HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG



Nguyễn Quang Huy

NGHIÊN CỨU XÂY DỰNG CHATBOT TRỢ LÝ ĐỌC HIỂU TÀI LIỆU

ĐỀ ÁN TỐT NGHIỆP THẠC SỸ KỸ THUẬT

(Theo định hướng ứng dụng)

HÀ NỘI – NĂM 2026

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG



Nguyễn Quang Huy

NGHIÊN CỨU XÂY DỰNG CHATBOT TRỢ LÝ ĐỌC HIỂU TÀI LIỆU

Chuyên ngành: Hệ thống thông tin

Mã số: 8.48.01.04

ĐỀ ÁN TỐT NGHIỆP THẠC SỸ KỸ THUẬT

(Theo định hướng ứng dụng)

NGƯỜI HƯỚNG DẪN KHOA HỌC:

TS. Dương Trần Đức

HÀ NỘI – NĂM 2026

LỜI CAM ĐOAN

Tôi cam đoan rằng đề án tốt nghiệp thạc sĩ: ***"NGHIÊN CỨU XÂY DỰNG CHATBOT TRỢ LÝ ĐỌC HIỂU TÀI LIỆU"*** là công trình nghiên cứu của chính tôi.

Tôi cam đoan các số liệu, kết quả nêu trong đề án là trung thực và chưa từng được ai công bố trong bất kỳ công trình nào khác.

Không có sản phẩm/nghiên cứu nào của người khác được sử dụng trong đề án nàymà không được trích dẫn theo đúng quy định.

Hà Nội, ngày 20 tháng 5 năm 2025

Học viên thực hiện đề án

Nguyễn Quang Huy

LỜI CẢM ƠN

Trong suốt quá trình học tập và nghiên cứu thực hiện đề án tốt nghiệp thạc sĩ, ngoài nỗ lực của bản thân, tôi đã nhận được sự hướng dẫn nhiệt tình quý báu của quý Thầy Cô, cùng với sự động viên và ủng hộ của gia đình, bạn bè và đồng nghiệp. Với lòng kính trọng và biết ơn sâu sắc, tôi xin gửi lời cảm ơn chân thành tới:

Ban Giám Đốc, Phòng đào tạo sau đại học và quý Thầy Cô đã tạo mọi điều kiện thuận lợi giúp tôi hoàn thành đề án.

Tôi xin chân thành cảm ơn Thầy TS. Dương Trần Đức, người thầy kính yêu đã hết lòng giúp đỡ, hướng dẫn, động viên, tạo điều kiện cho tôi trong suốt quá trình thực hiện và hoàn thành đề án tốt nghiệp thạc sĩ.

Tôi xin chân thành cảm ơn gia đình, bạn bè, đồng nghiệp trong cơ quan đã động viên, hỗ trợ tôi trong lúc khó khăn để tôi có thể học tập và hoàn thành đề án. Mặc dù đã có nhiều cố gắng, nỗ lực, nhưng do thời gian và kinh nghiệm nghiên cứu khoa học còn hạn chế nên không thể tránh khỏi những thiếu sót. Tôi rất mong nhận được sự góp ý của quý Thầy Cô cùng bạn bè đồng nghiệp để kiến thức của tôi ngày một hoàn thiện hơn.

Xin chân thành cảm ơn!

