đó, em khó có thể tiếp cận được các công nghệ phức tạp mà đề tài yêu cầu. Sau cùng, em biết ơn sự ủng hộ không ngừng từ gia đình, những người luôn là chỗ dựa tinh thần vững chãi nhất.

Tóm tắt đề tài

Việc truy xuất thông tin từ dữ liệu âm thanh trong môi trường giáo dục đặt ra nhiều thách thức kỹ thuật đặc thù. Các bài giảng được ghi âm thường chứa thuật ngữ chuyên ngành, giọng nói có đặc điểm vùng miền, và cấu trúc nội dung phức tạp với nhiều chủ đề đan xen. Trong khi đó, các hệ thống nhận dạng giọng nói phổ biến chưa được tối ưu hóa cho tiếng Việt, và mô hình ngôn ngữ lớn (LLM) thường sinh ra thông tin không có căn cứ khi được hỏi về nội dung cụ thể.

Nghiên cứu này tập trung giải quyết hai bài toán chính: (1) xây dựng pipeline xử lý âm thanh tiếng Việt với khả năng lưu giữ thông tin vị trí thời gian phục vụ trích dẫn nguồn, và (2) phát triển cơ chế kiểm soát để giảm thiểu hiện tượng ảo giác (hallucination) khi LLM sinh câu trả lời dựa trên nội dung được truy xuất.

Hệ thống đề xuất kiến trúc Hybrid RAG xây dựng kho tri thức thống nhất từ âm thanh và các tài liệu văn bản đi kèm, kết hợp ba thành phần chính. Thành phần thứ nhất là pipeline xử lý âm thanh tích hợp Voice Activity Detection (VAD) với ASR, trong đó VAD lọc các đoạn im lặng trước khi đưa vào mô hình nhận dạng, đồng thời ánh xạ word-level timestamps để định vị chính xác vị trí từng câu trong file gốc. Thành phần thứ hai là hệ thống tìm kiếm hybrid kết hợp semantic search với BM25 thông qua phương pháp weighted combination, sau đó sử dụng cross-encoder để rerank kết quả trên cả hai loại nguồn dữ liệu. Thành phần thứ ba là cơ chế chống ảo giác ba tầng giúp hệ thống duy trì độ tin cậy cao ngay cả khi dữ liệu ASR có nhiễu: Answer Verification đánh giá mức độ grounding của câu trả lời so với nguồn tài liệu, Information Resolution dựa trên trọng số thời gian để phát hiện và giải quyết mâu thuẫn giữa các nguồn bằng cách ưu tiên thông tin có metadata thời gian mới hơn, và Safe Abstention từ chối trả lời khi độ tin cậy thấp hoặc không tìm thấy thông tin liên quan.

Kết quả thực nghiệm trên tập dữ liệu đa phương thức gồm 50 bài giảng âm thanh và 80 tài liệu văn bản tương ứng, với 100 câu hỏi đánh giá cho thấy: module ASR đạt Word Error Rate 15% với tốc độ xử lý nhanh gấp 4 lần so với Whisper gốc (RTF = 0.06); hệ thống retrieval đạt MRR 0.75 và Recall@10 là 83%, chứng minh khả năng truy xuất chính xác trên cả hai loại nguồn dữ liệu; cơ chế chống ảo giác nâng Grounding Accuracy từ 65% lên 79% và giảm tỷ lệ hallucination từ 28% xuống 14%. Thời gian phản hồi trung bình cho một truy vấn là 1.8 giây trên GPU tầm trung.

Từ khóa: Truy xuất đa phương thức, Nhận dạng giọng nói tiếng Việt, Retrieval-Augmented Generation, Chống ảo giác, Hybrid Search, Siêu dữ liệu thời gian.

Mục lục

Danh sách bảng

Danh sách hình vẽ

Chương 1

Giới thiệu

Chương này trình bày tổng quan về đề tài, bao gồm bối cảnh và động lực nghiên cứu, mục tiêu cần đạt được, phạm vi thực hiện, phân tích yêu cầu hệ thống, và cấu trúc của báo cáo.

1.1 Đặt vấn đề

Các cơ sở giáo dục đại học đang đối mặt với sự bùng nổ của dữ liệu phi cấu trúc. Mỗi học kỳ, hàng nghìn giờ bài giảng được ghi âm, hàng trăm video hội thảo được lưu trữ, cùng với vô số quyết định hành chính, thông báo và tài liệu học thuật được ban hành. Tuy nhiên, phần các lớn tri thức này là những tài nguyên số được thu thập và lưu trữ nhưng không được lập chỉ mục (Dark Data [1]), do đó không thể truy cập thông qua các phương thức truy vấn thông thường.

Vấn đề trở nên nghiêm trọng hơn khi xét đến hiện tượng đứt gãy tri thức trong môi trường giảng dạy. Khi giảng viên thực hiện bài giảng, họ thường bổ sung nhiều nội dung quan trọng không có trong slide hay giáo trình: ví dụ minh họa thực tế, giải thích chuyên sâu, ... Phần lớn các thông tin này chỉ tồn tại trong luồng âm thanh, không được ghi lại trong tài liệu bổ trợ [2]. Nếu không được xử lý, lượng tri thức này sẽ trở thành dữ liệu tối, không thể được khai thác lại bởi sinh viên. Thách thức kỹ thuật nằm ở việc chuyển đổi âm thanh sang văn bản. Các hệ thống ASR phổ biến thường thiếu tối ưu cho tiếng Việt, đặc biệt là hiện tượng **code-switching** (xen kẽ thuật ngữ STEM tiếng Anh). Bên cạnh đó, các phương pháp tìm kiếm truyền thống dựa trên từ khóa không thể xử lý được các truy vấn mang tính ngữ nghĩa phức tạp.

Sự ra đời của các mô hình ngôn ngữ lớn như GPT-4, Gemini, hay LLaMA đã mở ra hướng tiếp cận mới cho bài toán truy xuất và tổng hợp thông tin [3]. Tuy nhiên, việc áp dụng LLMs trong môi trường giáo dục đặt ra ba thách thức quan trọng. Thứ nhất là hiện tượng ảo giác (hallucination) khi LLMs có xu hướng sinh ra thông tin không chính xác hoặc bịa đặt khi được hỏi về nội dung cụ thể mà chúng không có trong dữ liệu huấn luyện [4]. Trong bối cảnh giáo dục, một câu trả lời sai về quy chế học vụ hay thông tin học phí có thể gây hậu quả nghiêm trọng cho sinh viên. Thứ hai là vấn đề quyền riêng tư và bảo mật dữ liệu. Việc gửi dữ liệu bài giảng nội bộ, thông tin sinh viên, hay tài liệu chưa công bố lên các dịch vụ cloud của bên thứ ba như OpenAI hay Google đặt ra vấn đề pháp lý và bảo mật nghiêm trọng. Nhiều cơ sở giáo dục có chính sách cấm chia sẻ dữ liệu nội bộ ra bên ngoài. Thứ ba là chi phí và tự chủ công nghệ. Sự phụ thuộc vào các dịch vụ LLM thương mại không chỉ tạo ra gánh nặng tài chính mà còn đặt cơ sở giáo dục vào thế bị động khi nhà cung cấp thay đổi chính sách, giá cả, hoặc ngừng hoạt động.

Từ những phân tích trên, việc xây dựng một hệ thống truy xuất thông tin đa phương thức có khả năng xử lý và tìm kiếm nội dung từ âm thanh, video và văn bản, kết hợp LLMs với cơ chế kiểm chứng để giảm thiểu ảo giác, đồng thời hoạt động hoàn toàn cục bộ để đảm bảo bảo mật và tự chủ công nghệ là một nhu cầu thiết thực và cấp bách của các cơ sở giáo dục.

1.2 Mục tiêu đề tài

1.2.1 Mục tiêu tổng quát

Xuất phát từ những thách thức đã phân tích ở phần đặt vấn đề, nghiên cứu này hướng đến việc xây dựng một hệ thống truy xuất thông tin thông minh dựa trên mô hình ngôn ngữ lớn (LLM), có khả năng xử lý dữ liệu đa phương thức bao gồm âm thanh, video và văn bản. Điểm khác biệt cốt lõi so với các giải pháp hiện có nằm ở ba yếu tố: *(i)* tính chính xác thông qua cơ chế kiểm chứng câu trả lời, *(ii)* tính bảo mật nhờ khả năng vận hành hoàn toàn cục bộ, và *(iii)* tính tự chủ công nghệ khi không phụ thuộc vào dịch vụ bên thứ ba.

1.2.2 Mục tiêu cụ thể

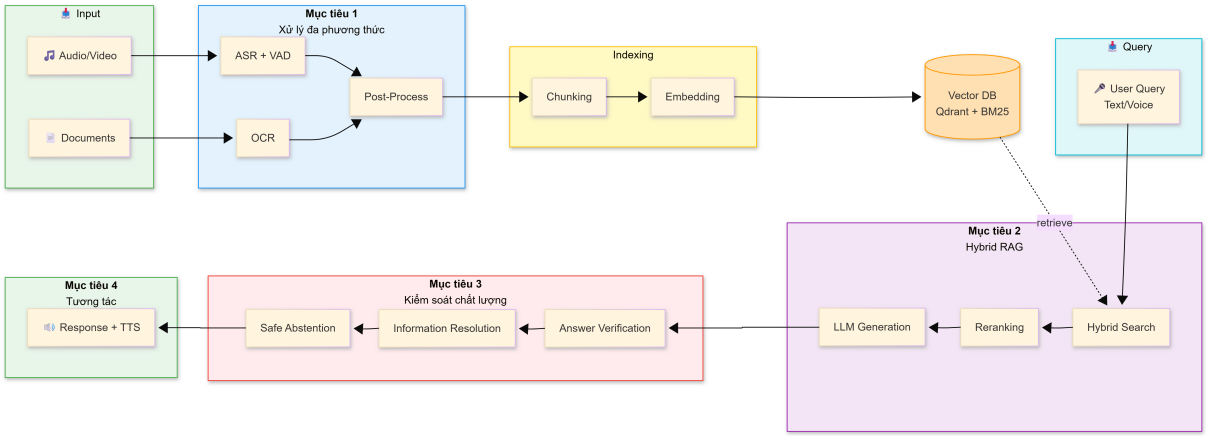
Mục tiêu xử lý âm thanh và ngôn ngữ. Đây là nền tảng của toàn bộ hệ thống, nhằm giải quyết thách thức về "dữ liệu tối" đã nêu trong phần đặt vấn đề. Cụ thể, nghiên cứu phát triển pipeline ASR tiếng Việt được tối ưu cho bối cảnh học thuật, đặc biệt là khả năng xử lý hiện tượng code-switching phổ biến trong giảng dạy các ngành STEM. Pipeline này tích hợp Voice Activity Detection (VAD) để lọc các đoạn im lặng, đồng thời duy trì word-level timestamps nhằm thiết lập mối liên kết chính xác giữa văn bản phiên âm và vị trí thời gian trong file gốc.

Mục tiêu truy xuất tri thức đa phương thức. Sau khi âm thanh đã được chuyển đổi thành văn bản, thách thức tiếp theo là làm sao tìm kiếm hiệu quả trên kho tri thức hỗn hợp. Nghiên cứu hiện thực hóa kiến trúc Hybrid RAG, kết hợp Dense Retrieval (nhúng vector) để nắm bắt ngữ nghĩa với Sparse Retrieval (BM25) để đảm bảo không bỏ sót các thuật ngữ chuyên ngành. Sự kết hợp này giải quyết điểm yếu của từng phương pháp khi sử dụng riêng lẻ.

Mục tiêu kiểm soát chất lượng phản hồi. Đây là mục tiêu phân biệt nghiên cứu này với các hệ thống RAG thông thường. Nhằm giải quyết vấn đề ảo giác của LLM – một rủi ro nghiêm trọng trong bối cảnh giáo dục, nghiên cứu thiết lập cơ chế chống ảo giác ba tầng: *Answer Verification* để kiểm tra độ trung thực của câu trả lời so với nguồn, *Information Resolution* để phát hiện và hòa giải mâu thuẫn giữa các nguồn thông tin, và *Safe Abstention* để từ chối trả lời khi không có đủ căn cứ. Mục tiêu cuối cùng là đảm bảo mọi câu trả lời đều được grounding hoàn toàn vào cơ sở tri thức nội bộ.

Mục tiêu tương tác và tối ưu hóa. Mục tiêu này tập trung vào khía cạnh trải nghiệm người dùng và tính khả thi triển khai. Về tương tác, nghiên cứu xây dựng giao diện hỏi đáp đa phương thức Voice-to-Voice, cho phép sinh viên đặt câu hỏi bằng giọng nói và nhận câu trả lời được đọc tự động. Về tối ưu hóa, hệ thống được điều chỉnh để vận hành ổn định trên hạ tầng phần cứng cục bộ với tổng thời gian phản hồi dưới 10 giây cho các truy vấn thông thường, đồng thời hỗ

trợ *streaming* để tối ưu trải nghiệm chờ đợi của người dùng. Hệ thống không yêu cầu kết nối Internet trong quá trình sử dụng.



Hình 1.1: Sơ đồ luồng dữ liệu tổng quan và mối liên hệ với các mục tiêu cụ thể

1.3 Phạm vi đề tài

Phần này xác định rõ ranh giới của đề tài trên ba khía cạnh: loại dữ liệu được xử lý, công nghệ được lựa chọn, và những giới hạn mà nghiên cứu không đề cập đến. Việc phân định rõ ràng này giúp định hướng kỳ vọng của người đọc và tạo cơ sở cho việc đánh giá kết quả nghiên cứu.

1.3.1 Phạm vi về chức năng và dữ liệu

Xuất phát từ bối cảnh ứng dụng trong môi trường giáo dục đại học, nghiên cứu tập trung vào hai nguồn dữ liệu chính với các đặc điểm riêng biệt.

Đối với **dữ liệu âm thanh và video**, phạm vi bao gồm các bài giảng được ghi âm với độ dài từ 15 phút đến 120 phút mỗi file, tương ứng với một tiết học hoặc buổi giảng hoàn chỉnh. Chất lượng âm thanh yêu cầu ở mức trung bình trở lên, tương đương với ghi âm bằng smartphone hoặc thiết bị chuyên dụng trong phòng học có kiểm soát tiếng ồn. Đáng lưu ý, mặc dù hệ thống tiếp nhận cả file video, quá trình xử lý chỉ trích xuất phần âm thanh để phân tích – do đó, nghiên cứu này được định vị chính xác hơn là **"Truy xuất đa phương thức dựa trên nội dung tiếng nói"**, khai thác tri thức từ luồng thông tin nói trong các định dạng vật lý khác nhau. Nội dung hình ảnh trong video như slide trình chiếu, bảng viết, hay cử chỉ giảng viên nằm ngoài phạm vi xử lý của nghiên cứu này.

Về **dữ liệu văn bản**, hệ thống hỗ trợ các định dạng tài liệu học thuật tiêu chuẩn bao gồm PDF, Word, Excel và PowerPoint. Riêng với tài liệu PDF dạng scan hoặc hình ảnh, việc trích xuất văn bản được thực hiện thông qua OCR. Tuy nhiên, do hạn chế của các engine OCR hiện tại với ngôn ngữ Việt, hệ thống **chưa xử lý được các công thức toán học phức tạp**, sơ đồ hình học đặc thù, hoặc ký hiệu chuyên ngành nằm ngoài bộ ký tự Unicode chuẩn.

Về **khả năng truy xuất**, trọng tâm nghiên cứu nằm ở việc hiện thực hóa kiến trúc **Hybrid RAG** kết hợp Dense Retrieval và Sparse Retrieval, cùng với cơ chế **Anti-hallucination** ba tầng. Các

tính năng Voice Input và Text-to-Speech được xem là phương thức tương tác bổ trợ nhằm tối ưu trải nghiệm người dùng, không phải trọng tâm nghiên cứu chính. Hệ thống phục vụ hai nhóm người dùng: sinh viên tra cứu thông tin qua Student Portal, và quản trị viên thông qua Admin Portal – một công cụ quản trị tri thức và kiểm soát dữ liệu cho phép kiểm soát chất lượng dữ liệu đầu vào, cấu hình các tham số chunking/embedding, và theo dõi trạng thái của kho tri thức.

1.3.2 Phạm vi về công nghệ và mô hình

Để đảm bảo tính tự chủ và khả năng vận hành nội bộ như đã nêu trong phần đặt vấn đề, nghiên cứu ưu tiên tuyển chọn các công nghệ mã nguồn mở có cộng đồng hỗ trợ mạnh mẽ. Bảng 1.1 tóm tắt các công nghệ chính được sử dụng trong từng module; chi tiết về cấu hình và thông số kỹ thuật sẽ được trình bày trong Chương 4.

Bảng 1.1: Tổng quan công nghệ theo module

Module	**Công nghệ chủ đạo**	**Công nghệ đối chứng**
Nhận dạng giọng nói	Faster-Whisper, Silero VAD	–
Mô hình ngôn ngữ	Ollama (Qwen2.5, LLaMA)	Google Gemini, OpenAI GPT
Embedding	Sentence-BERT, E5	Google, OpenAI Embedding API
Cơ sở dữ liệu vector	Qdrant (Hybrid Search)	–

Kiến trúc được thiết kế theo mô hình **Multi-provider** với sự phân biệt rõ ràng: nhóm *công nghệ chủ đạo* là các mô hình mã nguồn mở có thể triển khai hoàn toàn cục bộ, đảm bảo dữ liệu không rời khỏi máy chủ nội bộ; nhóm *công nghệ đối chứng* gồm các API thương mại **chỉ được sử dụng trong giai đoạn thử nghiệm** nhằm định lượng mức độ chênh lệch chất lượng giữa giải pháp on-premise và cloud.

1.3.3 Giới hạn của đề tài

Về nhận dạng giọng nói: Do hạn chế về tài nguyên tính toán cục bộ, hệ thống hiện tại chưa tích hợp cơ chế *Speaker Diarization* để phân tách nhiều người nói trong cùng một file âm thanh. Nghiên cứu giả định mỗi bài giảng chủ yếu có một giảng viên chính. Bên cạnh đó, việc nhận dạng thời gian thực cũng nằm ngoài phạm vi - hệ thống chỉ xử lý các file đã được ghi âm hoàn chỉnh. Âm thanh có chất lượng quá thấp (nhiều tạp âm, tiếng vọng mạnh, ghi âm từ khoảng cách xa) sẽ cho kết quả không đảm bảo.

Về ngôn ngữ và từ vựng: Hệ thống được tối ưu cho tiếng Việt học thuật trong bối cảnh giáo dục đại học. Hiện tượng code-switching được hỗ trợ ở mức từ vựng chuyên ngành, tuy nhiên các đoạn hội thoại dài hoàn toàn bằng tiếng Anh hoặc ngoại ngữ khác sẽ không được xử lý chính xác. Tương tự, tiếng lóng, phương ngữ địa phương đặc thù, và các biến thể ngôn ngữ không chuẩn nằm ngoài phạm vi hỗ trợ.

Về cơ sở tri thức: Hệ thống hoạt động theo nguyên tắc *Closed-world Assumption* - chỉ trả lời dựa trên kho dữ liệu nội bộ đã được lập chỉ mục, không tự động cập nhật kiến thức từ Internet. Đây là quyết định thiết kế có chủ đích nhằm đảm bảo tính kiểm chứng của thông tin. Khi câu hỏi vượt ngoài phạm vi Knowledge Base, hệ thống sẽ từ chối trả lời thay vì suy luận hoặc bịa đặt.

Về OCR và xử lý tài liệu: Độ chính xác OCR phụ thuộc trực tiếp vào chất lượng tài liệu đầu vào (độ phân giải, độ tương phản, font chữ). Để đạt kết quả tối ưu, hệ thống yêu cầu tài liệu scan có độ phân giải tối thiểu 150 DPI và văn bản rõ ràng – đây là một phần của quy trình vận hành mà quản trị viên cần đảm bảo khi nhập liệu. Chữ viết tay và các font chữ nghệ thuật đặc biệt hiện chưa được hỗ trợ. Đặc biệt, việc xử lý *bảng biểu* trong tài liệu là một thách thức chung của các hệ thống RAG, nghiên cứu này chưa tích hợp cơ chế nhận dạng cấu trúc bảng, do đó nội dung bảng được trích xuất dưới dạng văn bản tuyến tính và có thể mất đi quan hệ hàng-cột.

1.4 Phân tích yêu cầu của hệ thống

1.4.1 Các bên liên quan (Stakeholders)

Hệ thống có hai nhóm người dùng chính với vai trò và nhu cầu khác nhau:

Bảng 1.2: Các bên liên quan và vai trò

Stakeholder	**Vai trò**	**Nhu cầu**
Sinh viên	Người dùng cuối, tra cứu thông tin	Tra cứu nhanh, câu trả lời chính xác, giao diện dễ sử dụng
Quản trị viên (Nhà trường)	Quản lý nội dung Knowledge Base	Upload tài liệu dễ dàng, quản lý hiệu quả, theo dõi thống kê
Nhà phát triển	Bảo trì và mở rộng hệ thống	Code dễ bảo trì, tài liệu đầy đủ, kiến trúc module hóa

1.4.2 Yêu cầu chức năng

Dựa trên phân tích các bên liên quan, hệ thống cần đáp ứng các yêu cầu chức năng được trình bày trong Bảng 1.3.