Hà Nội, ngày 20 tháng 5 năm 2025

Học viên thực hiện đề án

Nguyễn Quang Huy

MỤC LỤC

LỜI CAM ĐOAN .....
LỜI CẢM ƠN .....
MỤC LỤC .....
DANH MỤC CÁC THUẬT NGỮ, CHỮ VIẾT TẮT .....
DANH SÁCH HÌNH VẼ .....
I. MỞ ĐẦU 1
1. Lý do chọn đề tài 1
2. Tổng quan về vấn đề nghiên cứu trong và ngoài nước: 1
3. Mục đích nghiên cứu 2
4. Đối tượng và phạm vi nghiên cứu 3
5. Phương pháp nghiên cứu 4
II. NỘI DUNG 6
CHƯƠNG 1: TỔNG QUAN VỀ VẤN ĐỀ 6
1.1 Tổng quan về xử lý ngôn ngữ tự nhiên 6
1.2 Tổng quan về mô hình ngôn ngữ lớn 7
1.3 Tổng quan về Retrieval-Augmented Generation 8
1.4 Các công trình và hệ thống liên quan 11
1.5 Ý nghĩa của đề tài 13
1.6 Định hướng giải pháp 13
CHƯƠNG 2: CƠ SỞ LÝ THUYẾT VÀ THIẾT KẾ KIẾN TRÚC 14
2.1 Cơ sở lý thuyết cho hệ thống 15
2.2 Kiến trúc tổng thể và pipeline bốn giai đoạn 18
2.3 Thiết kế API, giao diện người dùng và tổ chức dữ liệu 22
2.4 Phân tích các quyết định thiết kế và đánh đổi kỹ thuật 26
2.5 Khả năng tích hợp và mở rộng kiến trúc trong tương lai 27
CHƯƠNG 3: THỰC NGHIỆM VÀ ĐÁNH GIÁ 29
3.1 Môi trường và cấu hình thực nghiệm 29
3.2 Bộ dữ liệu kiểm thử 30
3.3 Quy trình thực nghiệm và phương pháp đánh giá 33

3.4 Kết quả đánh giá và thảo luận.....35
3.5 Khả năng ứng dụng thực tế và định hướng mở rộng thực nghiệm.....40
III. KẾT LUẬN.....42
1. Kết quả và đóng góp của đề án.....42
2. Đánh giá đóng góp ứng dụng của đề án.....43
3. Nhận xét đề án.....44
4. Hướng phát triển tiếp theo.....45
IV. DANH MỤC CÁC TÀI LIỆU THAM KHẢO.....46

DANH MỤC CÁC THUẬT NGỮ, CHỮ VIẾT TẮT

VIẾT TẮT	NGHĨA TIẾNG ANH	NGHĨA TIẾNG VIỆT
AI	Artificial Intelligence	Trí tuệ nhân tạo
API	Application Programming Interface	Giao diện lập trình ứng dụng
CoT	Chain-of-Thought	Suy luận từng bước
CORS	Cross-Origin Resource Sharing	Chia sẻ tài nguyên đa nguồn gốc
FAISS	Facebook AI Similarity Search	Thư viện tìm kiếm vector của Facebook AI
HTML	HyperText Markup Language	Ngôn ngữ đánh dấu siêu văn bản
HTTP	HyperText Transfer Protocol	Giao thức truyền tải siêu văn bản
ICL	In-Context Learning	Học trong ngữ cảnh
LLM	Large Language Model	Mô hình Ngôn ngữ Lớn
NLP	Natural Language Processing	Xử lý Ngôn ngữ Tự nhiên
PDF	Portable Document Format	Định dạng tài liệu di động
RAG	Retrieval Augmented Generation	Tạo sinh dựa trên truy xuất tăng cường
REST	Representational State Transfer	Kiến trúc API phổ biến
BM25	Best Matching 25	Thuật toán tìm kiếm theo từ khóa
OCR	Optical Character Recognition	Nhận dạng ký tự quang học
MMR	Maximal Marginal Relevance	Độ phù hợp cận biên cực đại
L2	L2 distance	Khoảng cách Euclid
all-MiniLM-L6-v2	Trained on all datasets - Mini Language Model - 6 Layers - Version 2	mô hình ngôn ngữ thu nhỏ, 6 lớp, phiên bản 2, được huấn luyện trên khối lượng dữ liệu lớn
IndexFlatL2	IndexFlatL2	Chỉ mục tìm kiếm duyệt dựa trên

		khoảng cách L2
POS tagging	Part-Of-Speech tagging	Dán nhãn, phân loại
NER	Named Entity Recognition	Nhận dạng thực thể định dan
BERT	Bidirectional Encoder Representations from Transformers	Biểu diễn Bộ mã hóa Hai chiều từ các Transformer
BF	Brute-Force	Thuật toán vét cạn
RRF	Reciprocal Rank Fusion	Dung hợp hạng nghịch đảo