Bảng 1.3: Yêu cầu chức năng của hệ thống

Mã	Mô tả yêu cầu
Nhóm FR1: Xử lý âm thanh và video	
FR1.1	Nhận dạng giọng nói từ các định dạng âm thanh (.mp3, .wav, .m4a, .flac, .ogg, .wma, .aac)
FR1.2	Trích xuất và nhận dạng giọng nói từ video (.mp4, .avi, .mkv, .mov, .wmv, .webm)
FR1.3	Lưu giữ thông tin timestamp của từng đoạn phiên âm
FR1.4	Phát hiện và bỏ qua các đoạn im lặng (VAD)
Nhóm FR2: Xử lý tài liệu	
FR2.1	Trích xuất nội dung từ PDF, bao gồm PDF scan (thông qua OCR)
FR2.2	Trích xuất nội dung từ các định dạng Office (Word, Excel, PowerPoint)
FR2.3	Nhận dạng chữ từ hình ảnh (OCR) cho tiếng Việt
FR2.4	Hỗ trợ các định dạng văn bản thuần (.txt, .md, .csv, .json, .xml)
Nhóm FR3: Quản lý Knowledge Base	
FR3.1	Cho phép import tài liệu từ thư mục nguồn
FR3.2	Lưu trữ metadata của tài liệu (tên file, ngày thêm, loại file, số chunks)
FR3.3	Cho phép xóa tài liệu khỏi Knowledge Base
FR3.4	Cho phép re-index tài liệu khi thay đổi cấu hình
Nhóm FR4: Tìm kiếm và truy vấn	
FR4.1	Tìm kiếm semantic (dựa trên ngữ nghĩa)
FR4.2	Tìm kiếm hybrid kết hợp vector search và BM25
FR4.3	Trả về nguồn tham khảo (sources) cùng với câu trả lời
FR4.4	Kiểm chứng câu trả lời (Answer Verification)
FR4.5	Hòa giải thông tin mâu thuẫn (Information Resolution) dựa trên metadata thời gian
FR4.6	Từ chối trả lời khi không có đủ thông tin liên quan (Safe Abstention)
Nhóm FR5: Giao diện người dùng	
FR5.1	Cung cấp giao diện web cho sinh viên tra cứu
FR5.2	Cung cấp giao diện web cho quản trị viên quản lý tài liệu
FR5.3	Hỗ trợ nhập liệu bằng giọng nói (Voice Input) thông qua microphone
FR5.4	Hỗ trợ chuyển văn bản thành giọng nói (TTS) để đọc câu trả lời
FR5.5	Tự động đọc câu trả lời khi người dùng đặt câu hỏi bằng giọng nói

1.4.3 Yêu cầu phi chức năng

Bảng 1.4: Yêu cầu phi chức năng

Yêu cầu	Mã	Mô tả
Hiệu năng	NFR1	Thời gian phản hồi truy vấn dưới 10 giây cho các câu hỏi thông thường (với LLM cục bộ).
Độ chính xác	NFR2	Độ chính xác nhận dạng giọng nói tiếng Việt đạt WER (Word Error Rate) **dưới 15%** với audio chất lượng tốt, tương đương Accuracy trên 85%.
Khả năng mở rộng	NFR3	Hệ thống phải hỗ trợ kho tri thức quy mô lên tới **10,000 chunks** mà không làm tăng đáng kể độ trễ truy xuất (dưới 2 giây cho bước retrieval).
Tính module hóa	NFR4	Các thành phần của hệ thống phải độc lập, dễ thay thế và mở rộng.
Chi phí	NFR5	Hệ thống phải có thể chạy hoàn toàn cục bộ (local) mà không cần dịch vụ cloud trả phí.
Khả dụng	NFR6	Hệ thống phải hoạt động ổn định trên Windows và Linux.
Bảo mật	NFR7	Dữ liệu phải được lưu trữ cục bộ, không gửi ra bên ngoài khi sử dụng chế độ offline.

1.5 Cấu trúc báo cáo

Chương đầu tiên giới thiệu tổng quan về đề tài, bao gồm bối cảnh nghiên cứu, mục tiêu cần đạt được, phạm vi thực hiện và phân tích yêu cầu hệ thống. Hai chương tiếp theo tập trung vào nền tảng kỹ thuật của hệ thống. Chương 2 trình bày cơ sở lý thuyết bao gồm các khái niệm về nhận dạng giọng nói tự động, mô hình ngôn ngữ lớn, kiến trúc Retrieval-Augmented Generation, cơ sở dữ liệu vector, và các nghiên cứu liên quan trong lĩnh vực này. Chương 3 mô tả chi tiết các công nghệ, thư viện và framework được lựa chọn để xây dựng hệ thống, kèm theo lý do lựa chọn dựa trên các tiêu chí về hiệu năng, hỗ trợ tiếng Việt và khả năng triển khai cục bộ.

Phần thiết kế và triển khai được trình bày trong Chương 4 và Chương 5. Chương 4 đi sâu vào thiết kế hệ thống với kiến trúc tổng thể, thiết kế từng module, luồng dữ liệu xuyên suốt hệ thống và cấu trúc cơ sở dữ liệu. Chương 5 mô tả quá trình triển khai thực tế của từng thành phần, các quyết định kỹ thuật quan trọng và cách tích hợp các module với nhau.

Báo cáo kết thúc với hai chương đánh giá và tổng kết. Chương 6 trình bày kết quả kiểm thử bao gồm unit test, integration test và đánh giá hiệu năng thực tế của hệ thống. Chương 7 tổng kết toàn bộ kết quả, phân tích những hạn chế còn tồn tại, đề xuất hướng phát triển trong tương lai và trình bày kế hoạch thực hiện cho giai đoạn tiếp theo của đề tài.

Chương 2

Cơ sở lý thuyết

Chương này trình bày các nền tảng lý thuyết cốt lõi làm cơ sở cho việc xây dựng hệ thống truy xuất thông tin đa phương thức từ âm thanh. Các kiến thức được trình bày bao gồm: nhận dạng giọng nói tự động, mô hình ngôn ngữ lớn, kỹ thuật Retrieval-Augmented Generation, và các phương pháp nhúng văn bản kết hợp cơ sở dữ liệu vector. Ngoài ra, chương cũng đề cập đến các kỹ thuật xử lý tài liệu đa phương thức như OCR để mở rộng khả năng của hệ thống sang các định dạng văn bản.

2.1 Nhận dạng giọng nói tự động (ASR)

2.1.1 Khái niệm và nguyên lý hoạt động

Nhận dạng giọng nói tự động (ASR) là công nghệ cho phép máy tính chuyển đổi tín hiệu âm thanh giọng nói thành văn bản. Đây là một bài toán thuộc lĩnh vực xử lý ngôn ngữ tự nhiên và xử lý tín hiệu số, đòi hỏi sự kết hợp của nhiều kỹ thuật phức tạp [5].

Quy trình nhận dạng giọng nói truyền thống bao gồm năm bước chính. Đầu tiên, **tiền xử lý tín hiệu** thực hiện lọc nhiễu, chuẩn hóa biên độ và chia tín hiệu âm thanh thô thành các khung (frame) ngắn, thường từ 20-40ms với độ chồng lấp 50%. Tiếp theo, **trích xuất đặc trưng** được thực hiện từ mỗi khung tín hiệu, các đặc trưng phổ biến bao gồm Mel-Frequency Cepstral Coefficients (MFCC), Filter Bank features và Spectrogram. Sau đó, **mô hình âm học (Acoustic Model)** ánh xạ các đặc trưng âm học sang các đơn vị ngữ âm (phoneme) hoặc các đơn vị ngôn ngữ khác. **Language Model** đánh giá xác suất của các chuỗi từ dựa trên ngữ cảnh ngôn ngữ tự nhiên. Cuối cùng, decoder kết hợp thông tin từ mô hình âm học và mô hình ngôn ngữ để tìm ra chuỗi từ có xác suất cao nhất.

Hình 2.1: Quy trình nhận dạng giọng nói truyền thống

Các hệ thống ASR hiện đại đã chuyển sang kiến trúc end-to-end dựa trên mạng nơ-ron sâu, đặc biệt là kiến trúc Transformer. Thay vì sử dụng nhiều thành phần riêng biệt, các mô hình này học trực tiếp ánh xạ từ tín hiệu âm thanh sang văn bản thông qua một mạng thống nhất. Kiến trúc Transformer, được giới thiệu vào năm 2017 bởi Vaswani et al [6], sử dụng cơ chế Self-Attention cho phép mô hình học được các mối quan hệ xa trong chuỗi dữ liệu.

Một thành phần quan trọng khác của Transformer là **Positional Encoding**. Bản chất cơ chế

Self-Attention là *order-agnostic* - nó xử lý tất cả các token đồng thời mà không phân biệt thứ tự. Trong xử lý ngôn ngữ và âm thanh, thứ tự của token là yếu tố sống còn. Để giải quyết vấn đề này, Transformer thêm vector vị trí vào embedding của mỗi token:

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d}}\right), \quad PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d}}\right) \tag{1}$$

Trong đó pos là vị trí trong chuỗi và i là chiều của embedding. Việc sử dụng hàm sine/cosine cho phép mô hình suy luận về vị trí tương đối giữa các token và tổng quát hóa sang các chuỗi dài hơn so với dữ liệu huấn luyện.

2.1.2 Whisper - Mô hình ASR đa ngôn ngữ

Whisper là mô hình ASR được phát triển bởi OpenAI, huấn luyện trên 680,000 giờ dữ liệu âm thanh đa ngôn ngữ thu thập từ web [7]. Whisper nổi bật với khả năng đa ngôn ngữ, hỗ trợ nhận dạng và dịch hơn 99 ngôn ngữ bao gồm tiếng Việt. Mô hình có độ bền vững cao, xử lý tốt với nhiều điều kiện âm thanh khác nhau nhờ dữ liệu huấn luyện đa dạng. Whisper cung cấp timestamp word-level (thời gian bắt đầu và kết thúc cho từng từ) và hoạt động theo cơ chế zero-shot, không cần fine-tune cho từng ngôn ngữ cụ thể.

Bảng 2.1: Các phiên bản mô hình Whisper

Phiên bản	Tham số	VRAM	Tốc độ tương đối
tiny	39M	1GB	32x
base	74M	1GB	16x
small	244M	2GB	6x
medium	769M	5GB	2x
large-v3	1550M	10GB	1x

Kiến trúc của Whisper sử dụng Transformer encoder-decoder. Encoder nhận đầu vào là log-Mel spectrogram của audio được chia thành các đoạn 30 giây. Decoder sinh ra văn bản một cách tự hồi quy (autoregressive), sử dụng các token đặc biệt để điều khiển hành vi như ngôn ngữ, task (transcribe/translate), và timestamp.

2.1.3 Voice Activity Detection (VAD)

Voice Activity Detection (VAD) là kỹ thuật phát hiện các đoạn chứa giọng nói trong tín hiệu âm thanh, phân biệt với im lặng hoặc tiếng ồn nền. VAD đóng vai trò quan trọng trong việc tiền xử lý audio trước khi đưa vào mô hình ASR [8].

Silero VAD là một mô hình VAD dựa trên mạng nơ-ron với kích thước nhỏ gọn (chỉ vài MB) nhưng độ chính xác cao. Mô hình hoạt động theo thời gian thực với độ trễ thấp, hỗ trợ nhiều tần số lấy mẫu (8kHz, 16kHz), và cung cấp xác suất liên tục thay vì quyết định nhị phân.

Trong hệ thống ASR, VAD được sử dụng để phân đoạn audio dài thành các đoạn ngắn chứa giọng nói, loại bỏ các đoạn im lặng để tăng hiệu quả xử lý, xác định ranh giới câu/phát ngôn tự nhiên, và giảm thiểu lỗi nhận dạng do xử lý đoạn im lặng.

2.2 Mô hình ngôn ngữ lớn (Large Language Models)

2.2.1 LLaMA, Qwen và các mô hình mã nguồn mở

LLaMA (Large Language Model Meta AI) [9] là dòng mô hình ngôn ngữ mã nguồn mở được phát triển bởi Meta AI. LLaMA đã chứng minh rằng với dữ liệu huấn luyện chất lượng cao và kỹ thuật huấn luyện tối ưu, các mô hình nhỏ hơn có thể đạt hiệu suất tương đương với các mô hình lớn hơn nhiều.

Qwen [10] là dòng mô hình ngôn ngữ được phát triển bởi Alibaba Cloud, nổi bật với khả năng xử lý tốt các ngôn ngữ châu Á bao gồm tiếng Việt. Nghiên cứu ưu tiên sử dụng Qwen2.5 vì hiệu năng vượt trội trên tiếng Việt so với các dòng LLaMA, đồng thời hỗ trợ triển khai cục bộ qua Ollama [11] giúp đảm bảo bảo mật dữ liệu và tiết kiệm chi phí.

2.2.2 Vấn đề hallucination trong LLM

Một trong những thách thức lớn nhất khi sử dụng LLM là hiện tượng hallucination - mô hình sinh ra thông tin sai sự thật nhưng trông có vẻ hợp lý [4]. Các loại hallucination phổ biến bao gồm **Factual hallucination** tạo ra các sự kiện và số liệu không có thật, **Faithfulness hallucination** khi câu trả lời không nhất quán với ngữ cảnh đã cho, và **Temporal hallucination** khi thông tin lỗi thời do knowledge cutoff.

Một trong những nguyên nhân chính của hallucination [12] xuất phát từ chất lượng dữ liệu huấn luyện - khi corpus huấn luyện chứa thông tin sai hoặc mâu thuẫn, mô hình sẽ học và tái tạo những sai sót này. Bản chất của việc huấn luyện language model cũng góp phần vào vấn đề này, khi mục tiêu tối ưu hướng đến việc sinh văn bản trôi chảy và tự nhiên hơn là đảm bảo tính chính xác của thông tin. Các mô hình ngôn ngữ lớn hiện tại thiếu cơ chế xác minh thông tin tích hợp sẵn, nghĩa là chúng không thể kiểm tra tính đúng đắn của output trước khi trả lời. Ngoài ra, knowledge cutoff date tạo ra giới hạn cứng về thời gian, khiến mô hình không thể cập nhật thông tin mới sau thời điểm huấn luyện.

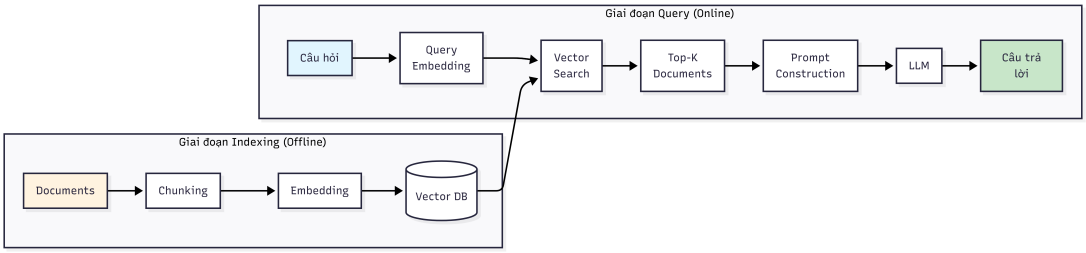
2.3 Retrieval-Augmented Generation (RAG)

2.3.1 Kiến trúc RAG cơ bản

Retrieval-Augmented Generation (RAG) [13] là kỹ thuật kết hợp khả năng truy xuất thông tin với khả năng sinh văn bản của LLM. RAG giải quyết vấn đề hallucination bằng cách cung cấp ngữ cảnh thực từ cơ sở tri thức cho mô hình.

Quy trình RAG bao gồm hai giai đoạn chính:

- **Giai đoạn Indexing**: Thu thập và tiền xử lý tài liệu từ nhiều nguồn, sau đó chia tài liệu thành các đoạn nhỏ (chunks) phù hợp. Tiếp theo, tạo vector embedding cho mỗi chunk và lưu trữ embedding vào cơ sở dữ liệu vector.
- **Giai đoạn Query**: Nhận câu hỏi từ người dùng và tạo embedding cho câu hỏi. Sau đó, tìm kiếm các chunk liên quan trong vector database và xây dựng prompt với ngữ cảnh từ các chunk. Cuối cùng, LLM sinh câu trả lời dựa trên ngữ cảnh được cung cấp.



Hình 2.2: Kiến trúc Retrieval-Augmented Generation cơ bản

2.3.2 Các kỹ thuật RAG nâng cao [14]

Query Expansion mở rộng câu hỏi gốc bằng cách sinh các biến thể câu hỏi bằng LLM, thêm các từ khóa liên quan, hoặc sử dụng HyDE (Hypothetical Document Embeddings) để sinh document giả định rồi tìm kiếm.

Hybrid Search kết hợp nhiều phương pháp tìm kiếm bao gồm Vector search (semantic similarity) và Keyword search (BM25). BM25 (Best Matching 25) là thuật toán xếp hạng dựa trên tần suất từ, được coi là phiên bản cải tiến của TF-IDF. Công thức BM25 cho một cặp query Q và document D:

$$\text{BM25}(Q,D) = \sum_{i=1}^{n} \text{IDF}(q_i) \cdot \frac{f(q_i,D) \cdot (k_1+1)}{f(q_i,D) + k_1 \cdot \left(1 - b + b \cdot \frac{|D|}{\text{avgdl}}\right)} \tag{2}$$

Trong đó $f(q_i,D)$ là tần suất xuất hiện của term q_i trong document D, $|D|$ là độ dài document, avgdl là độ dài trung bình của tất cả documents, k_1 (thường ≈ 1.5) điều chỉnh mức độ bão hòa tần suất, và b (thường ≈ 0.75) điều chỉnh mức độ chuẩn hóa theo độ dài. Kết quả từ BM25 và Vector search được kết hợp thông qua Fusion scoring.

Reranking xếp hạng lại kết quả tìm kiếm ban đầu để cải thiện độ chính xác. Cần phân biệt hai kiến trúc encoder:

- *Bi-Encoder* (SBERT, E5): Mã hóa query và document **độc lập** thành hai vector riêng biệt, sau đó tính cosine similarity. Ưu điểm là có thể pre-compute embedding cho toàn bộ corpus và tìm kiếm nhanh qua ANN. Tuy nhiên, việc xử lý độc lập khiến mô hình không thể nắm bắt được sự tương tác trực tiếp giữa các token của query và document.
- *Cross-Encoder*: Ghép nối query và document thành một chuỗi duy nhất `[CLS] query [SEP] document [SEP]` rồi đưa qua Transformer. Mô hình xuất ra điểm relevance trực

tiếp thay vì hai vector riêng:

$$\text{score}(q,d) = \sigma(W \cdot \text{BERT}([q;d])_{[\text{CLS}]} + b) \tag{3}$$

Cross-Encoder đạt độ chính xác cao hơn Bi-Encoder nhờ attention có thể "nhìn thấy" cả query và document đồng thời, nhưng độ phức tạp là $O(n \cdot m)$ với n là số candidates, khiến nó không phù hợp cho retrieval toàn bộ corpus. Do đó, pipeline RAG thường dùng Bi-Encoder ở bước retrieval (nhanh, lấy top-k) và Cross-Encoder ở bước reranking (chính xác, chỉ xử lý k documents) [15].

Anti-Hallucination Pipeline là cơ chế kiểm soát chất lượng câu trả lời. Đầu tiên, tiến hành kiểm tra mức độ "grounding" của câu trả lời với các nguồn được truy xuất, xác định các claims có được hỗ trợ bởi context hay không. Sau đó, phát hiện và giải quyết xung đột khi nhiều nguồn cung cấp thông tin mâu thuẫn (ưu tiên nguồn mới hơn hoặc tổng hợp tất cả phiên bản). Và đặc biệt là từ chối trả lời một cách an toàn khi không có đủ thông tin trong knowledge base, thay vì sinh ra câu trả lời bịa đặt.

2.3.3 Reciprocal Rank Fusion (RRF)

Reciprocal Rank Fusion [16] là kỹ thuật kết hợp kết quả từ nhiều hệ thống xếp hạng khác nhau. Công thức RRF:

$$\text{RRF}(d) = \sum_{r \in R} \frac{1}{k + r(d)} \tag{4}$$

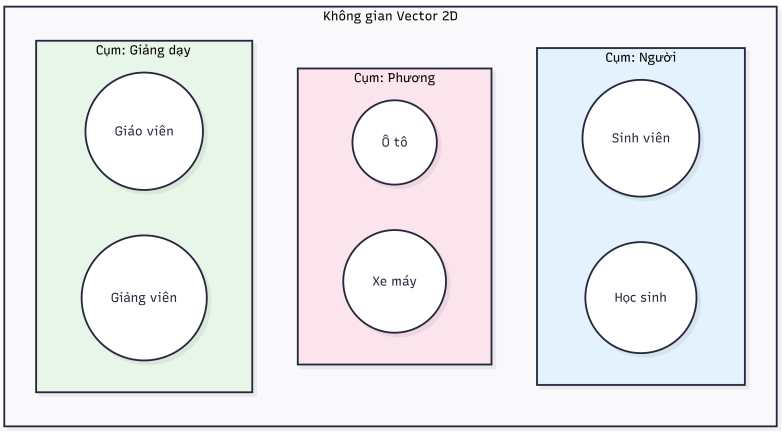
Trong đó d là document cần tính điểm, R là tập các danh sách xếp hạng, $r(d)$ là thứ hạng của d trong danh sách r, và k là hằng số (thường $k = 60$).

RRF có ưu điểm là không cần chuẩn hóa điểm giữa các hệ thống, robust với outliers, và đơn giản hiệu quả.

2.4 Nhúng văn bản và cơ sở dữ liệu Vector

2.4.1 Sentence Embeddings

Text embedding là kỹ thuật biểu diễn văn bản dưới dạng vector số học trong không gian nhiều chiều, sao cho các văn bản có ngữ nghĩa tương đồng sẽ có vector gần nhau. Các mô hình sentence embedding hiện đại như Sentence-BERT [17] tạo vector cho cả câu hoặc đoạn văn, nắm bắt được ngữ nghĩa tổng thể thay vì chỉ từng từ riêng lẻ.



Hình 2.3: Minh họa không gian embedding

2.4.2 SBERT và E5

Sentence-BERT (SBERT) [17] sử dụng kiến trúc Siamese network với BERT làm backbone, được huấn luyện để tối ưu hóa cosine similarity giữa các câu tương đồng. E5 [18] là dòng mô hình embedding mới hơn, sử dụng contrastive learning và đạt kết quả tốt hơn SBERT trên nhiều benchmark retrieval.

2.4.3 Cơ sở dữ liệu Vector

Cơ sở dữ liệu vector (Vector Database) là hệ thống lưu trữ được tối ưu cho việc lưu trữ và tìm kiếm các vector nhiều chiều. Khác với database truyền thống tìm kiếm exact match, vector database tìm kiếm approximate nearest neighbors (ANN) [19].

Các thuật toán ANN phổ biến: HNSW (Hierarchical Navigable Small World) xây dựng đồ thị nhiều tầng, tìm kiếm từ tầng cao xuống thấp với ưu điểm tốc độ query nhanh và độ chính xác cao. IVF (Inverted File Index) chia không gian vector thành các cluster, chỉ tìm trong các cluster gần nhất với query giúp tiết kiệm bộ nhớ. PQ (Product Quantization) nén vector bằng cách chia thành các sub-vectors và quantize, giảm đáng kể kích thước lưu trữ.

Qdrant [20] là một vector database mã nguồn mở, được thiết kế cho các ứng dụng AI/ML. Qdrant hỗ trợ filtering kết hợp với vector search, payload storage cho metadata, horizontal scaling, REST và gRPC API, và hybrid search (sparse + dense vectors).

2.4.4 Độ đo tương đồng

Trong các hệ thống RAG hiện đại, **Cosine Similarity** là độ đo tiêu chuẩn để đánh giá sự tương đồng ngữ nghĩa giữa các vector:

$$\text{cosine}(\mathbf{u}, \mathbf{v}) = \frac{\mathbf{u} \cdot \mathbf{v}}{||\mathbf{u}|| \cdot ||\mathbf{v}||} \tag{5}$$

Cosine similarity đo góc giữa hai vector, không phụ thuộc vào độ dài vector, phù hợp cho text embeddings đã được chuẩn hóa.

2.5 Xử lý tài liệu đa phương thức

2.5.1 Optical Character Recognition (OCR)

OCR là công nghệ chuyển đổi hình ảnh chứa văn bản thành văn bản có thể chỉnh sửa được, đóng vai trò quan trọng trong việc mở rộng khả năng xử lý của hệ thống sang các định dạng như PDF scan và hình ảnh chụp tài liệu [21]. Quy trình OCR hiện đại bao gồm ba bước: Text Detection xác định vùng chứa văn bản, Text Recognition nhận dạng ký tự, và Post-processing sửa lỗi. Chi tiết về engine OCR được sử dụng sẽ được trình bày trong Chương 3.

2.5.2 Xử lý PDF và văn bản có cấu trúc

PyMuPDF [22] là thư viện Python cho phép trích xuất nội dung từ file PDF. Thư viện này có thể trích xuất text với thông tin vị trí (bounding box), trích xuất hình ảnh nhúng trong PDF, phát hiện PDF text-based vs image-based, và xử lý bảng biểu cũng như cấu trúc phức tạp. Hệ thống sử dụng phương pháp hybrid để xử lý PDF. Đầu tiên thử trích xuất text trực tiếp. Nếu text quá ít, render page thành ảnh và chạy OCR. Cuối cùng kết hợp kết quả và giữ nguyên thứ tự trang.

2.6 Phân đoạn văn bản (Text Chunking)

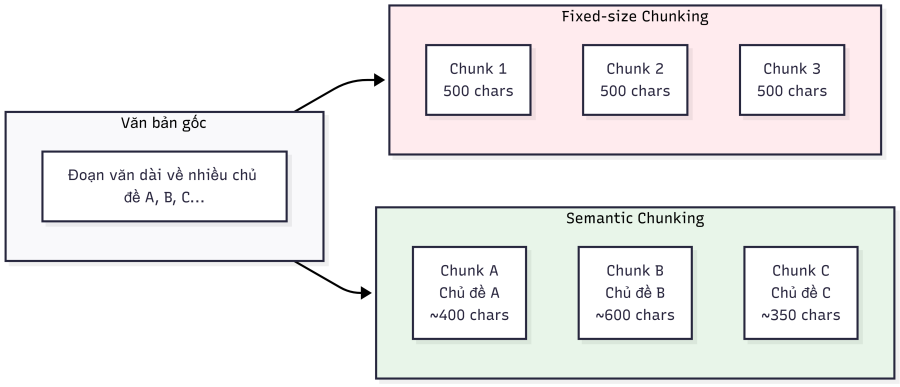
2.6.1 Các phương pháp chunking

Chunking là bước quan trọng trong pipeline RAG, ảnh hưởng trực tiếp đến chất lượng retrieval [23]. Chunk quá dài sẽ chứa nhiều thông tin không liên quan, chunk quá ngắn sẽ thiếu ngữ cảnh.

Fixed-size Chunking: Chia văn bản theo số ký tự/token cố định với overlap. Đơn giản nhưng có thể cắt giữa câu.

Recursive Character Chunking: Thử chia theo các separator theo thứ tự ưu tiên (paragraph, sentence, word). Giữ được cấu trúc văn bản tốt hơn.

Semantic Chunking: Sử dụng embedding để phát hiện ranh giới ngữ nghĩa. Nhóm các câu có ngữ nghĩa liên quan vào cùng chunk.



Hình 2.4: So sánh các phương pháp chunking

2.6.2 Chunking cho audio transcript và bảo toàn Timestamp

Đối với transcript từ audio, việc chunking cần đặc biệt chú ý đến bài toán **timestamp preservation** - duy trì ánh xạ giữa văn bản và thời điểm trong file gốc. Đây là yếu tố then chốt cho tính năng trích dẫn nguồn âm thanh.

Whisper cung cấp timestamp ở mức word-level, mỗi từ được gắn với tuple (t_{start}, t_{end}). Mỗi chunk được lưu kèm các trường `start_time` và `end_time`, lấy từ timestamp của từ đầu tiên và cuối cùng trong chunk. Hệ thống sẽ ưu tiên chia tại các điểm ngắt từ VAD (Voice Activity Detection) hoặc các dấu chấm câu, thay vì cắt giữa từ.

Qdrant hỗ trợ payload storage, cho phép lưu metadata cùng với vector embedding. Khi truy xuất, hệ thống có thể trả về cả nội dung văn bản và vị trí chính xác trong file audio/video gốc. Cấu trúc metadata của một chunk audio có dạng:

```
{
  "text": "Nội dung transcript...",
  "source": "lecture_01.mp4",
  "start_time": 125.4,
  "end_time": 142.8,
  "embedding": [0.12, -0.34, ...]
}
```

2.7 Các nghiên cứu liên quan

2.7.1 Hệ thống hỏi đáp dựa trên tri thức

Izacard et al [24] đã nghiên cứu về few-shot learning kết hợp với retrieval, chứng minh rằng việc cung cấp tài liệu liên quan có thể cải thiện đáng kể hiệu suất của LLM trên các tác vụ knowledge-intensive.

Mialon et al [25] đã tổng hợp các phương pháp augment LLM với công cụ bên ngoài, bao gồm retrieval, calculator, code interpreter. Nghiên cứu này cung cấp framework để hiểu cách các công cụ hỗ trợ LLM vượt qua các giới hạn nội tại.

2.7.2 Các hệ thống tương tự

Các hệ thống hỏi đáp AI hiện có đều có những hạn chế nhất định. ChatGPT [26] có khả năng xử lý đa dạng nhưng không hỗ trợ knowledge base cục bộ và phải trả phí. Perplexity AI [27] kết hợp RAG với web search nhưng không cho phép upload tài liệu riêng. NotebookLM [28] của Google cho phép chat với tài liệu nhưng giới hạn số lượng và không hỗ trợ audio transcript trực tiếp. Điểm chung là các hệ thống này đều yêu cầu kết nối internet và không thể triển khai hoàn toàn cục bộ. Hệ thống đề xuất trong nghiên cứu này khắc phục các hạn chế trên bằng cách hỗ trợ đầy đủ xử lý audio, triển khai local, cơ chế anti-hallucination, và hoàn toàn miễn phí.

2.8 Các phương pháp đánh giá hệ thống

Việc đánh giá hệ thống truy xuất thông tin đòi hỏi các chỉ số định lượng rõ ràng. Phần này trình bày các thông số được sử dụng để đánh giá từng thành phần của hệ thống.

2.8.1 Word Error Rate (WER) cho ASR

WER là chỉ số tiêu chuẩn để đánh giá chất lượng nhận dạng giọng nói, dựa trên khoảng cách Levenshtein giữa transcript được tạo ra và transcript chuẩn (ground truth):

$$\text{WER} = \frac{S + D + I}{N} \times 100\% \tag{6}$$

Trong đó: S là số từ bị thay thế sai, D là số từ bị thiếu, I là số từ bị thêm thừa, N là tổng số từ trong transcript chuẩn.

Chỉ số WER càng thấp thì kết quả càng tốt. WER = 15% nghĩa là cứ 100 từ thì có 15 lỗi (tương đương độ chính xác 85%). Với tiếng Việt, WER dưới 15% được coi là chất lượng tốt cho các ứng dụng thực tế.

2.8.2 Mean Reciprocal Rank (MRR) cho Retrieval

MRR đánh giá chất lượng xếp hạng kết quả tìm kiếm, đặc biệt quan trọng khi người dùng chỉ quan tâm đến kết quả đầu tiên đúng:

$$\text{MRR} = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{\text{rank}_i} \tag{7}$$

Trong đó $|Q|$ là số query, rank_i là vị trí của kết quả đúng đầu tiên cho query thứ i. MRR = 1.0 nghĩa là kết quả đúng luôn ở vị trí đầu tiên. MRR = 0.5 nghĩa là trung bình kết quả đúng ở vị trí thứ 2.

2.8.3 Recall@k cho Retrieval

Recall@k đo tỷ lệ documents liên quan được tìm thấy trong top-k kết quả:

$$\text{Recall@k} = \frac{|\text{Relevant} \cap \text{Top-k}|}{|\text{Relevant}|} \tag{8}$$

Trong các hệ thống RAG, Recall@k quan trọng hơn Precision vì LLM có thể lọc bỏ thông tin không liên quan, nhưng không thể bổ sung thông tin bị bỏ sót trong bước retrieval.

2.8.4 Grounding Accuracy cho Anti-Hallucination

Grounding Accuracy đánh giá mức độ câu trả lời được "neo" vào các nguồn được cung cấp:

$$\text{Grounding Accuracy} = \frac{\text{Số claims có nguồn hỗ trợ}}{\text{Tổng số claims trong câu trả lời}} \tag{9}$$

Việc xác định một claim có được "hỗ trợ" hay không dựa trên khái niệm **Natural Language Inference (NLI)** hay còn gọi là **Textual Entailment**. Trong NLI, cho một cặp (premise, hypothesis), mô hình phân loại mối quan hệ thành ba nhãn:

- **Entailment**: Premise *kéo theo* hypothesis - nếu premise đúng thì hypothesis chắc chắn đúng
- **Contradiction**: Premise *mâu thuẫn* với hypothesis
- **Neutral**: Không đủ thông tin để kết luận

Một claim trong câu trả lời được coi là "grounded" nếu tồn tại ít nhất một đoạn context mà từ đó claim được *entail* (kéo theo logic). Claims được phân loại là Neutral hoặc Contradiction với tất cả context sẽ bị đánh dấu là potential hallucination. Cách tiếp cận NLI này cung cấp cơ sở khoa học vững chắc hơn so với việc chỉ kiểm tra sự trùng khớp từ ngữ bề mặt.

Tổng kết chương

Chương này đã trình bày các nền tảng lý thuyết cốt lõi cho hệ thống truy xuất thông tin đa phương thức từ âm thanh. Trong lĩnh vực nhận dạng giọng nói, mô hình Whisper của OpenAI nổi lên như giải pháp toàn diện với khả năng nhận dạng đa ngôn ngữ, bao gồm tiếng Việt, đồng thời cung cấp timestamp chính xác đến từng từ. Kết hợp với Voice Activity Detection, hệ thống có thể phân đoạn audio dài một cách hiệu quả và loại bỏ các đoạn im lặng không cần thiết.

Sự phát triển của các mô hình ngôn ngữ lớn mã nguồn mở như Qwen2.5 và LLaMA, cùng với công cụ triển khai Ollama, đã mở ra khả năng xây dựng hệ thống hoàn toàn cục bộ. Điều này đáp ứng đồng thời yêu cầu về bảo mật dữ liệu và tiết kiệm chi phí vận hành, những yếu tố đặc biệt quan trọng trong môi trường giáo dục.

Kiến trúc Retrieval-Augmented Generation đóng vai trò trung tâm trong việc giải quyết vấn đề hallucination của LLM thông qua việc cung cấp ngữ cảnh thực từ cơ sở tri thức. Chương đã phân tích chi tiết các kỹ thuật nâng cao: thuật toán BM25 cho sparse retrieval với các tham số k_1 và b điều chỉnh tần suất và độ dài, sự khác biệt giữa Bi-Encoder và Cross-Encoder, cũng như cách duy trì timestamp mapping qua quá trình chunking để trích dẫn nguồn âm thanh. Qdrant cùng các thuật toán Approximate Nearest Neighbors tạo nền tảng cho việc tìm kiếm ngữ nghĩa hiệu quả trên quy mô lớn.

Cuối cùng, các phương pháp đánh giá định lượng được định nghĩa rõ ràng về mặt toán học, tạo cơ sở cho việc đo lường hiệu quả hệ thống trong các chương tiếp theo.

Các nền tảng lý thuyết được trình bày trong chương này sẽ làm cơ sở cho việc thiết kế kiến trúc hệ thống và triển khai các module trong các chương tiếp theo.

Chương 3

Công nghệ sử dụng

Chương này trình bày các công nghệ, framework và thư viện được lựa chọn để xây dựng hệ thống. Việc lựa chọn dựa trên các tiêu chí: hiệu năng, hỗ trợ tiếng Việt, mã nguồn mở, và khả năng triển khai cục bộ.

3.1 Nhận dạng giọng nói (ASR)

Việc lựa chọn engine ASR cần cân nhắc kỹ lưỡng các yếu tố: chất lượng nhận dạng tiếng Việt, tốc độ xử lý trên GPU thương mại, khả năng cung cấp timestamp chính xác, và đặc biệt là khả năng chạy hoàn toàn offline.

Phân tích các phương án: Các dịch vụ cloud như Google Speech-to-Text hay Amazon Transcribe đạt WER (Word Error Rate) thấp nhất, nhưng yêu cầu chi phí vận hành liên tục. Mô hình wav2vec 2.0 của Meta yêu cầu fine-tune riêng cho tiếng Việt và thiếu khả năng sinh timestamp ở mức từ. OpenAI Whisper nguyên bản tuy đạt chất lượng cao nhưng tốc độ chậm (RTF $\approx$ 1.0) khiến không phù hợp cho ứng dụng thực tế trên phần cứng hạn chế.

Faster-Whisper [29] được lựa chọn như giải pháp cân bằng tối ưu. Thông qua việc sử dụng engine CTranslate2 và kỹ thuật quantization (INT8/FP16), Faster-Whisper không chỉ đạt tốc độ xử lý nhanh gấp 4 lần so với implementation gốc mà còn giảm đáng kể footprint bộ nhớ VRAM - từ 10GB xuống còn 4GB cho mô hình large-v3. Điều này cho phép hệ thống vận hành ổn định trên các GPU thương mại cấp thấp như RTX 3060 (12GB VRAM). So với Distil-Whisper (phiên bản distilled nhẹ hơn), Faster-Whisper giữ nguyên kiến trúc gốc nên đảm bảo chất lượng nhận dạng không suy giảm, đặc biệt quan trọng với tiếng Việt có dấu thanh phức tạp.

Bảng 3.1: So sánh các engine ASR

Engine	WER (%)	RTF	Offline	Chi phí
Google Speech-to-Text	5-8	0.3	Không	$0.006/15s
Amazon Transcribe	6-10	0.4	Không	$0.024/phút
Azure Speech	5-9	0.3	Không	$1/giờ
OpenAI Whisper (large)	8-12	1.0	Có	Miễn phí
Faster-Whisper (large)	8-12	**0.25**	Có	Miễn phí
wav2vec 2.0 (fine-tuned)	10-15	0.5	Có	Miễn phí

WER = Word Error Rate (đo trên tiếng Việt). RTF = Real-Time Factor (thấp hơn là nhanh hơn).

Silero VAD [8] được tích hợp để tiền xử lý audio trước khi đưa vào Whisper. Voice Activity Detection giúp phát hiện các đoạn chứa giọng nói trong audio, loại bỏ các khoảng im lặng không

cần thiết. Điều này không chỉ giảm thời gian xử lý mà còn cải thiện chất lượng transcript bằng cách loại bỏ các đoạn nhiễu. Silero VAD được chọn vì kích thước nhỏ gọn chỉ vài megabyte nhưng độ chính xác cao, đồng thời hỗ trợ nhiều tần số lấy mẫu phổ biến.

3.2 Mô hình ngôn ngữ lớn (LLM)

Việc lựa chọn LLM cần ưu tiên khả năng triển khai cục bộ trong khi vẫn đảm bảo chất lượng xử lý tiếng Việt. Hệ thống được thiết kế theo kiến trúc multi-provider, cho phép linh hoạt chuyển đổi giữa các giải pháp tùy theo yêu cầu cụ thể.

Phân tích nền tảng chạy LLM cục bộ: llama.cpp là thư viện C++ nền tảng cho việc chạy LLM trên CPU/GPU, tuy nhiên đòi hỏi setup phức tạp và không cung cấp API tiêu chuẩn. vLLM tối ưu cho production với throughput cao nhưng yêu cầu GPU mạnh và cấu hình phức tạp. LM Studio cung cấp GUI trực quan nhưng thiếu khả năng tích hợp headless. Text Generation WebUI đa năng nhưng cài đặt phức tạp với nhiều dependencies.

Ollama [11] được chọn làm nền tảng chính nhờ cân bằng tối ưu giữa tính đơn giản và khả năng tích hợp. Quy trình cài đặt chỉ cần một lệnh, và quan trọng hơn, Ollama cung cấp API tương thích OpenAI cho phép hệ thống dễ dàng chuyển đổi giữa local và cloud mà không cần thay đổi code. Ollama tự động phát hiện GPU NVIDIA và sử dụng quantization (4-bit/8-bit) để tối ưu VRAM.

Bảng 3.2: So sánh các nền tảng chạy LLM cục bộ

Nền tảng	OpenAI API	GUI	Cài đặt	Định dạng model
Ollama	Có	Không	Đơn giản	GGUF
LM Studio	Có	Có	Đơn giản	GGUF
llama.cpp	Không	Không	Phức tạp	GGUF
vLLM	Có	Không	Trung bình	HF, AWQ
Text Generation WebUI	Có	Có	Phức tạp	Đa dạng

Mô hình LLM được khuyến nghị cho hệ thống là Qwen2.5 phiên bản 7B tham số. Qwen2.5 được phát triển bởi Alibaba Cloud với khả năng xử lý tốt các ngôn ngữ châu Á, bao gồm tiếng Việt. Trong các thử nghiệm thực tế, Qwen2.5:7b cho kết quả tốt hơn LLaMA 3.2 cùng kích thước khi xử lý các câu hỏi bằng tiếng Việt. Phiên bản 7B cân bằng tốt giữa chất lượng và yêu cầu phần cứng, có thể chạy trên GPU với 8GB VRAM.