DANH SÁCH HÌNH VẼ

Hình 1.1 Kiến trúc tổng quát của Retrieval-Augmented Generation..... 18

Hình 2.1 Cơ chế chunking theo ranh giới câu với vùng chồng lấp (overlap) 26

Hình 2.2 Luồng truy xuất ngữ cảnh bốn tầng 27

Hình 2.3 Pipeline bốn giai đoạn xử lý yêu cầu hỏi đáp 31

Hình 2.4 Luồng nạp tài liệu và bước khử trùng lặp chunk 33

Hình 2.5 Kiến trúc thành phần của hệ thống 35

Hình 3.1 Quy trình thực nghiệm và đánh giá.....47

I. MỞ ĐẦU

1. Lý do chọn đề tài

Trong bối cảnh hiện nay nhu cầu của người dùng cá nhân trong việc khai thác, xử lý và đọc hiểu các tài liệu văn bản như PDF, Word, sách, hợp đồng, báo cáo,... ngày càng tăng. Dù vậy người dùng cá nhân lại gặp khó khăn trong việc tóm tắt, dịch thuật, tìm kiếm hoặc hiểu rõ nội dung tài liệu một cách nhanh chóng và chính xác. Sự phát triển của các mô hình ngôn ngữ lớn như GPT-4, LLaMA 3, Claude 3, ... và các kiến trúc như RAG đã mở ra khả năng xây dựng các chatbot trợ lý tài liệu có thể hỗ trợ người dùng hiệu quả trong việc đọc hiểu và tương tác với tài liệu. Các hệ thống này đã được ứng dụng trong nhiều lĩnh vực: Trong y tế bác sĩ có thể tham khảo nhanh các phương án điều trị; trong luật và tài chính các công cụ tự động phân tích rủi ro trong hợp đồng; trong giáo dục sinh viên sử dụng chatbot để học tập và nghiên cứu thuận tiện hơn.

2. Tổng quan về vấn đề nghiên cứu trong và ngoài nước:

Hiện nay các tập đoàn công nghệ hàng đầu như Google, Meta, OpenAI, Microsoft không chỉ tập trung phát triển các mô hình ngôn ngữ lớn, mà còn đầu tư mạnh vào các framework hỗ trợ xây dựng ứng dụng RAG như LangChain hay LlamaIndex. Các nghiên cứu học thuật liên tục ra đời nhằm tối ưu từng thành phần của RAG, tiêu biểu như Semantic Chunking, Propositional Indexing, Hybrid Search, Re-ranking và các bộ tiêu chuẩn đánh giá tự động. Vì vậy đề tài đã cân nhắc giữa hai công cụ phổ biến là LangChain và LlamaIndex. Nếu LlamaIndex rất mạnh về lập index và truy xuất tài liệu, thì LangChain lại linh hoạt hơn trong việc kết nối các luồng tác vụ và kỹ thuật thiết kế prompt. Vì hệ thống cần xử lý nhiều tác vụ khác nhau như hỏi đáp, tóm tắt và trích xuất thông tin thay vì chỉ tìm kiếm đơn thuần, đề tài quyết định chọn LangChain làm nền tảng phát triển.

Tại Việt Nam, một số tập đoàn công nghệ lớn như FPT, Viettel, Zalo AI, VinBigdata đang xây dựng các mô hình ngôn ngữ lớn cho tiếng Việt. Tuy nhiên các nghiên cứu chuyên sâu và việc xây dựng các ứng dụng hoàn chỉnh dựa trên kiến trúc RAG cho bài toán trợ lý tài liệu vẫn còn ở giai đoạn đầu. Hầu hết các ứng dụng mới chỉ dừng ở mức thử nghiệm và chưa có nhiều công trình công bố chi tiết và bài bản. Do đó đây là một lĩnh vực còn rất nhiều không gian để khám phá, đóng góp và tạo ra các sản phẩm có chất lượng cao.