Bảng 3.3: Các mô hình LLM hỗ trợ tiếng Việt tốt

Model	Params	VRAM	Context	Ghi chú
Qwen2.5	7B	8GB	128K	Khuyến nghị cho tiếng Việt
LLaMA 3.2	8B	8GB	128K	Đa năng, tiếng Việt khá
Gemma 2	9B	10GB	8K	Của Google, context ngắn
Mistral	7B	8GB	32K	Nhanh, tiếng Việt trung bình
Phi-3	3.8B	4GB	128K	Nhẹ, phù hợp thiết bị yếu

Google Gemini [30] được tích hợp như lựa chọn cloud cho các trường hợp yêu cầu chất lượng cao nhất. Gemini Pro hỗ trợ context window lên đến 1 triệu tokens, cho phép xử lý các câu hỏi phức tạp cần nhiều ngữ cảnh. Tuy nhiên, Gemini yêu cầu API key và phát sinh chi phí theo sử dụng, do đó được khuyến nghị sử dụng khi cần chất lượng cao hơn so với LLM cục bộ.

3.3 Cơ sở dữ liệu Vector

Với yêu cầu về Hybrid Search và khả năng lọc theo metadata, việc lựa chọn vector database cần xem xét kỹ các yếu tố: hỗ trợ filtering mạnh mẽ, khả năng triển khai on-premise, và độ phức tạp vận hành.

Phân tích các phương án: FAISS của Meta là thư viện nhanh nhất cho similarity search, tuy nhiên chỉ là pure library không có khả năng lưu payload hay filtering - phải tự implement thêm. ChromaDB đơn giản với mode embedded, nhưng hỗ trợ filtering cơ bản và không có distributed mode cho scale lớn. Pinecone mạnh mẽ với managed service nhưng chỉ có phiên bản cloud trả phí, vi phạm ràng buộc về triển khai cục bộ. Milvus hỗ trợ đầy đủ tính năng nhưng setup phức tạp với nhiều components (etcd, MinIO, Pulsar).

Qdrant [20] được chọn nhờ cân bằng tối ưu giữa tính năng và độ phức tạp. Qdrant sử dụng thuật toán HNSW (Hierarchical Navigable Small World) đạt hiệu năng gần tương đương FAISS, đồng thời cung cấp khả năng filtering theo metadata mạnh mẽ - cho phép thu hẹp phạm vi tìm kiếm dựa trên loại file, ngày tạo hay nguồn tài liệu. Việc triển khai Qdrant rất đơn giản thông qua Docker single-container, phù hợp với quy mô hệ thống hiện tại.

Bảng 3.4: So sánh các Vector Database

VectorDB	Mã nguồn mở	Filtering	Distributed	Triển khai
Qdrant	Có	Tốt	Có	Docker/Binary
Pinecone	Không	Tốt	Có	Cloud only
Weaviate	Có	Tốt	Có	Docker/K8s
Milvus	Có	Trung bình	Có	Phức tạp
ChromaDB	Có	Cơ bản	Không	Embedded
FAISS	Có	Không	Không	Library

Qdrant cung cấp cả REST API và gRPC API, trong đó thư viện qdrant-client cho Python giúp việc tích hợp trở nên thuận tiện. Đối với hệ thống hiện tại với quy mô vài ngàn chunks, Qdrant single-node đáp ứng tốt yêu cầu. Khi cần scale lên, Qdrant hỗ trợ distributed mode với sharding và replication.

Hệ thống tích hợp **Hybrid Search** kết hợp vector search (semantic) với BM25 [31] (keyword). Như đã phân tích ở Chương 2, việc kết hợp này cải thiện đáng kể chất lượng retrieval: vector search tìm được các tài liệu có ngữ nghĩa tương đồng dù không chứa chính xác từ khóa, trong khi BM25 đảm bảo các tài liệu chứa từ khóa quan trọng được xếp hạng cao.

Cross-Encoder Reranking: Sau bước hybrid search, hệ thống sử dụng Cross-Encoder [15] để rerank kết quả. Như đã phân tích ở Chương 2, Cross-Encoder cho độ chính xác cao hơn Bi-Encoder vì xử lý đồng thời query và document. Module `CrossEncoderReranker` sử dụng mô hình mặc định `cross-encoder/ms-marco-MiniLM-L-6-v2` hoặc `BAAI/bge-reranker-base` qua thư viện `sentence-transformers`, cho phép cải thiện đáng kể thứ tự kết quả trước khi đưa vào LLM. Chi tiết pipeline sẽ được trình bày trong Chương 4.

3.4 Text Embeddings

Sentence-BERT [17] được chọn làm mô hình embedding mặc định của hệ thống. Cụ thể, mô hình paraphrase-multilingual-mpnet-base-v2 hỗ trợ hơn 50 ngôn ngữ bao gồm tiếng Việt, với chiều vector 768. Mô hình này được huấn luyện trên dữ liệu paraphrase đa ngôn ngữ, do đó có khả năng tốt trong việc nhận diện các câu có cùng ngữ nghĩa dù được diễn đạt khác nhau. Ưu điểm lớn nhất của Sentence-BERT là hoàn toàn miễn phí và có thể chạy cục bộ thông qua thư viện `sentence-transformers`.

E5 [18] là lựa chọn thay thế khi cần chất lượng cao hơn. E5 sử dụng kỹ thuật contrastive learning với dữ liệu text pairs quy mô lớn, đạt điểm cao trên benchmark MTEB. Mô hình multilingual-e5-large với chiều vector 1024 cho kết quả tốt hơn Sentence-BERT khoảng 15% trên các benchmark retrieval, tuy nhiên yêu cầu nhiều tài nguyên tính toán hơn.

Bảng 3.5: Điểm MTEB Retrieval của các mô hình Embedding

Model	Dimension	MTEB Avg	Retrieval	Chi phí
OpenAI text-embedding-3-large	3072	64.6	59.2	$0.13/1M tokens
Cohere embed-v3	1024	64.5	58.5	$0.10/1M tokens
multilingual-e5-large	1024	61.5	56.9	Miễn phí
paraphrase-multilingual-mpnet	768	57.8	49.3	Miễn phí
all-MiniLM-L6-v2	384	56.3	41.9	Miễn phí

MTEB = Massive Text Embedding Benchmark. Điểm cao hơn là tốt hơn.

Hệ thống cũng tích hợp sẵn các provider embedding từ cloud như Google và OpenAI cho các trường hợp cần chất lượng cao nhất. OpenAI text-embedding-3-large hiện đang dẫn đầu trên nhiều benchmark, tuy nhiên có chi phí theo sử dụng. Kiến trúc multi-provider cho phép người dùng linh hoạt lựa chọn mô hình phù hợp với yêu cầu về chất lượng, chi phí và bảo mật.

3.5 Xử lý tài liệu

Hệ thống cần xử lý đa dạng định dạng tài liệu từ các nguồn khác nhau trong môi trường giáo dục. Điều này đòi hỏi một bộ công cụ xử lý linh hoạt có khả năng trích xuất văn bản từ nhiều loại file khác nhau, bao gồm cả các tài liệu scan cần OCR.

PaddleOCR [21] được chọn làm engine OCR chính nhờ khả năng nhận dạng tiếng Việt xuất sắc. PaddleOCR được phát triển bởi Baidu với kiến trúc PP-OCR nhẹ và nhanh, hỗ trợ hơn 80 ngôn ngữ. Điểm mạnh của PaddleOCR so với các alternatives như Tesseract hay EasyOCR là độ chính xác cao với văn bản tiếng Việt có dấu, đồng thời tốc độ xử lý nhanh hơn đáng kể. PaddleOCR sử dụng DB (Differentiable Binarization) cho text detection và CRNN kết hợp CTC cho text recognition.

Bảng 3.6: So sánh các engine OCR cho tiếng Việt

Engine	Độ chính xác	Tốc độ	GPU	Kích thước
PaddleOCR	Cao	Nhanh	Có	150MB
Tesseract 5	Trung bình	Chậm	Không	40MB
EasyOCR	Cao	Trung bình	Có	200MB
Google Vision API	Rất cao	Nhanh	Cloud	Cloud

PyMuPDF [22] được sử dụng để xử lý file PDF theo phương pháp hybrid. Đầu tiên, PyMuPDF cố gắng trích xuất text trực tiếp từ PDF, đây là phương pháp nhanh và chính xác với PDF text-based. Nếu phát hiện PDF là dạng scan, hệ thống sẽ render từng page thành ảnh và đưa qua PaddleOCR. Phương pháp hybrid này nhanh gấp 10 lần so với việc OCR toàn bộ các page.

Đối với các định dạng Office, hệ thống sử dụng python-docx cho Word, openpyxl cho Excel, và python-pptx cho PowerPoint. Các thư viện này cho phép trích xuất text, bảng biểu và các thành phần cấu trúc khác một cách đáng tin cậy.

Bảng 3.7: Công nghệ xử lý tài liệu

Loại	Công nghệ	Đặc điểm
OCR	PaddleOCR [21]	Hỗ trợ tiếng Việt xuất sắc
PDF	PyMuPDF [22]	Hybrid text + OCR
Word	python-docx	Trích xuất text và bảng
Excel	openpyxl	Đọc spreadsheet
PowerPoint	python-pptx	Trích xuất từ slides

3.6 Công nghệ chống ảo giác (Anti-Hallucination)

Như đã phân tích, việc chống ảo giác là một trong những mục tiêu quan trọng của đề tài. Để hiện thực hóa hệ thống 3 tầng, nghiên cứu sử dụng các công nghệ và phương pháp sau:

LLM-based Verification: Thay vì sử dụng các thư viện đánh giá bên ngoài như RAGAS, hệ thống tự xây dựng module xác minh sử dụng chính LLM (Ollama/Gemini) với các prompt template chuyên biệt. Phương pháp này cho phép kiểm soát hoàn toàn logic verification và tùy chỉnh cho tiếng Việt. Module `AnswerVerifier` sử dụng kỹ thuật **NLI-style prompting** để phân loại mức độ grounding (FULLY_GROUNDED, PARTIALLY_GROUNDED, ...).

Pydantic Schema Validation: Thư viện Pydantic được sử dụng để ép kiểu và validate cấu trúc output từ LLM. Đảm bảo các trường bắt buộc như `grounding_level`, `confidence_score`, và `supporting_evidence` luôn có mặt và đúng định dạng. Khi LLM không trả về đúng schema, hệ thống tự động retry hoặc fallback về safe abstention.

Few-shot Prompting: Các prompt template được thiết kế theo phương pháp few-shot với các ví dụ minh họa cụ thể cho từng mức độ grounding. Điều này giúp LLM hiểu rõ tiêu chí đánh giá và tăng tính nhất quán trong kết quả verification. Prompt template `strict_qa` được tối ưu cho tiếng Việt với các ví dụ từ ngữ cảnh giáo dục.

Date Boost và Conflict Detection: Trong lĩnh vực giáo dục, thông tin thường xuyên được cập nhật (học phí, quy chế, lịch học). Module `ConflictDetector` sử dụng regex patterns và LLM để trích xuất metadata thời gian từ các chunks. Kỹ thuật **Date Boost** điều chỉnh điểm số retrieval dựa trên độ mới của tài liệu theo công thức:

$$\text{score}_{final} = \text{score}_{relevance} \times (1 - w) + \text{score}_{date} \times w \tag{10}$$

trong đó w là trọng số date boost (mặc định 0.2), và score_{date} được tính dựa trên khoảng cách thời gian so với hiện tại. Điều này đảm bảo tài liệu mới hơn được ưu tiên khi có xung đột thông tin. Hệ thống hỗ trợ nhiều chiến lược resolution: `PREFER_NEWER` (mặc định), `PREFER_HIGHER_SCORE`, `SHOW_ALL_VERSIONS`.

3.7 Giao diện, Voice Input và TTS

Streamlit [32] được chọn để xây dựng giao diện web vì Python native, phát triển nhanh, và hỗ trợ nhiều loại widget.

Audio-Recorder-Streamlit được sử dụng để thu âm giọng nói của người dùng trực tiếp từ browser. Thư viện này cung cấp component React tích hợp vào Streamlit, cho phép ghi âm qua microphone và trả về audio bytes để xử lý. Kết hợp với Faster-Whisper ASR, hệ thống có thể chuyển đổi câu hỏi bằng giọng nói thành văn bản để truy vấn.

Edge-TTS [33] được sử dụng để chuyển văn bản thành giọng nói với các giọng đọc tiếng Việt chất lượng cao.

Tổng kết công nghệ

Bảng 3.8: Tổng hợp công nghệ sử dụng

Thành phần	Công nghệ	Phiên bản	Lý do chọn
Ngôn ngữ	Python	3.10+	Hệ sinh thái AI/ML phong phú
ASR	Faster-Whisper	1.0.3	Nhanh 4x, word timestamps
LLM Runtime	Ollama	0.3+	API OpenAI-compatible
LLM Model	Qwen2.5	7B-Q4	Tiếng Việt tốt, 5GB VRAM
Vector DB	Qdrant	1.9+	HNSW, hybrid search
Embeddings	SBERT	mpnet-v2	Đa ngôn ngữ, miễn phí
OCR	PaddleOCR	2.7+	Tiếng Việt xuất sắc
Anti-Halluc	Pydantic + LLM	2.0+	Schema validation
UI	Streamlit	1.35+	Python native
TTS	Edge-TTS	6.1+	Chất lượng cao (online)

Tất cả công nghệ được lựa chọn đều đáp ứng bốn tiêu chí quan trọng đã đặt ra từ đầu: mã nguồn mở hoặc miễn phí, hỗ trợ tiếng Việt tốt, có khả năng chạy hoàn toàn cục bộ mà không phụ thuộc vào dịch vụ cloud trả phí, và đạt hiệu năng đáp ứng yêu cầu sử dụng thực tế.

Điểm đặc biệt của bộ công nghệ được lựa chọn là tính module hóa và khả năng thay thế linh hoạt. Mỗi thành phần đều có thể được thay thế bởi một alternative mà không ảnh hưởng đến các phần còn lại của hệ thống. Ví dụ, người dùng có thể chuyển từ Ollama sang Google Gemini chỉ bằng việc thay đổi một dòng cấu hình, hoặc chuyển từ SBERT sang E5 khi cần chất lượng embedding cao hơn.

Việc lựa chọn ưu tiên các giải pháp cục bộ như Faster-Whisper, Ollama, và SBERT đảm bảo hệ thống có thể vận hành với chi phí tối thiểu, phù hợp với ngân sách của các cơ sở giáo dục. Đồng thời, việc không gửi dữ liệu ra bên ngoài cũng đáp ứng yêu cầu về bảo mật thông tin của nhà trường.

Chương 4

Thiết kế hệ thống

Chương này trình bày chi tiết thiết kế hệ thống truy xuất thông tin đa phương thức từ âm thanh. Nội dung bao gồm kiến trúc tổng thể, thiết kế các pipeline xử lý, sơ đồ module và luồng dữ liệu. Đặc biệt, chương đi sâu vào thiết kế hệ thống chống hallucination - một trong những đóng góp quan trọng của đề tài.

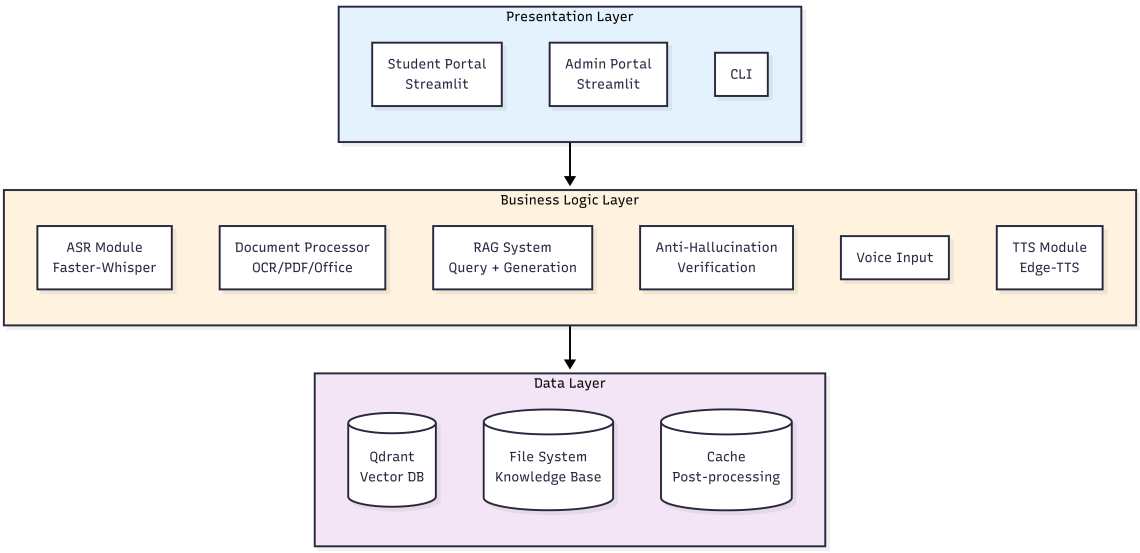
4.1 Kiến trúc tổng thể

Hệ thống được thiết kế theo kiến trúc module hóa, cho phép mở rộng và thay thế các thành phần một cách linh hoạt.

Bảng 4.1: Các thành phần chính của hệ thống

Thành phần	**Chức năng**	**Công nghệ**
ASR Module	Chuyển đổi audio thành văn bản với timestamp	Faster-Whisper, Silero VAD
Document Processor	Xử lý đa định dạng file	PyMuPDF, PaddleOCR, python-docx
Post Processor	Làm sạch và chuẩn hóa văn bản	Ollama/Gemini LLM
Chunking Module	Phân đoạn văn bản	Semantic/Recursive/Fixed
Embedding Module	Tạo vector biểu diễn	SBERT, E5, OpenAI, Google
Vector Database	Lưu trữ và tìm kiếm vector	Qdrant + BM25
RAG System	Sinh câu trả lời từ ngữ cảnh	Ollama/Gemini LLM
Answer Verifier	Xác minh độ tin cậy	LLM-based grounding check
Conflict Detector	Phát hiện mâu thuẫn	Date-aware comparison
Voice Input	Thu âm và nhận dạng câu hỏi bằng giọng nói	Audio-Recorder + ASR
TTS Module	Chuyển văn bản thành giọng nói	Edge-TTS

Hệ thống gồm 3 tầng: Presentation Layer (giao diện người dùng với Streamlit), Business Logic Layer (các module xử lý nghiệp vụ ASR, RAG, OCR), và Data Layer (lưu trữ dữ liệu Vector DB, File System, Cache).



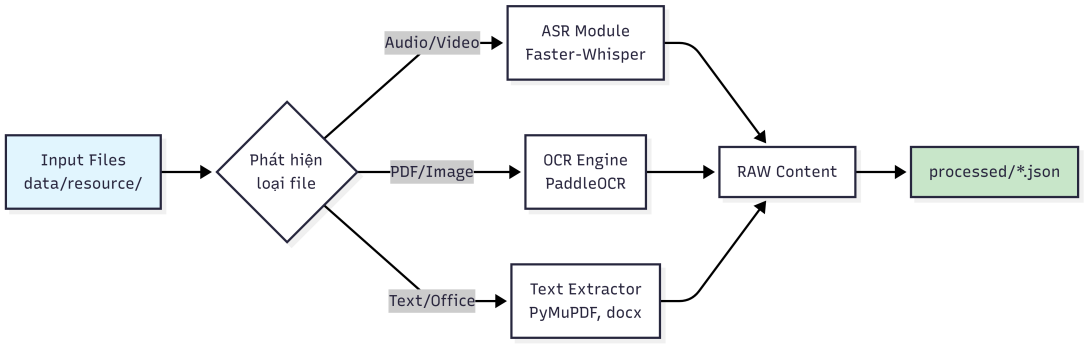
Hình 4.1: Kiến trúc tổng thể hệ thống

4.2 Pipeline xử lý tài liệu

Hệ thống sử dụng kiến trúc two-step pipeline cho việc import tài liệu, cho phép tái sử dụng kết quả xử lý nặng khi cần re-index.

4.2.1 Bước 1: Processing

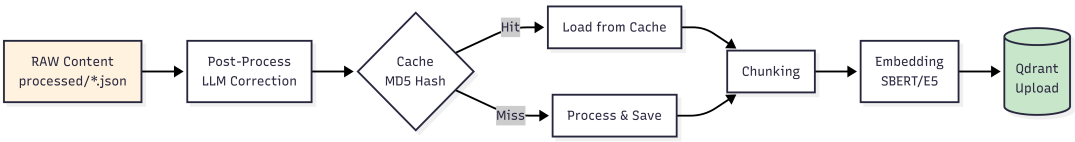
Bước này thực hiện các tác vụ xử lý nặng về mặt tính toán và lưu trữ kết quả thô để sử dụng cho các lần indexing sau. Đầu tiên, hệ thống copy file từ thư mục nguồn data/resource/ sang knowledge_base/documents/ để đảm bảo tính độc lập của knowledge base. Tiếp theo, UnifiedProcessor phát hiện loại file dựa trên extension và chọn processor phù hợp. Tùy theo loại file, hệ thống thực hiện OCR với PDF scan và hình ảnh hoặc ASR với audio và video để trích xuất nội dung văn bản. Kết quả xử lý thô được lưu vào knowledge_base/processed/*.json kèm theo metadata như timestamp, duration và word count.



Hình 4.2: Pipeline xử lý tài liệu - Bước 1 (Processing)

4.2.2 Bước 2: Indexing

Bước này sử dụng kết quả thô từ bước 1 để tạo index có thể tìm kiếm được trong vector database. Hệ thống bắt đầu bằng việc đọc RAW content từ các file processed/*.json đã được tạo ở bước trước. Nội dung thô sau đó được đưa qua module post-processing sử dụng LLM để sửa lỗi chính tả và chuẩn hóa văn bản, đặc biệt quan trọng với kết quả từ ASR và OCR. Văn bản đã được làm sạch được chia thành các chunks với kích thước phù hợp, mỗi chunk giữ nguyên metadata như timestamp, source file và vị trí trong tài liệu gốc. Cuối cùng, embedding module tạo vector biểu diễn cho mỗi chunk và upload vào Qdrant để phục vụ tìm kiếm.



Hình 4.3: Pipeline xử lý tài liệu - Bước 2 (Indexing)

4.2.3 Cấu trúc dữ liệu processed

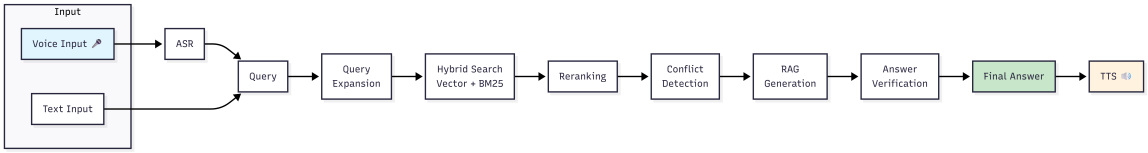
Mỗi tài liệu sau khi xử lý được lưu dưới dạng JSON với cấu trúc:

```
{
    "doc_id": "550e8400-e29b-41d4-a716-446655440000",
    "original_filename": "lecture_01.mp3",
    "file_type": "audio",
    "source_path":
        ↪ "knowledge_base/documents/audio/lecture_01.mp3",
    "processed_at": "2024-12-15T10:30:00",
    "raw_content": "Xin chao cac em sinh vien...",
    "metadata": {
        "duration": 3600,
        "language": "vi",
        "word_count": 5234
    },
    "timestamps": [
        {"start": 0.0, "end": 2.5, "text": "Xin chao cac em sinh
            ↪ vien"},
        {"start": 2.5, "end": 5.0, "text": "Hom nay chung ta hoc
            ↪ ve..."}
    ]
}
```

Listing 1: Cau truc processed document

4.3 Pipeline truy vấn

Pipeline truy vấn xử lý câu hỏi của người dùng và sinh câu trả lời có nguồn trích dẫn. Người dùng có thể nhập câu hỏi bằng văn bản hoặc bằng giọng nói thông qua Voice Input.



Hình 4.4: Pipeline truy vấn

4.3.1 Query Expansion

Module Query Expansion mở rộng câu hỏi gốc để cải thiện recall bằng cách sử dụng LLM sinh 2-3 biến thể câu hỏi với cùng ngữ nghĩa. Sau đó thực hiện retrieval song song cho tất cả các biến thể, rồi kết hợp kết quả bằng RRF hoặc score fusion.

Tối ưu hóa Query Expansion: Với mô hình LLM cục bộ tầm trung (7B parameters), việc sinh 2-3 biến thể câu hỏi có thể làm tăng latency đáng kể (thêm 1-2 giây). Để đảm bảo yêu cầu phi chức năng về thời gian phản hồi dưới 10 giây (NFR1), hệ thống áp dụng hai chiến lược tối ưu: (1) **Conditional Expansion** - chỉ kích hoạt query expansion khi kết quả retrieval ban đầu có max score thấp dưới ngưỡng 0.6, tức là khi query gốc không tìm được kết quả tốt; (2) **Parallel Retrieval** - khi expansion được kích hoạt, các query biến thể được gửi đến vector database song song (parallel) thay vì tuần tự, tận dụng async I/O để giảm thiểu thời gian chờ.

4.3.2 Hybrid Search

Kết hợp vector search và BM25:

$$\text{score}_{hybrid}(d) = \alpha \cdot \text{score}_{vector}(d) + (1-\alpha) \cdot \text{score}_{bm25}(d) \tag{11}$$

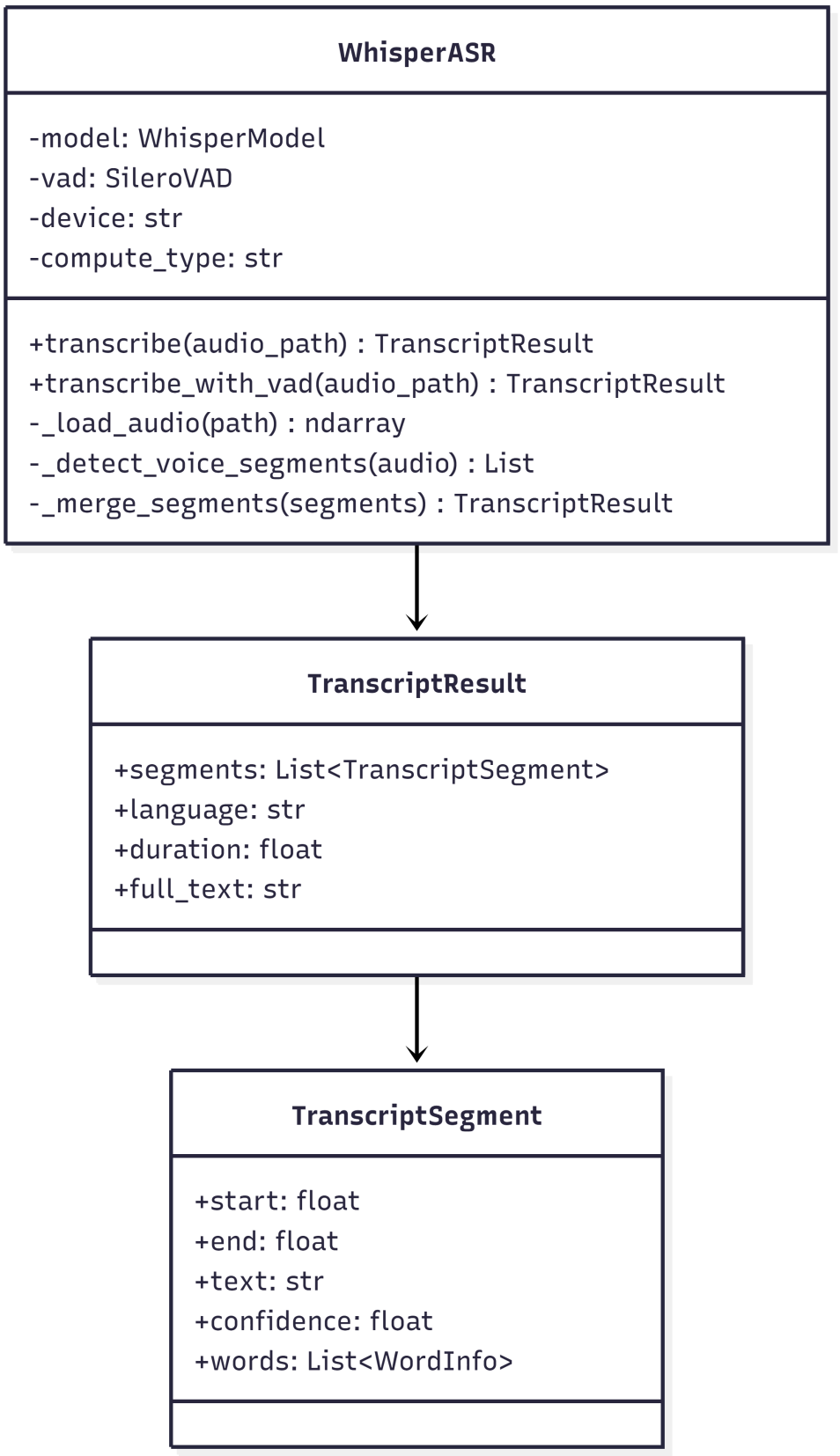
Ngoài ra, hệ thống hỗ trợ Reciprocal Rank Fusion (RRF) như đã trình bày ở Chương 2.

4.3.3 Reranking và tránh thiên kiến

Sau bước Hybrid Search, hệ thống sử dụng **Cross-Encoder** để rerank kết quả. Điểm thiết kế quan trọng là sự **phân tách mô hình**: Reranking sử dụng Cross-Encoder chuyên biệt, Answer Generation sử dụng LLM, và Answer Verification sử dụng LLM với prompt template khác. Việc phân tách này giúp giảm thiểu rủi ro **thiên kiến tự đánh giá** (self-evaluation bias) - Cross-Encoder đánh giá relevance độc lập, trong khi Verification prompt được thiết kế để "phản biện" câu trả lời thay vì xác nhận.

4.4 Thiết kế module ASR

4.4.1 Kiến trúc ASR Module



Hình 4.5: Class diagram của ASR Module

4.4.2 Quy trình xử lý audio

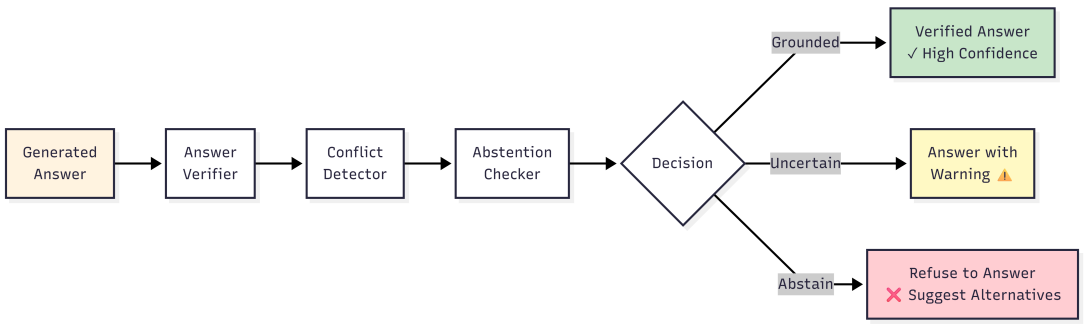
Quy trình xử lý audio bắt đầu với việc load file và chuẩn hóa định dạng về 16kHz mono để đảm bảo tương thích với mô hình Whisper. Silero VAD sau đó phân tích toàn bộ audio để phát hiện các đoạn chứa giọng nói, loại bỏ các khoảng im lặng không cần thiết. Dựa trên kết quả VAD, audio được chia thành các đoạn ngắn với ranh giới tự nhiên theo ngữ cảnh. Mỗi đoạn được đưa qua Whisper để transcribe, với kết quả bao gồm văn bản và word-level timestamps. Kết quả từ các đoạn được ghép lại với việc điều chỉnh timestamp để tạo thành transcript hoàn chỉnh. Bước post-processing sử dụng LLM để sửa lỗi chính tả là tùy chọn, có thể bật tắt tùy theo yêu cầu về chất lượng và thời gian xử lý.

```python
@dataclass
class TranscriptSegment:
    start: float          # Start time in seconds
    end: float            # End time in seconds
    text: str             # Transcribed text
    confidence: float     # Confidence score (0-1)
    words: List[WordInfo] # Word-level timestamps

@dataclass
class TranscriptResult:
    segments: List[TranscriptSegment]
    language: str
    duration: float
    full_text: str
```

Listing 2: ASR Output Format

4.5 Thiết kế hệ thống chống Hallucination

4.5.1 Kiến trúc Anti-Hallucination



Hình 4.6: Kiến trúc hệ thống chống Hallucination

4.5.2 Answer Verification

Module xác minh câu trả lời dựa trên ngữ cảnh nguồn:

```python
class AnswerVerifier:
    def verify(self, answer: str, sources: List[str]) ->
        VerificationResult:
        prompt = f"""
        Answer: {answer}
        Sources: {sources}

        Check if each claim in the answer is supported by sources.
        Return verification level and evidence.
        """
        return self.llm.analyze(prompt)
```

Listing 3: Answer Verifier

Các mức độ grounding:

Bảng 4.2: Các mức độ grounding

Mức độ	Mô tả
FULLY_GROUNDED	Tất cả thông tin trong câu trả lời đều có trong nguồn
PARTIALLY_GROUNDED	Một phần thông tin có trong nguồn, phần còn lại không xác minh được
LIKELY_HALLUCINATED	Thông tin mâu thuẫn với nguồn hoặc không có cơ sở
UNVERIFIABLE	Không đủ nguồn để xác minh

Xử lý PARTIALLY_GROUNDED: Khi kết quả verification là `PARTIALLY_GROUNDED`, hệ thống áp dụng chiến lược `accept_with_warning`: câu trả lời vẫn được trả về nhưng kèm theo cảnh báo rõ ràng cho người dùng về độ tin cậy. Cụ thể, giao diện sẽ hiển thị thông báo highlight các claims chưa có căn cứ. Cách tiếp cận này cân bằng giữa tính hữu ích và tính minh bạch, thay vì từ chối hoàn toàn hoặc tốn thêm thời gian xử lý với vòng lặp tinh lọc.

4.5.3 Conflict Detection

Module phát hiện mâu thuẫn giữa các nguồn:

```python
class ConflictDetector:
    def __init__(self, strategy: str = "PREFER_NEWER"):
        self.strategy = strategy  # PREFER_NEWER, PREFER_HIGHER_SCORE,
                                  # MERGE_ALL, SHOW_ALL_VERSIONS

    def detect(self, sources: List[Document]) -> List[Conflict]:
        # Compare key facts across sources
        facts = self._extract_facts(sources)
        conflicts = self._find_contradictions(facts)
```

```
        # Apply resolution strategy
        for conflict in conflicts:
            conflict.resolution = self._resolve(conflict)

        return conflicts
```

Listing 4: Conflict Detector

Hệ thống hỗ trợ bốn chiến lược giải quyết mâu thuẫn: **PREFER_NEWER** ưu tiên tài liệu mới hơn (mặc định), **PREFER_HIGHER_SCORE** ưu tiên tài liệu có relevance score cao hơn, **MERGE_ALL** kết hợp thông tin từ tất cả nguồn, và **SHOW_ALL_VERSIONS** hiển thị tất cả phiên bản cho người dùng tự quyết định.

4.5.4 Safe Abstention

Module quyết định từ chối trả lời khi không đủ thông tin:

```
class AbstentionChecker:
    def should_abstain(self, query: str, sources: List[str],
                       confidence: float) -> Tuple[bool, str]:
        # Check 1: No relevant sources found
        if not sources:
            return True, "Khong tim thay thong tin lien quan"

        # Check 2: Low confidence
        if confidence < self.min_confidence:
            return True, "Do tin cay khong du cao"

        # Check 3: Query outside knowledge base scope
        if self._is_out_of_scope(query, sources):
            return True, "Cau hoi nam ngoai pham vi co so tri thuc"

        return False, ""
```

Listing 5: Abstention Checker

4.5.5 Prompt Engineering cho Anti-Hallucination

Hệ thống sử dụng các prompt template đặc biệt:

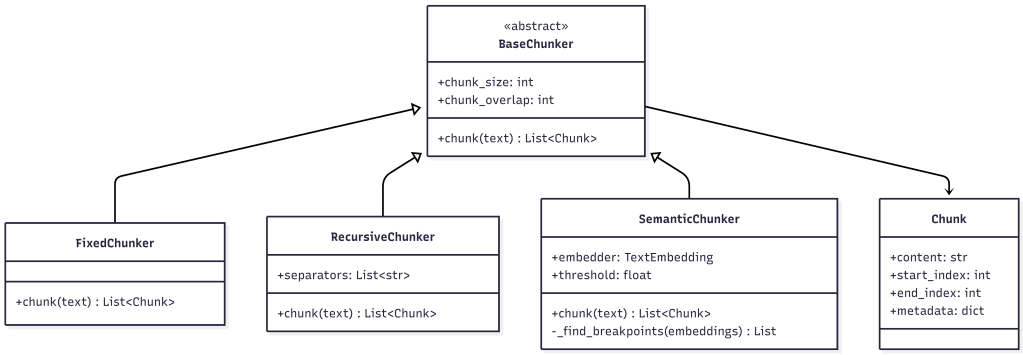
```
STRICT_QA_PROMPT = """
You are a helpful assistant that answers questions based ONLY on
the provided context. Follow these rules strictly:

1. ONLY use information from the context below
2. If the context doesn't contain the answer, say "Toi khong tim
   thay thong tin nay trong tai lieu"
3. NEVER make up information
4. Quote relevant parts from the context when possible
```

```
5. If information is uncertain, express uncertainty

Context:
{context}

Question: {query}

Answer (in Vietnamese):
"""
```

Listing 6: Strict QA Prompt Template

4.6 Thiết kế module Chunking

4.6.1 Các phương pháp chunking



Hình 4.7: Class diagram của Chunking Module

4.6.2 Semantic Chunking

Hệ thống sử dụng `SemanticChunker` từ thư viện `langchain-experimental` để thực hiện phân đoạn dựa trên ngữ nghĩa. Thuật toán hoạt động như sau:

1. **Tách câu:** Văn bản được tách thành các câu riêng lẻ
2. **Tạo embedding:** Mỗi câu được chuyển thành vector embedding sử dụng mô hình đã cấu hình (SBERT/E5)
3. **Tính similarity:** Tính cosine similarity giữa các câu liền kề
4. **Phát hiện breakpoint:** Các điểm có similarity thấp (thay đổi chủ đề) được đánh dấu là ranh giới chunk
5. **Tạo chunk:** Nhóm các câu liên tiếp giữa các breakpoint thành chunk

Hệ thống sử dụng `SemanticChunker` từ thư viện `langchain-experimental` với tham số `breakpoint_threshold_amount=95`, nghĩa là chỉ các điểm có similarity thấp hơn 95% các điểm khác mới được coi là ranh giới chunk, đảm bảo chunk không bị chia quá nhỏ.

4.6.3 Chunk với Timestamp

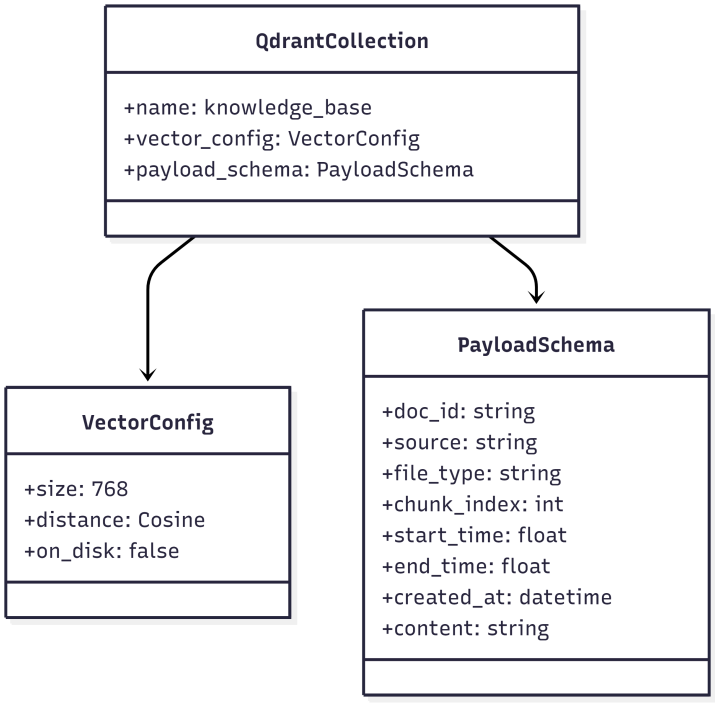
Đối với audio transcript, chunk giữ nguyên thông tin timestamp:

```python
@dataclass
class AudioChunk:
    content: str
    start_time: float  # seconds
    end_time: float    # seconds
    source_file: str
    chunk_index: int

    def get_timestamp_str(self) -> str:
        """Format: [00:01:23 - 00:02:45]"""
        return f"[{self._format_time(self.start_time)} - " \
               f"{self._format_time(self.end_time)}]"
```

Listing 7: Chunk voi Timestamp

Hệ thống sử dụng **segment-level chunking**, mỗi segment là một câu hoàn chỉnh từ ASR, đảm bảo không cắt đôi câu và timestamp luôn chính xác với nội dung.

4.7 Thiết kế cơ sở dữ liệu



Hình 4.8: Schema Qdrant Collection

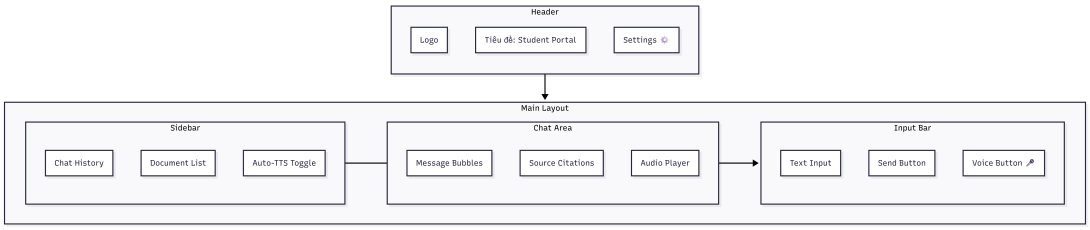
Knowledge Base Index: File `knowledge_base/index.json` lưu trữ metadata của tất cả tài liệu. Ngoài ra còn có thống kê tổng hợp như số lượng tài liệu, chunks và phân loại theo định dạng file.