3. Mục đích nghiên cứu

Mục đích chính của đề tài là ứng dụng thành công các kỹ thuật tiên tiến trong lĩnh vực trí tuệ nhân tạo, cụ thể là kiến trúc Retrieval Augmented Generation và các mô hình ngôn ngữ lớn để xây dựng một hệ thống chatbot trợ lý đọc hiểu tài liệu hoàn chỉnh. Hệ thống cần có khả năng tương tác thông minh, cung cấp thông tin chính xác và đáng tin cậy được trích xuất trực tiếp từ tài liệu của người dùng. Đề tài đặt ra hai cấp mục tiêu: tổng quát và cụ thể, được trình bày lần lượt ở hai phần dưới đây.

3.1. Mục tiêu tổng quát

Mục tiêu tổng quát của đề tài là xây dựng một hệ thống trợ lý tài liệu thông minh có khả năng tiếp nhận tài liệu văn bản từ người dùng, hiểu nội dung của tài liệu đó và phản hồi các yêu cầu hỏi đáp, tóm tắt, trích xuất thông tin một cách chính xác. Hệ thống được kỳ vọng vừa có nền tảng lý thuyết vững chắc từ các kỹ thuật NLP và AI hiện đại, vừa có khả năng vận hành thực tế thông qua giao diện web đơn giản. Để hiện thực mục tiêu tổng quát, đề tài cụ thể hóa thành năm mục tiêu chi tiết được trình bày dưới đây.

3.2. Mục tiêu cụ thể

Nắm vững lý thuyết: Tìm hiểu và ứng dụng kiến trúc Transformer, LLM, cơ chế RAG, các kỹ thuật embedding và cơ sở dữ liệu vector vào hệ thống.

Chọn và tích hợp công nghệ: Khảo sát các công cụ, thư viện mã nguồn mở và phương pháp cho từng giai đoạn của pipeline RAG (tách văn bản, sinh vector, truy xuất thông tin) để chọn cấu hình phù hợp.

Thiết kế kiến trúc hệ thống: Xây dựng hệ thống theo hướng module hóa để dễ bảo trì và mở rộng, có thể xử lý các tài liệu văn bản phổ biến như PDF, DOCX, TXT phục vụ nhu cầu người dùng cá nhân trong việc đọc hiểu và tra cứu thông tin. Kiến trúc của đề tài chia thành năm nhóm chức năng chính: ingestion, chunking, embedding, pipeline và conversion. FastAPI làm lớp giao tiếp backend, cho phép các module xử lý tài liệu, truy xuất tri thức và sinh phản hồi hoạt động cùng nhau qua các endpoint REST, đồng thời vẫn giữ được tính độc lập để dễ bảo trì và mở rộng.

Triển khai hệ thống: Xây dựng và đưa vào hoạt động một ứng dụng chatbot, cho phép người dùng tải tài liệu lên, đặt câu hỏi bằng ngôn ngữ tự nhiên và nhận được câu trả lời. Hệ thống đã hỗ trợ ba tác vụ chính: hỏi đáp documentQA, tóm tắt documentSummary và trích xuất documentExtraction. Đầu vào hệ thống chấp

nhận các định dạng tài liệu gồm PDF, DOCX, DOC, TXT. Người dùng có thể nạp tài liệu, đặt yêu cầu bằng ngôn ngữ tự nhiên và nhận phản hồi theo đúng loại tác vụ mong muốn.

Đánh giá hiệu suất: Xây dựng phương pháp và tiêu chí để đánh giá hiệu suất của hệ thống RAG về độ chính xác và trải nghiệm người dùng.

4. Đối tượng và phạm vi nghiên cứu

Đối tượng và phạm vi nghiên cứu xác định rõ những gì đề tài tập trung giải quyết và những gì nằm ngoài phạm vi của nghiên cứu này. Việc giới hạn phạm vi không phải là hạn chế mà là điều kiện để đi sâu vào một vấn đề cụ thể, thay vì dàn trải qua nhiều bài toán khác nhau. Phần tiếp theo trình bày lần lượt đối tượng nghiên cứu chính và phạm vi giới hạn của đề tài.