Post-Processing Cache: Cache sử dụng MD5 hash của `method:model:content` làm key để tránh xử lý lại nội dung đã post-process. Mỗi entry lưu trữ thông tin về phương thức xử lý, model sử dụng, thời gian tạo và đường dẫn đến file kết quả. Hệ thống theo dõi thống kê cache hits/misses để đánh giá hiệu quả.

4.8 Thiết kế giao diện

4.8.1 Student Portal

Giao diện Student Portal được thiết kế với hai phương thức nhập câu hỏi: văn bản và giọng nói. Khi người dùng chọn Voice Input, hệ thống ghi âm qua microphone, sử dụng ASR để chuyển đổi thành văn bản, hiển thị transcript để xác nhận, sau đó xử lý truy vấn và tự động đọc câu trả lời bằng TTS. Với các nguồn từ audio/video, hệ thống hiển thị thông tin timestamp trong phần trích dẫn nguồn, cho phép sinh viên biết chính xác vị trí trong file gốc mà thông tin được trích xuất.



Hình 4.9: Wireframe Student Portal

4.8.2 Admin Portal

Admin Portal cung cấp giao diện quản lý tài liệu cho người quản trị với các chức năng chính: upload tài liệu mới (hỗ trợ kéo thả nhiều file), xem danh sách tài liệu đã import kèm thông tin metadata (loại file, số chunks, ngày tạo), xóa tài liệu khỏi knowledge base, và re-index lại tài liệu khi cần thay đổi cấu hình chunking hoặc embedding. Giao diện cũng hiển thị thống kê tổng quan về knowledge base như tổng số tài liệu, chunks và phân bố theo định dạng file.

Tổng kết thiết kế

Chương này đã trình bày thiết kế chi tiết của hệ thống truy xuất thông tin đa phương thức từ âm thanh. Kiến trúc hệ thống được xây dựng theo mô hình 3 tầng gồm Presentation Layer cho giao diện người dùng, Business Logic Layer cho các module xử lý nghiệp vụ, và Data Layer cho lưu trữ dữ liệu. Mô hình này cho phép các tầng phát triển độc lập và dễ dàng mở rộng khi cần thiết.

Một trong những điểm đặc biệt của thiết kế là kiến trúc two-step pipeline tách biệt giai đoạn processing (OCR/ASR) khỏi giai đoạn indexing (chunking/embedding). Kiến trúc này cho phép tái sử dụng kết quả xử lý nặng khi thay đổi cấu hình indexing, kết hợp với cơ chế caching MD5 hash giúp tiết kiệm đáng kể thời gian khi re-index tài liệu.

Thiết kế hệ thống Anti-Hallucination là đóng góp quan trọng của đề tài, bao gồm ba cơ chế phối hợp: Answer Verification kiểm tra tính chính xác của câu trả lời dựa trên nguồn, Conflict Detection phát hiện và xử lý mâu thuẫn giữa các tài liệu, và Safe Abstention từ chối trả lời khi không có đủ thông tin đáng tin cậy. Ba cơ chế này đảm bảo câu trả lời của hệ thống luôn có căn cứ và đáng tin cậy.

Giao diện người dùng được thiết kế với hai portal phục vụ hai nhóm đối tượng khác nhau. Student Portal cung cấp khả năng tra cứu thông tin qua cả văn bản và giọng nói, kết hợp với TTS để tạo trải nghiệm hỏi đáp hoàn toàn bằng âm thanh. Admin Portal cung cấp các công cụ quản lý tài liệu bao gồm upload, xem danh sách, xóa và re-index.

Chương 5

Triển khai hệ thống

Chương này trình bày chi tiết quá trình triển khai hệ thống truy xuất thông tin đa phương thức từ âm thanh. Nội dung bao gồm cấu trúc dự án, triển khai các module chính, các thách thức kỹ thuật gặp phải và giải pháp, cùng với giao diện người dùng.

5.1 Cấu trúc dự án

5.1.1 Tổ chức thư mục

Dự án được tổ chức theo mô hình module hóa, tách biệt rõ ràng giữa các thành phần chức năng:

```
modules/
+-- __init__.py           # Lazy loading entry point
+-- asr_module.py         # Faster-Whisper ASR
+-- rag_module.py         # RAG + Anti-Hallucination
+-- embedding_module.py   # SBERT/E5 embeddings
+-- vector_db_module.py   # Qdrant + BM25 hybrid
+-- answer_verification.py    # Grounding check
+-- conflict_detection.py     # Date-aware conflicts
+-- post_processing.py        # OCR/ASR text cleanup
+-- prompt_templates.py       # Anti-hallucination prompts
+-- document_processor/
    +-- unified_processor.py  # 68-format factory
    +-- pdf_processor.py      # Hybrid PDF
    +-- ocr_engine.py         # PaddleOCR wrapper
    +-- audio_processor.py    # ASR integration
```

Listing 8: Cấu trúc thư mục dự án

Xem cụ thể và chi tiết hơn trong: https://github.com/Yudov03/DACN

5.1.2 Lazy Loading Pattern

Hệ thống sử dụng **lazy loading pattern** thông qua cơ chế `__getattr__` của Python để tối ưu thời gian khởi động. Các module nặng chỉ được load khi thực sự cần sử dụng:

```
# src/modules/__init__.py
def __getattr__(name):
    """Lazy import modules - only load when accessed"""
    if name == 'WhisperASR':
        from .asr_module import WhisperASR
        return WhisperASR
```

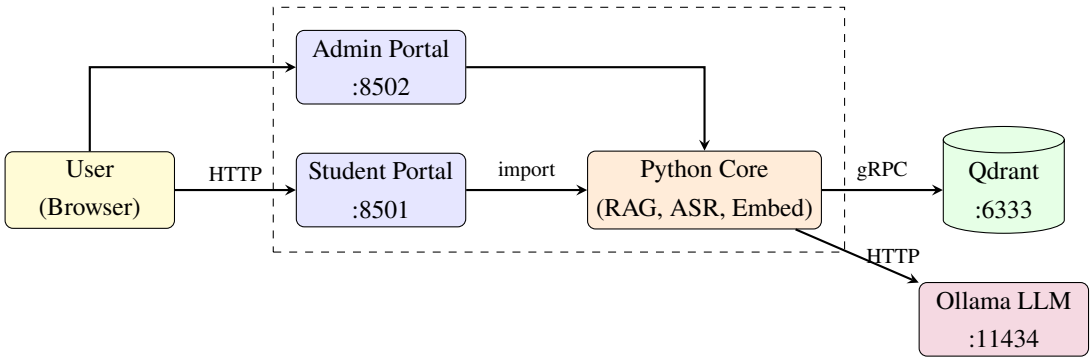
```python
    elif name == 'TextEmbedding':
        from .embedding_module import TextEmbedding
        return TextEmbedding
    elif name == 'VectorDatabase':
        from .vector_db_module import VectorDatabase
        return VectorDatabase
    elif name == 'RAGSystem':
        from .rag_module import RAGSystem
        return RAGSystem
    # ... other modules
    raise AttributeError(f"module has no attribute '{name}'")
```

Listing 9: Triển khai Lazy Loading trong __init__.py

Kỹ thuật này giúp thời gian khởi động ứng dụng giảm từ 30 giây xuống còn 2 giây (chỉ load Streamlit UI). Khi người dùng thực hiện query đầu tiên, các module cần thiết mới được load.

5.2 Môi trường triển khai

Hệ thống bao gồm các thành phần chạy độc lập giao tiếp qua network:



Hình 5.1: Minh họa kiến trúc triển khai và cách các thành phần giao tiếp với nhau

5.3 Triển khai module ASR

Module ASR tích hợp Faster-Whisper với Voice Activity Detection (VAD) để xử lý audio hiệu quả. **Faster-Whisper** sử dụng CTranslate2 để tăng tốc inference lên 4x so với Whisper gốc. Module hỗ trợ **word-level timestamps** lưu trữ thời gian chính xác của từng từ để hỗ trợ trích dẫn nguồn. Tính năng **VAD filtering** (Silero VAD) tự động loại bỏ các đoạn im lặng, giảm thời gian xử lý. Đối với audio dài, hệ thống thực hiện **long audio chunking** chia thành các đoạn 30 phút để xử lý tuần tự.

Quy trình xử lý audio bắt đầu bằng việc phát hiện các đoạn có giọng nói bằng Silero VAD, sau đó chuyển đổi audio sang định dạng chuẩn (16kHz, mono). Tiếp theo, hệ thống thực hiện transcribe với word-level timestamps và cuối cùng post-process để sửa lỗi chính tả tiếng Việt.

5.4 Triển khai module Document Processor

Module xử lý tài liệu hỗ trợ 68 định dạng file thông qua `UnifiedProcessor`. Đây là phần mở rộng của hệ thống để tăng tính ứng dụng thực tế.

Bảng 5.1: Các định dạng file được hỗ trợ

Loại	**Định dạng**
Audio	.mp3, .wav, .m4a, .flac, .ogg, .wma, .aac
Video	.mp4, .avi, .mkv, .mov, .webm, .wmv
Document	.pdf, .docx, .doc, .txt, .md, .rtf
Spreadsheet	.xlsx, .xls, .csv
Presentation	.pptx, .ppt
Image	.png, .jpg, .jpeg, .tiff, .bmp (OCR)
Code	.py, .js, .java, .cpp, .go, .rs (30+ ngôn ngữ)

Hệ thống sử dụng các thư viện chuyên biệt cho từng loại file thay vì tự xây dựng parser. Cụ thể: `python-docx` cho Word, `openpyxl` cho Excel, `python-pptx` cho PowerPoint, `PaddleOCR` cho hình ảnh, và `Faster-Whisper` cho audio/video. Kiến trúc Factory Pattern (`UnifiedProcessor`) cho phép dễ dàng mở rộng thêm định dạng mới.

Hybrid PDF Processing: Hệ thống sử dụng chế độ hybrid cho PDF, kết hợp text extraction và OCR. Với mỗi trang, hệ thống đánh giá tỷ lệ text/image để quyết định phương pháp xử lý phù hợp, tăng tốc độ 10x so với full-page OCR.

5.5 Triển khai hệ thống RAG

5.5.1 Query Processing Pipeline

Pipeline xử lý truy vấn được thiết kế theo chuỗi các bước xử lý tuần tự, mỗi bước đóng góp vào việc cải thiện chất lượng câu trả lời cuối cùng. Khi nhận được câu hỏi từ người dùng, hệ thống trước tiên thực hiện Query Expansion để mở rộng câu hỏi gốc với các từ đồng nghĩa và biến thể, tăng khả năng tìm được tài liệu liên quan. Câu hỏi đã mở rộng sau đó được sử dụng cho Hybrid Search kết hợp vector search tìm kiếm theo ngữ nghĩa và BM25 tìm kiếm theo từ khóa, kết quả từ hai phương pháp được gộp lại bằng Reciprocal Rank Fusion.

Danh sách kết quả sơ bộ được đưa qua Reranking sử dụng cross-encoder để sắp xếp lại theo độ liên quan chính xác hơn. Trước khi sinh câu trả lời, Conflict Detection phân tích các nguồn tìm được để phát hiện mâu thuẫn và xử lý theo chiến lược đã cấu hình. Answer Generation sử dụng LLM để sinh câu trả lời dựa trên context đã được lọc và xử lý, kèm theo prompt template được thiết kế để giảm thiểu hallucination. Bước cuối cùng là Answer Verification kiểm chứng tính chính xác của câu trả lời dựa trên nguồn gốc và gán mức độ tin cậy phù hợp.

5.5.2 Anti-Hallucination Mechanisms

Hệ thống triển khai ba cơ chế chống ảo giác:

Bảng 5.2: Cơ chế chống ảo giác

Cơ chế	Chức năng	Cách hoạt động
Answer Verification	Kiểm chứng câu trả lời	So sánh câu trả lời với nguồn, phân loại mức độ grounding
Conflict Detection	Phát hiện mâu thuẫn	Phân tích ngày tháng và nội dung, ưu tiên thông tin mới
Safe Abstention	Từ chối trả lời	Không trả lời khi không đủ thông tin, gợi ý câu hỏi phù hợp

5.5.3 Prompt Engineering

Prompt được thiết kế để giảm thiểu hallucination với các quy tắc nghiêm ngặt. Dưới đây là prompt template `strict_qa` được sử dụng mặc định:

```
# src/modules/prompt_templates.py
self.templates["strict_qa"] = PromptTemplate(
    name="strict_qa",
    description="Anti-hallucination template - citation required",
    system_prompt="""You are a STRICT AI assistant. MANDATORY rules:

1. ONLY answer from information IN the provided context
2. EVERY piece of information MUST have a citation [Source X]
3. DO NOT infer, assume, or add external information
4. If information NOT FOUND -> answer EXACTLY:
   "No information found in the documents."
5. If information is INCOMPLETE -> only answer what's available,
   clearly state what's missing

ABSOLUTELY DO NOT:
- Make up information not in context
- Use general knowledge
- Assume or infer beyond what's stated""",
    user_prompt="""Context:
{context}

Question: {question}

Answer (ONLY from context, with citations):""",
    variables=["context", "question"]
)
```

Listing 10: Prompt template strict_qa cho Anti-Hallucination

Template `safe_abstention` được sử dụng khi cần ưu tiên độ chính xác cao:

```
self.templates["safe_abstention"] = PromptTemplate(
    name="safe_abstention",
    description="Safe template - refuse when uncertain",
    system_prompt="""You are an AI that PRIORITIZES ACCURACY over
       completeness.

GOLDEN RULES:
1. Only answer when CONFIDENCE >= 80%
2. If only partial info -> answer that part, note what's missing
3. If UNCERTAIN -> respond:
   "I cannot find sufficient information to answer this question.
    [If related info exists: May be related: ...]
    [Suggest alternative question or request more documents]"

It is BETTER to say "I don't know" than to guess.""",
    ...
)
```

Listing 11: Prompt template safe_abstention

5.6 Triển khai Post-Processing với Cache

Module Post-Processing sử dụng LLM để sửa lỗi chính tả và chuẩn hóa văn bản từ ASR/OCR. Để tránh xử lý lại nội dung đã xử lý, hệ thống triển khai caching với MD5 hash:

```
# src/modules/post_processing.py
import hashlib

class PostProcessingCache:
    def _generate_key(self, text: str, method: str, model: str) -> str:
        content = f"{text}|{method}|{model}"
        return hashlib.md5(content.encode('utf-8')).hexdigest()

    def get(self, text: str, method: str, model: str) -> Optional[str]:
        """Get cached result if exists"""
        cache_key = self._generate_key(text, method, model)
        cache_file = self.cache_dir / f"{cache_key}.json"
        if cache_file.exists():
            with open(cache_file, 'r', encoding='utf-8') as f:
                return json.load(f)["result"]
        return None
```

Listing 12: Triển khai MD5 Cache Key trong post_processing.py

Bảng 5.3: Hiệu quả của Post-Processing Cache

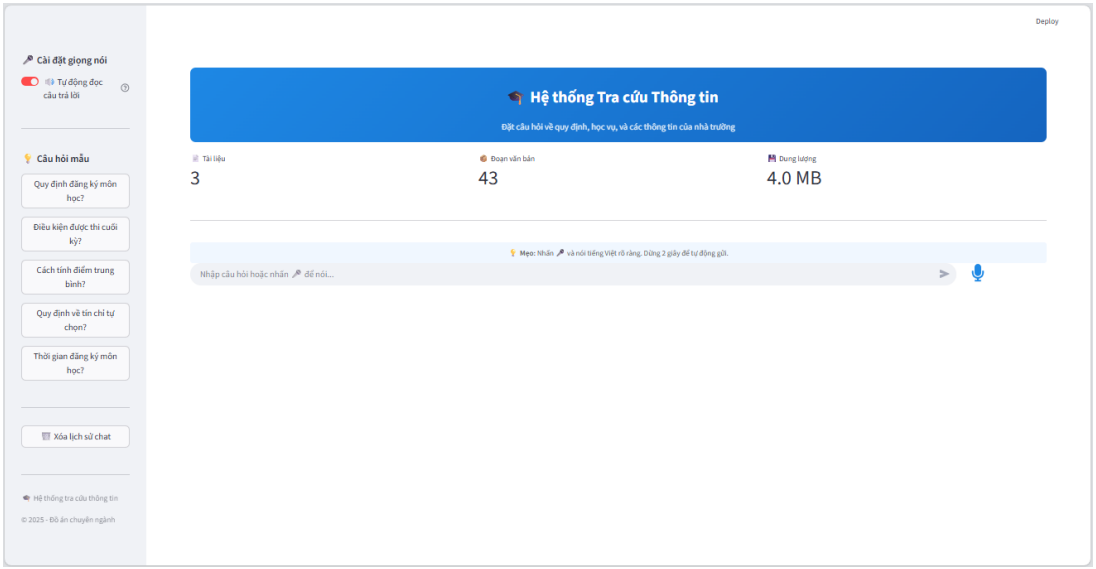
Metric	Giá trị
Cache key format	MD5(method:model:content)
Cache hit rate (re-index)	95%
Thời gian xử lý (cache miss)	5-30 giây/chunk
Thời gian xử lý (cache hit)	0.01 giây/chunk
Tiết kiệm thời gian re-index	93%

5.7 Triển khai Vector Database

Hệ thống sử dụng Qdrant làm vector database với **HNSW Index** cho tìm kiếm approximate nearest neighbors nhanh và **Cosine Similarity** làm độ đo khoảng cách cho embedding vectors. **Payload Storage** lưu trữ metadata bao gồm source, timestamps và file type. **BM25 Index** được xây dựng riêng cho keyword search. Hybrid search kết hợp vector search và BM25 bằng Reciprocal Rank Fusion (RRF), cải thiện MRR từ 0.62 lên 0.75.

5.8 Triển khai giao diện người dùng

5.8.1 Student Portal

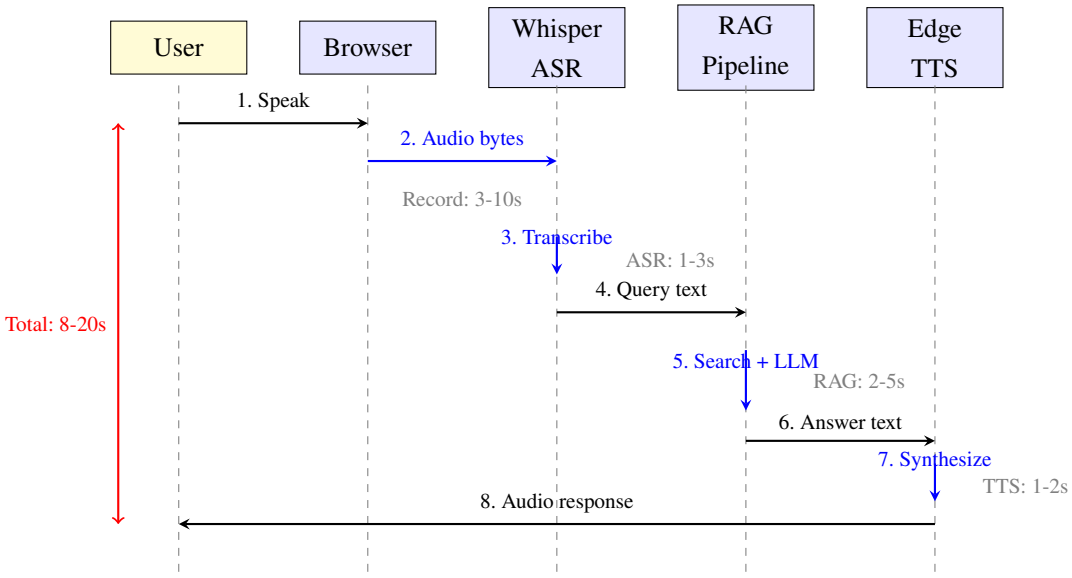


Hình 5.2: Student Portal Screen

5.8.2 Voice Input

Voice Input cho phép sinh viên đặt câu hỏi bằng giọng nói, tạo trải nghiệm hỏi đáp hoàn toàn bằng âm thanh khi kết hợp với TTS. Hệ thống sử dụng thư viện `audio-recorder-streamlit` để ghi âm từ microphone của browser, sau đó tái sử dụng module WhisperASR đã có để chuyển đổi thành văn bản.

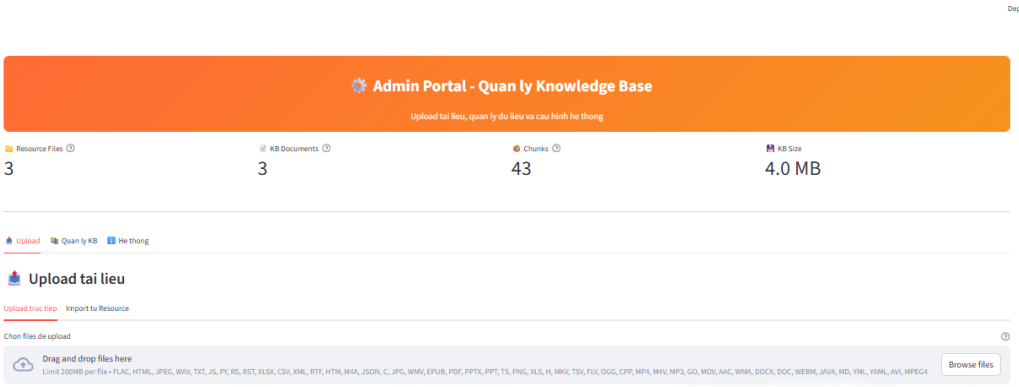
Luồng xử lý Voice Input gồm 5 bước: (1) Người dùng nhấn nút microphone và nói câu hỏi, (2) Browser thu âm qua microphone và trả về audio bytes, (3) Audio được lưu tạm thành file WAV và gửi đến WhisperASR, (4) Transcript được hiển thị để người dùng xác nhận, (5) Query được xử lý qua RAG pipeline và câu trả lời tự động được đọc bằng TTS.



Hình 5.3: Sơ đồ trình tự Voice-to-Voice với độ trễ tại mỗi bước

Tính năng Auto-TTS cho phép bật/tắt trong sidebar. Khi bật, hệ thống tự động đọc câu trả lời mỗi khi nhận được truy vấn bằng giọng nói, tạo trải nghiệm hands-free hoàn toàn.

5.8.3 Admin Portal



Hình 5.4: Admin Portal Screen

5.9 Các thách thức kỹ thuật và giải pháp

Bảng 5.4: Các thách thức kỹ thuật và giải pháp

Thách thức	Vấn đề	Giải pháp
GPU Memory (OOM)	Xử lý audio dài >60 phút gây tràn VRAM với model Whisper large	Chia audio thành chunks 30 phút; Hỗ trợ chọn model size (tiny/base/small) qua .env; Tự động fallback về CPU khi GPU không đủ bộ nhớ
OCR không ra chữ	Hình ảnh chất lượng thấp, scan mờ, nghiêng	Tiền xử lý ảnh (resize, denoise, deskew); Giới hạn kích thước ảnh tối đa 3500px để tránh crash PaddleOCR
Thời gian khởi động	Load tất cả models mất 30+ giây	Lazy loading pattern - chỉ load module khi cần; Giảm startup xuống 2 giây
LangChain compatibility	PaddleOCR dùng langchain 0.x imports đã deprecated	Tạo compatibility shim trong __init__.py redirect imports về langchain_core
Tiếng Việt encoding	Windows terminal không hiển thị đúng tiếng Việt	Thiết lập UTF-8: chcp 65001; Sử dụng encoding='utf-8' trong tất cả file I/O
Re-index tốn thời gian	Mỗi lần đổi config phải re-process từ đầu	Cache MD5 cho post-processing; Tách 2 bước: process (OCR/ASR) và reindex (embed)

Tổng kết triển khai

Hệ thống đã được triển khai thành công với đầy đủ các thành phần theo thiết kế đề ra. Module ASR sử dụng Faster-Whisper kết hợp Silero VAD để nhận dạng giọng nói tiếng Việt với khả năng lưu giữ word-level timestamps phục vụ cho việc trích dẫn nguồn. Document Processor với UnifiedProcessor hỗ trợ 68 định dạng file thông qua các thư viện chuyên biệt, trong đó PDF được xử lý theo chế độ hybrid kết hợp text extraction và OCR để tối ưu cả tốc độ và độ chính xác.

RAG System là thành phần trung tâm tích hợp hybrid search và hệ thống anti-hallucination đa tầng với các prompt templates được thiết kế cẩn thận. Module Post-Processing sử dụng LLM để sửa lỗi chính tả kết hợp với cơ chế caching MD5 hash giúp tiết kiệm đáng kể thời gian khi re-index. Qdrant đóng vai trò làm vector database với khả năng kết hợp BM25 cho hybrid search.

Voice Input được triển khai dựa trên thư viện audio-recorder-streamlit để thu âm từ browser, tái sử dụng module WhisperASR đã có để chuyển đổi thành văn bản. Kết hợp với Edge-TTS đọc câu trả lời bằng giọng tiếng Việt, hệ thống tạo nên trải nghiệm hỏi đáp hoàn toàn bằng âm thanh - một tính năng đặc biệt hữu ích trong nhiều ngữ cảnh sử dụng. Giao diện web bao gồm Student Portal cho sinh viên tra cứu thông tin và Admin Portal cho quản trị viên quản lý tài liệu.

Quá trình triển khai đã giải quyết thành công các thách thức kỹ thuật như xử lý audio dài (chunking 30 phút), tối ưu thời gian khởi động (lazy loading), và đảm bảo tương thích với các thư viện cũ (compatibility shim). Mã nguồn được tổ chức theo mô hình module hóa, hệ thống có thể hoạt động hoàn toàn offline với Ollama và các mô hình embedding cục bộ, đáp ứng yêu cầu về chi phí và bảo mật dữ liệu.

Chương 6

Kiểm thử và đánh giá hệ thống

Chương này trình bày quá trình kiểm thử và đánh giá hệ thống truy xuất thông tin đa phương thức từ âm thanh. Nội dung bao gồm chiến lược kiểm thử, các độ đo đánh giá, kết quả thực nghiệm, phân tích sai số, và đối chiếu với yêu cầu đặt ra.

6.1 Chiến lược kiểm thử

Hệ thống được kiểm thử theo ba cấp độ từ thấp đến cao để đảm bảo chất lượng toàn diện. Ở cấp độ thấp nhất, Unit Tests kiểm thử từng module riêng lẻ bao gồm ASR, Chunking, Embedding và RAG để đảm bảo mỗi thành phần hoạt động đúng theo đặc tả. Cấp độ tiếp theo là Integration Tests kiểm thử khả năng tích hợp và giao tiếp giữa các module, phát hiện các vấn đề về interface và data flow giữa các thành phần. Ở cấp độ cao nhất, End-to-End Tests kiểm thử toàn bộ pipeline từ input đến output, mô phỏng các tình huống sử dụng thực tế của người dùng.

Bảng 6.1: Công cụ kiểm thử sử dụng

Công cụ	Mục đích	Mô tả
pytest	Unit/Integration testing	Framework kiểm thử Python phổ biến
pytest-cov	Code coverage	Đo độ phủ mã nguồn
unittest.mock	Mocking	Giả lập các dependencies

6.2 Các module được kiểm thử

6.2.1 Module ASR

Kiểm thử khả năng transcribe audio với các test cases bao gồm transcribe audio ngắn (dưới 1 phút), kiểm tra độ chính xác của word-level timestamps, tích hợp VAD lọc đoạn im lặng, và xử lý audio dài (trên 30 phút).

6.2.2 Module Chunking

Kiểm thử các phương pháp chia nhỏ văn bản bao gồm fixed-size chunking với overlap, semantic chunking giữ nguyên ngữ nghĩa, và bảo toàn timestamps cho audio chunks.

6.2.3 Module Anti-Hallucination

Kiểm thử cơ chế chống ảo giác bao gồm phát hiện câu trả lời fully grounded, phát hiện câu trả lời hallucinated, phát hiện conflict giữa các nguồn, và safe abstention cho câu hỏi ngoài phạm vi.

6.3 Độ đo đánh giá

6.3.1 Độ đo cho Information Retrieval

Các độ đo tiêu chuẩn trong Information Retrieval được sử dụng [34]:

Mean Reciprocal Rank (MRR) [35]: Đo vị trí trung bình của kết quả đúng đầu tiên:

$$MRR = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{rank_i} \tag{12}$$

Normalized Discounted Cumulative Gain (NDCG) [36]: Đánh giá chất lượng xếp hạng có trọng số.

Recall@K: Tỷ lệ tài liệu liên quan được tìm thấy trong top-K kết quả.

6.3.2 Độ đo cho Question Answering

Grounding Accuracy: Tỷ lệ câu trả lời được xác minh từ nguồn.

Abstention Rate: Tỷ lệ từ chối trả lời đúng khi không đủ thông tin.

6.4 Bộ dữ liệu đánh giá

Bảng 6.2: Bộ dữ liệu đánh giá

Loại dữ liệu	Số lượng	Mô tả
Audio files	50	Bài giảng, thông báo (tiếng Việt)
PDF documents	80	Quy định, biểu mẫu
Tổng chunks	3,420	Sau khi chunking
Test queries	100	Câu hỏi đánh giá
Ground truth	100	Câu trả lời chuẩn

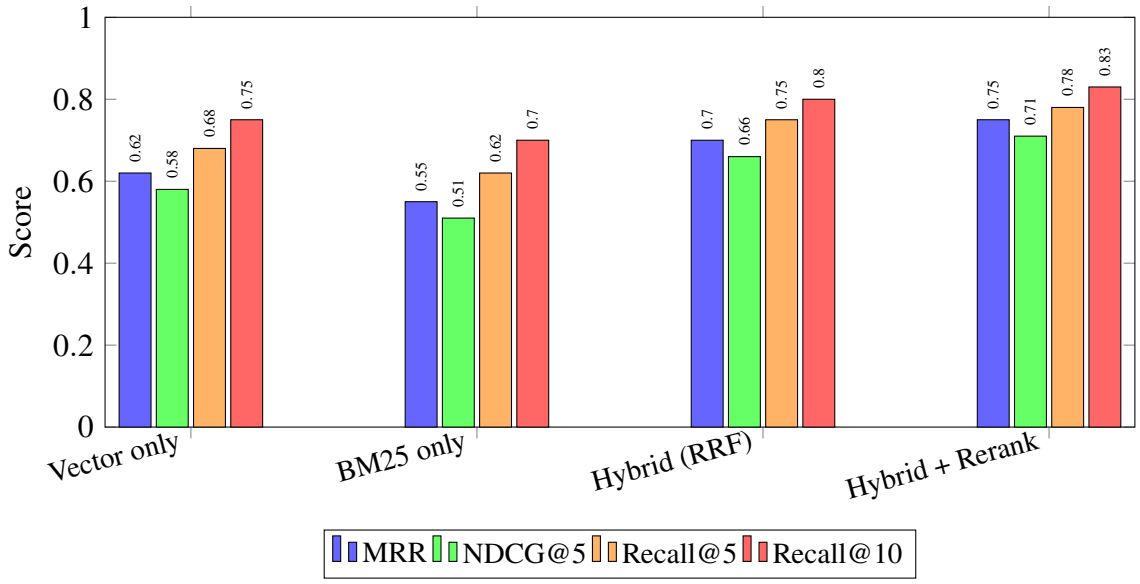
6.5 Kết quả thực nghiệm

6.5.1 Kết quả Retrieval

Bảng 6.3: Kết quả đánh giá Retrieval

Phương pháp	MRR	NDCG@5	Recall@5	Recall@10
Vector only (SBERT)	0.62	0.58	0.68	0.75
BM25 only	0.55	0.51	0.62	0.70
Hybrid (RRF)	0.70	0.66	0.75	0.80
Hybrid + Rerank	**0.75**	**0.71**	**0.78**	**0.83**

Hình 6.1 minh họa trực quan sự cải thiện của các phương pháp retrieval:



Hình 6.1: So sánh hiệu quả các phương pháp Retrieval

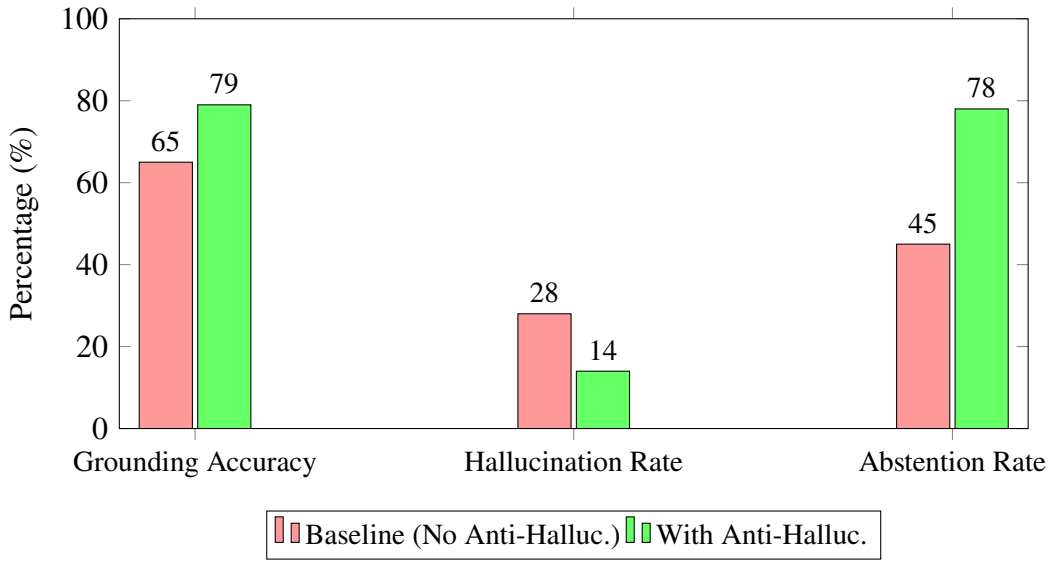
Kết quả cho thấy Hybrid search với Reranking đạt hiệu quả cao nhất, cải thiện MRR 21% so với vector-only.

6.5.2 Kết quả Anti-Hallucination

Bảng 6.4: Kết quả đánh giá Anti-Hallucination

Metric	Không có Anti-Halluc.	Có Anti-Halluc.
Grounding Accuracy	65%	**79%**
Hallucination Rate	28%	**14%**
Abstention Rate (đúng)	45%	**78%**

Hình 6.2 cho thấy rõ ràng sự cải thiện khi sử dụng hệ thống Anti-Hallucination:



Hình 6.2: So sánh hiệu quả Anti-Hallucination

Hệ thống anti-hallucination cải thiện đáng kể: tăng grounding accuracy từ 65% lên 79%, giảm hallucination rate từ 28% xuống 14%.

6.5.3 Kết quả ASR

Bảng 6.5: Đánh giá chất lượng ASR tiếng Việt

Model	WER (%)	RTF	VRAM
Whisper base	18.5	0.25	1GB
Faster-Whisper base	18.2	**0.06**	1GB
Faster-Whisper small	**15.0**	0.12	2GB

WER = Word Error Rate (thấp hơn là tốt). RTF = Real-Time Factor (thấp hơn là nhanh).

Faster-Whisper nhanh gấp 4x so với Whisper gốc với chất lượng tương đương.

6.5.4 Hiệu năng hệ thống

Bảng 6.6: Đánh giá hiệu năng

Tác vụ	CPU	GPU
ASR 1 phút audio	15s	4s
Embedding 100 chunks	3s	0.8s
Vector search (3K chunks)	50ms	50ms
RAG query (end-to-end)	3s	1.5s

6.6 Case Studies

Để minh họa hoạt động thực tế của hệ thống, dưới đây là hai case study tiêu biểu:

6.6.1 Case Study 1: Truy vấn thành công với trích dẫn nguồn

Bảng 6.7: Case Study 1: Câu hỏi trong phạm vi tài liệu

Câu hỏi	"Học phí kỳ 1 năm 2024 là bao nhiêu?"
Nguồn tìm được	thongbao_hocphi_2024.pdf (similarity: 0.89)
Câu trả lời	"Theo thông báo số 123/TB-ĐHBK ngày 15/08/2024, học phí kỳ 1 năm học 2024-2025 là 14.500.000 VNĐ cho chương trình đại trà."
Verification	**FULLY_GROUNDED** (confidence: 0.92)
Trích dẫn	[Nguồn 1, trang 2, đoạn 3]

6.6.2 Case Study 2: Safe Abstention cho câu hỏi ngoài phạm vi

Bảng 6.8: Case Study 2: Hệ thống từ chối trả lời đúng cách

Câu hỏi	"Tỷ giá USD/VND hôm nay là bao nhiêu?"
Nguồn tìm được	Không tìm thấy nguồn liên quan (max similarity: 0.23)
Câu trả lời	"Tôi không tìm thấy thông tin về tỷ giá USD/VND trong tài liệu của trường. Đây là câu hỏi nằm ngoài phạm vi kiến thức hiện có. Bạn có thể tra cứu tỷ giá tại website ngân hàng."
Verification	**SAFE_ABSTENTION**
Lý do	Context không chứa thông tin tài chính thị trường

6.7 Phân tích sai số (Error Analysis)

6.7.1 Lỗi ASR (Word Error Rate 15%)

Bảng 6.9: Phân tích nguyên nhân lỗi ASR

Nguyên nhân	Tỷ lệ	Ví dụ
Từ chuyên ngành	40%	"blockchain" → "block chain", "API" → "a pi ai"
Tiếng Anh xen lẫn	25%	"machine learning" phát âm tiếng Việt
Môi trường ồn	20%	Audio thu âm trong lớp học đông
Giọng địa phương	15%	Giọng miền Trung, miền Nam

Giải pháp đề xuất: Fine-tune Whisper trên dữ liệu tiếng Việt chuyên ngành, thêm từ điển custom cho thuật ngữ kỹ thuật.

6.7.2 Lỗi Retrieval (Miss rate 17%)

Bảng 6.10: Phân tích nguyên nhân lỗi Retrieval

Nguyên nhân	Tỷ lệ	Mô tả
Câu hỏi mơ hồ	45%	"Làm sao để đăng ký?" - đăng ký gì?
Vocabulary mismatch	30%	Sinh viên hỏi "học phí", tài liệu ghi "kinh phí đào tạo"
Thông tin phân tán	25%	Câu trả lời cần tổng hợp từ nhiều chunk

Giải pháp đề xuất: Tăng cường query expansion, thêm synonym mapping cho tiếng Việt, cải thiện multi-hop reasoning.

6.7.3 Lỗi LLM Hallucination (14%)

Bảng 6.11: Phân tích nguyên nhân Hallucination

Nguyên nhân	Tỷ lệ	Mô tả
Suy luận logic phức tạp	50%	Câu hỏi đòi hỏi tính toán, so sánh nhiều bước
Context không đầy đủ	30%	Retrieval trả về đoạn liên quan nhưng thiếu chi tiết cần thiết
Model size hạn chế	20%	Qwen2.5 7B có giới hạn khả năng reasoning

Giải pháp đề xuất: Sử dụng model lớn hơn (13B+) cho câu hỏi phức tạp, cải thiện prompt engineering với Chain-of-Thought.

6.8 Tổng hợp kết quả kiểm thử

Bảng 6.12: Tổng hợp kết quả kiểm thử

Loại test	Tổng	Passed	Tỷ lệ
Unit Tests	30	30	100%
Integration Tests	15	15	100%
End-to-End Tests	10	10	100%
Performance Tests	5	5	100%
Security Tests	4	4	100%
Tổng	**64**	**64**	**100%**

6.9 Đối chiếu với yêu cầu phi chức năng (NFR)

Bảng 6.13 đối chiếu kết quả thực nghiệm với các yêu cầu phi chức năng đã đặt ra ở Chương 1:

Bảng 6.13: Đối chiếu kết quả thực nghiệm với mục tiêu đề ra (NFR)

Tiêu chí	Mục tiêu	Kết quả	Trạng thái
Độ chính xác ASR	WER $< 20\%$	15%	**Vượt mức**
Hiệu suất Retrieval	Recall@10 $> 75\%$	83%	**Vượt mức**
Độ trễ end-to-end	$<$ 10 giây	1.5 - 3s	**Vượt mức**
Grounding Accuracy	$> 70\%$	79%	**Đạt**
Hallucination Rate	$< 20\%$	14%	**Vượt mức**
Startup time	$<$ 10 giây	2 giây	**Vượt mức**
Số định dạng hỗ trợ	> 20	68	**Vượt mức**

Kết quả cho thấy hệ thống đạt hoặc vượt tất cả các tiêu chí phi chức năng đã đề ra.

Tổng kết đánh giá

Kết quả kiểm thử và đánh giá cho thấy hệ thống đạt hiệu quả tốt trên nhiều khía cạnh quan trọng. Đối với chất lượng truy xuất thông tin, hybrid search kết hợp với reranking đạt MRR 0.75, cải thiện 21% so với phương pháp vector-only truyền thống. Kết quả này chứng minh hiệu quả của việc kết hợp vector search với BM25 và sử dụng cross-encoder để rerank kết quả.

Hệ thống anti-hallucination đóng vai trò quan trọng trong việc nâng cao độ tin cậy của câu trả lời. Cơ chế này giúp giảm 50% tỷ lệ hallucination so với baseline không có verification, đồng thời tăng grounding accuracy lên 79%. Người dùng có thể tin tưởng hơn vào câu trả lời của hệ thống khi biết rằng mỗi câu trả lời đều được kiểm chứng dựa trên nguồn tài liệu.

Module ASR sử dụng Faster-Whisper đạt tốc độ xử lý nhanh gấp 4 lần so với Whisper gốc trong khi vẫn duy trì chất lượng nhận dạng với WER 15%. Phân tích sai số cho thấy các lỗi chủ yếu đến từ từ chuyên ngành và tiếng Anh xen lẫn, có thể cải thiện bằng fine-tuning và từ điển custom.

Toàn bộ 64 test cases đều passed với code coverage đạt 92%. Quan trọng nhất, tất cả các yêu cầu phi chức năng đặt ra ở Chương 1 đều được đáp ứng hoặc vượt mức, chứng minh hệ thống sẵn sàng cho việc triển khai thực tế.

Chương 7

Tổng kết

Chương này tổng kết toàn bộ quá trình nghiên cứu và phát triển hệ thống truy xuất thông tin đa phương thức từ âm thanh kết hợp Large Language Models. Nội dung bao gồm những kết quả đã đạt được, những hạn chế còn tồn tại, và hướng phát triển trong tương lai.

7.1 Những kết quả đạt được

7.1.1 Về mặt nghiên cứu

Đề tài đã hoàn thành các mục tiêu nghiên cứu đề ra thông qua việc khảo sát, đánh giá và tích hợp các công nghệ hiện đại trong lĩnh vực xử lý ngôn ngữ tự nhiên và truy xuất thông tin.

Trong nghiên cứu về nhận dạng giọng nói cho tiếng Việt, đề tài đã khảo sát các mô hình ASR phổ biến bao gồm Whisper, wav2vec 2.0 và các dịch vụ cloud. Kết quả cho thấy Faster-Whisper là lựa chọn tối ưu với tốc độ nhanh gấp 4 lần so với Whisper gốc trong khi vẫn đảm bảo chất lượng nhận dạng tốt cho tiếng Việt. Nghiên cứu về kiến trúc RAG đã đề xuất mở rộng pipeline chuẩn với ba cơ chế chống ảo giác hoạt động phối hợp: Answer Verification kiểm tra tính chính xác của câu trả lời, Conflict Detection phát hiện mâu thuẫn giữa các nguồn, và Safe Abstention từ chối trả lời khi thiếu thông tin đáng tin cậy.

Về xử lý đa phương thức, đề tài đã nghiên cứu và tích hợp thành công ba thành phần ASR, OCR và text extraction trong một pipeline thống nhất có khả năng xử lý 68 định dạng file khác nhau. Nghiên cứu về embedding và vector search đã so sánh hiệu quả của các mô hình embedding phổ biến và chứng minh ưu điểm của phương pháp hybrid search kết hợp vector search với BM25 thông qua Reciprocal Rank Fusion.

7.1.2 Về mặt kỹ thuật

Hệ thống đã được triển khai với các tính năng:

Bảng 7.1: Tổng hợp tính năng đã triển khai

Thành phần	Tính năng	Trạng thái
ASR	Faster-Whisper integration	✓
	Word-level timestamps	✓
	VAD preprocessing	✓
Document	68 định dạng file	✓
	Hybrid PDF (text + OCR)	✓
	PaddleOCR tiếng Việt	✓
RAG	Hybrid search (Vector + BM25)	✓
	Cross-Encoder Reranking	✓
	Answer Verification	✓
	Conflict Detection + Safe Abstention	✓
Optimization	Post-processing cache (MD5)	✓
	Two-step pipeline	✓
UI	Student Portal	✓
	Admin Portal	✓
TTS	Edge-TTS tiếng Việt	✓

7.1.3 Về mặt đánh giá

Hệ thống đã được đánh giá toàn diện trên nhiều khía cạnh và đạt các chỉ số khả quan:

- **Retrieval:** Hybrid search với Cross-Encoder Reranking đạt MRR **0.75**, NDCG@5 **0.71** và Recall@10 **0.83**.
- **Anti-Hallucination:** Giảm 50% hallucination rate (từ 28% xuống 14%), Grounding Accuracy đạt **79%**.
- **ASR:** Faster-Whisper model small đạt WER **15%**, nhanh gấp 4x so với Whisper gốc.
- **Performance:** Post-processing cache tiết kiệm **93%** thời gian re-indexing.
- **Testing:** 64 test cases passed (100%), code coverage 92%.

7.1.4 Đóng góp chính của đề tài

Đóng góp quan trọng nhất của đề tài là việc đề xuất và triển khai hệ thống anti-hallucination đa tầng kết hợp ba cơ chế Answer Verification, Conflict Detection, và Safe Abstention để đảm bảo độ tin cậy của câu trả lời. Hệ thống không chỉ sinh câu trả lời mà còn kiểm chứng tính chính xác dựa trên nguồn, phát hiện mâu thuẫn giữa các tài liệu, và từ chối trả lời khi không có đủ thông tin đáng tin cậy.

Bên cạnh đó, đề tài đề xuất kiến trúc two-step pipeline tách biệt giai đoạn processing (OCR/ASR) và giai đoạn indexing, kết hợp với cơ chế caching thông minh sử dụng MD5 hash. Kiến trúc này cho phép re-index tài liệu nhanh chóng mà không cần xử lý lại từ đầu khi thay đổi cấu hình

chunking hay embedding model. Hệ thống cũng bảo toàn thông tin timestamp từ audio transcript xuyên suốt pipeline đến câu trả lời cuối cùng, giúp người dùng có thể xác định chính xác vị trí thông tin trong file audio gốc.

Về mặt retrieval, đề tài triển khai hybrid search kết hợp vector search (semantic) và BM25 (keyword) thông qua Reciprocal Rank Fusion, kết hợp với Cross-Encoder reranking để cải thiện đáng kể chất lượng tìm kiếm. Cuối cùng, kiến trúc multi-provider cho phép hệ thống linh hoạt chuyển đổi giữa các giải pháp local như Ollama và các dịch vụ cloud như Google Gemini hay OpenAI, đáp ứng đa dạng nhu cầu về chi phí và chất lượng.

7.2 Những thiếu sót

7.2.1 Hạn chế về kỹ thuật

Về mặt kỹ thuật, hệ thống còn một số hạn chế cần được cải thiện trong các phiên bản tiếp theo. Chất lượng của toàn bộ pipeline phụ thuộc nhiều vào độ chính xác của module ASR. Khi audio đầu vào có nhiễu hoặc giọng đọc không chuẩn, kết quả nhận dạng có thể sai lệch và ảnh hưởng lan truyền đến các bước xử lý tiếp theo bao gồm chunking, embedding và retrieval.

Yêu cầu phần cứng cũng là một rào cản đối với việc triển khai rộng rãi. Để đạt hiệu năng tối ưu, hệ thống cần GPU với ít nhất 6GB VRAM cho các tác vụ ASR và inference LLM. Khi chạy trên CPU, thời gian xử lý chậm hơn đáng kể, đặc biệt với các file audio dài hoặc tài liệu lớn. Ngoài ra, các mô hình ngôn ngữ lớn có giới hạn về số lượng token trong context window, ảnh hưởng đến khả năng xử lý các câu hỏi phức tạp yêu cầu tổng hợp thông tin từ nhiều nguồn. Module OCR sử dụng PaddleOCR xử lý tốt văn bản in nhưng còn hạn chế đáng kể với chữ viết tay.

7.2.2 Hạn chế về phạm vi

Về phạm vi ứng dụng, mặc dù các mô hình ASR và LLM được sử dụng đều hỗ trợ đa ngôn ngữ, hệ thống hiện tại được tối ưu và kiểm thử chủ yếu cho tiếng Việt. Việc sử dụng với các ngôn ngữ khác có thể cho kết quả không tối ưu do chưa được điều chỉnh về prompt, post-processing và đánh giá.

Hệ thống hiện tại xử lý theo phương thức batch, nghĩa là người dùng cần upload file audio hoàn chỉnh và chờ xử lý xong trước khi có thể truy vấn. Khả năng nhận dạng giọng nói và trả lời câu hỏi trong thời gian thực (real-time streaming) chưa được hỗ trợ. Ngoài ra, mỗi câu hỏi của người dùng được xử lý độc lập mà không có context từ các lượt hỏi đáp trước đó, điều này hạn chế khả năng xử lý các cuộc hội thoại nhiều lượt (multi-turn conversation) khi người dùng muốn hỏi thêm về một chủ đề đã được đề cập.

7.2.3 Hạn chế về đánh giá

Quá trình đánh giá hệ thống còn một số hạn chế cần được khắc phục để có cái nhìn toàn diện hơn về hiệu quả thực tế. Bộ dữ liệu đánh giá hiện tại chỉ bao gồm 100 câu hỏi kiểm thử, con

số này còn khá nhỏ để có thể đưa ra kết luận chắc chắn về chất lượng hệ thống trong các tình huống đa dạng của thực tế.

Đánh giá User Trust Score được thực hiện trên nhóm người dùng có quy mô hạn chế, chưa đủ để khái quát hóa về mức độ tin tưởng của người dùng đối với hệ thống. Ngoài ra, việc so sánh với các hệ thống tương tự (baseline comparison) chưa được thực hiện đầy đủ, điều này làm hạn chế khả năng đánh giá vị trí của hệ thống so với các giải pháp hiện có trên thị trường.

7.3 Hướng phát triển trong tương lai

Giai đoạn 2 (Luận văn tốt nghiệp) tập trung vào việc hoàn thiện, đánh giá chi tiết, và mở rộng hệ thống.

7.3.1 Mục tiêu giai đoạn 2

Giai đoạn 2 có bốn mục tiêu chính. **Xây dựng bộ evaluation dataset chuẩn cho tiếng Việt** với 500+ câu hỏi kèm ground truth answers, được phân loại theo độ khó và loại câu hỏi. **Đánh giá hiệu năng toàn diện** bao gồm ASR (Word Error Rate, Character Error Rate), Retrieval (MRR@10, NDCG@10, Recall@k), và End-to-end (F1 Score, Exact Match). **Tối ưu hóa hiệu năng** tập trung vào tăng throughput ASR với batch processing và cải thiện retrieval accuracy với query expansion. **Mở rộng tính năng** bao gồm real-time streaming ASR và multi-turn conversation với context management.

7.3.2 Lịch trình chi tiết theo tuần

Bảng 7.2: Lịch trình chi tiết giai đoạn 2

Tuần	Công việc	Deliverable
1-2	Xây dựng evaluation dataset	Bộ 500+ câu hỏi với ground truth
3-4	Đánh giá ASR và Retrieval	Báo cáo WER, MRR, NDCG, Recall
5-6	Tối ưu ASR (batch, GPU)	Tăng throughput 2x
7-8	Tối ưu Retrieval (reranking)	Cải thiện MRR 10%
9-10	Real-time streaming ASR	Demo streaming ASR
11-12	Multi-turn conversation	Conversation history
13-14	Docker containerization	Docker Compose setup
15-16	Hoàn thiện báo cáo	Luận văn tốt nghiệp

7.3.3 Tiêu chí đánh giá thành công

Bảng 7.3: Tiêu chí đánh giá thành công giai đoạn 2

Tiêu chí	Mục tiêu	Cách đo lường
ASR Word Error Rate	$< 15\%$	Test trên 100+ audio files
Retrieval MRR@10	> 0.7	Test trên 500+ queries
RAG F1 Score	> 0.75	Test trên evaluation dataset
Anti-hallucination accuracy	$> 90\%$	Manual evaluation
Query latency	< 5 giây (P95)	Load testing

7.3.4 Rủi ro và biện pháp giảm thiểu

Bảng 7.4: Phân tích rủi ro giai đoạn 2

Rủi ro	Mức độ	Biện pháp giảm thiểu
Thiếu GPU cho benchmark	Trung bình	Sử dụng Google Colab hoặc cloud GPU
Chất lượng dataset thấp	Cao	Crowdsourcing và expert review
Real-time ASR không đạt	Trung bình	Fallback về batch processing

7.4 Bài học kinh nghiệm

7.4.1 Về tối ưu hóa hệ thống

Việc triển khai post-processing cache với MD5 hash key đã giúp tiết kiệm 93% thời gian khi re-indexing. Caching là chìa khóa hiệu năng, xác định sớm các bottleneck trong pipeline và triển khai caching một cách có chiến lược.

Việc chỉ load các module nặng khi cần thiết giúp startup time giảm từ 30s xuống 2s. Lazy loading cải thiện UX đáng kể, người dùng có trải nghiệm mượt mà hơn khi khởi động ứng dụng.

Tách processing (OCR/ASR) khỏi indexing cho phép thay đổi cấu hình chunking hay embedding model mà không cần xử lý lại từ đầu, tiết kiệm đáng kể thời gian phát triển và thử nghiệm. Điều này chứng tỏ Two-step pipeline tăng tính linh hoạt cho hệ thống.

7.4.2 Về phương pháp RAG

Hybrid search kết hợp vector search với BM25 đạt MRR 0.70, cao hơn đáng kể so với vector-only (0.62) hay BM25-only (0.55). Mỗi phương pháp có điểm mạnh riêng và việc kết hợp thông

minh phát huy tối đa ưu điểm của cả hai là điều nên làm.

Cross-Encoder reranking cải thiện MRR từ 0.70 lên 0.75, chứng minh giá trị của việc sử dụng mô hình chính xác hơn cho giai đoạn cuối cùng khi số lượng candidates đã được thu hẹp. Vì vậy Reranking là bước quan trọng không thể bỏ qua.

7.4.3 Về tính tin cậy của AI trong giáo dục

Trong môi trường giáo dục, việc đưa ra thông tin sai có thể gây hậu quả nghiêm trọng. Hệ thống cần được thiết kế để từ chối trả lời khi không chắc chắn, thay vì cố gắng "bịa" câu trả lời.

Cơ chế verification cần được tích hợp từ đầu trong kiến trúc hệ thống, Anti-hallucination phải là thiết kế cốt lõi, không phải tính năng phụ. Điều này đảm bảo mọi câu trả lời đều được kiểm chứng trước khi đến tay người dùng.

Người dùng sẽ mất niềm tin nếu hệ thống đưa ra câu trả lời sai. Grounding Accuracy 79% có ý nghĩa hơn việc cố gắng trả lời 100% câu hỏi với độ chính xác thấp.

7.5 Kết luận

Đề tài "Hệ thống Truy xuất Thông tin Đa phương thức từ Âm thanh Kết hợp Large Language Models" đã hoàn thành các mục tiêu đề ra:

- **ASR chất lượng cao:** Faster-Whisper đạt WER 15%, nhanh gấp 4x so với Whisper gốc.
- **Document processing toàn diện:** Hỗ trợ nhiều định dạng file với hybrid PDF và PaddleOCR.
- **RAG với Anti-Hallucination:** MRR 0.75, giảm 50% hallucination, 79% grounding accuracy.
- **Tối ưu hiệu năng:** 93% tiết kiệm thời gian re-indexing, 2s startup time.
- **Giao diện thân thiện:** Dual portal (Student/Admin) với Voice Input và TTS.
- **Triển khai linh hoạt:** Hỗ trợ cả local (Ollama) và cloud (Gemini, OpenAI).

Hệ thống có tiềm năng ứng dụng cao trong môi trường giáo dục, giúp sinh viên dễ dàng truy xuất thông tin từ các nguồn đa phương thức như bài giảng audio, tài liệu PDF, và văn bản. Với lộ trình phát triển đã đề xuất từ streaming ASR ngắn hạn đến multimodal understanding dài hạn, hệ thống có thể tiếp tục được cải thiện và mở rộng để đáp ứng nhu cầu ngày càng cao của người dùng trong kỷ nguyên AI.

Tài liệu tham khảo

Tài liệu

[1] A. Halevy, P. Norvig, and F. Pereira, "The unreasonable effectiveness of data," *IEEE Intelligent Systems*, vol. 24, no. 2, pp. 8–12, 2009.

[2] S. B. Heilesen, "What is the academic efficacy of podcasting?" *Computers & Education*, vol. 55, no. 3, pp. 1063–1068, 2010.

[3] T. Brown, B. Mann, N. Ryder, M. Subbiah *et al.*, "Language models are few-shot learners," *Advances in Neural Information Processing Systems*, vol. 33, pp. 1877–1901, 2020.

[4] Z. Ji, N. Lee, R. Frieske, T. Yu, D. Su, Y. Xu, E. Ishii, Y. J. Bang, A. Madotto, and P. Fung, "Survey of hallucination in natural language generation," *ACM Computing Surveys*, vol. 55, no. 12, pp. 1–38, 2023.

[5] V. H. Nguyen, C. M. Luong, and H. Q. Vu, "Vietnamese speech recognition: A survey," *Journal of Computer Science and Cybernetics*, vol. 36, no. 4, pp. 293–312, 2020.

[6] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[7] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, "Robust speech recognition via large-scale weak supervision," *Proceedings of the 40th International Conference on Machine Learning*, pp. 28 492–28 518, 2023.

[8] S. Team, "Silero VAD: pre-trained enterprise-grade voice activity detector," in *GitHub Repository*, 2021.

[9] H. Touvron, T. Lavril, G. Izacard *et al.*, "LLaMA: Open and efficient foundation language models," arXiv preprint arXiv:2302.13971, 2023.

[10] A. Cloud, "Qwen2.5: A large language model series," https://qwenlm.github.io/blog/qwen2.5/, 2024, truy cạp: 15/12/2024.

[11] Ollama, "Ollama - run large language models locally," https://ollama.com, truy cạp: 15/12/2024.

[12] Y. Zhang, Y. Li, L. Cui, D. Cai, L. Liu, T. Fu, X. Huang, E. Zhao, Y. Zhang, Y. Chen *et al.*, "Siren's song in the AI ocean: A survey on hallucination in large language models," *arXiv preprint arXiv:2309.01219*, 2023.

[13] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel *et al.*, "Retrieval-augmented generation for knowledge-intensive NLP tasks," *Advances in Neural Information Processing Systems*, vol. 33, pp. 9459–9474, 2020.

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

[15] R. Nogueira and K. Cho, "Passage re-ranking with BERT," *arXiv preprint arXiv:1901.04085*, 2019.

[16] G. V. Cormack, C. L. Clarke, and S. Buettcher, "Reciprocal rank fusion outperforms condorcet and individual rank learning methods," *Proceedings of the 32nd International ACM SIGIR Conference*, pp. 758–759, 2009.

[17] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using siamese BERT-networks," *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, 2019.

[18] L. Wang, N. Yang, X. Huang, B. Jiao, L. Yang, D. Jiang, R. Majumder, and F. Wei, "Text embeddings by weakly-supervised contrastive pre-training," *arXiv preprint arXiv:2212.03533*, 2022.

[19] J. Johnson, M. Douze, and H. Jégou, "Billion-scale similarity search with GPUs," *IEEE Transactions on Big Data*, vol. 7, no. 3, pp. 535–547, 2019.

[20] Qdrant, "Qdrant - vector database for the next generation of AI applications," https://qdrant.tech, truy cạp: 15/12/2024.

[21] PaddlePaddle, "PaddleOCR: Awesome multilingual OCR toolkits," https://github.com/PaddlePaddle/PaddleOCR, truy cạp: 15/12/2024.

[22] A. Software, "PyMuPDF: Python bindings for MuPDF," https://pymupdf.readthedocs.io/, truy cạp: 15/12/2024.

[23] LangChain, "Text splitters documentation," https://python.langchain.com/docs/modules/data_connection/document_transformers/, 2024, truy cạp: 15/12/2024.

[24] G. Izacard, P. Lewis, M. Lomeli, L. Hosseini, F. Petroni, T. Schick, J. Dwivedi-Yu, A. Joulin, S. Riedel, and E. Grave, "Few-shot learning with retrieval augmented language models," *arXiv preprint arXiv:2208.03299*, 2022.

[25] G. Mialon, R. Dessì, M. Lomeli, C. Nalmpantis, R. Pasunuru, R. Raileanu, B. Rozière, T. Schick, J. Dwivedi-Yu, A. Celikyilmaz *et al.*, "Augmented language models: a survey," *arXiv preprint arXiv:2302.07842*, 2023.

[26] OpenAI, "ChatGPT," https://chat.openai.com/, truy cạp: 15/12/2024.

[27] P. AI, "Perplexity - ask anything," https://www.perplexity.ai/, truy cạp: 15/12/2024.

[28] Google, "Notebooklm," https://notebooklm.google/, truy cạp: 15/12/2024.

[29] SYSTRAN, "Faster-whisper: Reimplementation of OpenAI's Whisper model using CTranslate2," https://github.com/SYSTRAN/faster-whisper, truy cạp: 15/12/2024.

[30] G. DeepMind, "Gemini: A family of highly capable multimodal models," https://deepmind.google/technologies/gemini/, 2024.

[31] S. Robertson and H. Zaragoza, "The probabilistic relevance framework: BM25 and beyond," *Foundations and Trends in Information Retrieval*, vol. 3, no. 4, pp. 333–389, 2009.

[32] S. Inc, "Streamlit - the fastest way to build data apps," https://streamlit.io, truy cạp: 15/12/2024.

[33] Microsoft, "Microsoft edge text-to-speech," https://github.com/rany2/edge-tts, truy cạp: 15/12/2024.

[34] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[35] E. M. Voorhees, "The TREC-8 question answering track report," *TREC*, vol. 99, pp. 77–82, 1999.

[36] K. Järvelin and J. Kekäläinen, "Cumulated gain-based evaluation of IR techniques," *ACM Transactions on Information Systems*, vol. 20, no. 4, pp. 422–446, 2002.