4.1. Đối tượng nghiên cứu

Mô hình Retrieval Augmented Generation (RAG): Là sự kết hợp giữa truy xuất thông tin và sinh ngôn ngữ. Trong hệ thống, thành phần retrieval dùng FAISS với IndexFlatL2 để tìm kiếm dense vector theo khoảng cách L2 trên các chunk đã vector hóa. Sau khi truy xuất ban đầu, các ứng viên được đưa qua CrossEncoder để chấm điểm lại. Pipeline hoạt động bằng cách chọn top 15 ứng viên, sau đó rerank và chọn ra top 5 đoạn liên quan nhất để đưa vào prompt cho tác vụ hỏi đáp.

Large Language Models (LLM): Mô hình hiểu và tạo ngôn ngữ tự nhiên dùng cho tóm tắt, trả lời câu hỏi. Hệ thống sử dụng Gemini với các siêu tham số mặc định: temperature=0.2, max_output_tokens=2048, top_p=0.95, cấu hình này giúp câu trả lời ổn định, nhất quán và kiểm soát được độ dài đầu ra, phù hợp cho tác vụ học thuật và tra cứu.

Hệ thống tích hợp RAG và LLM: Xây dựng một pipeline hoàn chỉnh, nhận tài liệu văn bản đầu vào và đưa ra phản hồi ngôn ngữ tự nhiên. Hệ thống sử dụng các công cụ open source và dịch vụ có sẵn như embedding, vector DB, API inference,…

4.2. Phạm vi nghiên cứu

Hệ thống xử lý tài liệu văn bản đa ngôn ngữ với các định dạng phổ biến như pdf, docx, và txt. Hệ thống không hỗ trợ xử lý các đối tượng phi văn bản như hình ảnh, bảng biểu, công thức toán học, sơ đồ kỹ thuật hoặc chữ viết tay. Hệ thống có các chức năng chính là hỏi đáp, truy xuất thông tin chính xác và có căn cứ, cùng với tóm tắt nội dung từ một hoặc nhiều tài liệu. Đề tài ưu tiên dùng các

mô hình embedding và mô hình ngôn ngữ lớn có sẵn như GPT-4, Gemini, không tự huấn luyện mô hình từ đầu mà khai thác khả năng tích hợp, thực hiện fine-tuning nếu cần và phù hợp với tài nguyên.

Giới hạn của đề tài: Các giới hạn này đến từ định hướng kỹ thuật của hệ thống: pipeline được tối ưu cho tài liệu có sẵn lớp văn bản có thể trích xuất. Hệ thống chưa tích hợp mô-đun OCR để nhận dạng chữ trong ảnh scan hoặc tài liệu chụp. Vì vậy đề tài tập trung vào bài toán text extraction và text understanding, không mở rộng sang các bài toán thị giác máy tính và đa phương tiện vốn cần hạ tầng xử lý khác, không xử lý các định dạng tài liệu đặc biệt như ảnh scan, chữ viết tay hoặc dữ liệu đa phương tiện như video, âm thanh,... Đề tài cũng không đi sâu nghiên cứu hay cải tiến thuật toán mà tập trung vào khả năng ứng dụng.

5. Phương pháp nghiên cứu

Đề tài áp dụng kết hợp hai nhóm phương pháp nghiên cứu: nghiên cứu lý thuyết và phát triển thực nghiệm. Phương pháp lý thuyết giúp xây dựng nền tảng kiến thức vững chắc về các kỹ thuật cốt lõi, phương pháp thực nghiệm cho phép kiểm chứng các thiết kế thông qua triển khai hệ thống thực tế. Hai phương pháp này bổ trợ nhau để tạo ra một sản phẩm vừa có nền tảng khoa học vừa có tính ứng dụng. Để đánh giá hiệu suất và độ chính xác, hệ thống được kiểm thử trực tiếp thông qua một bộ câu hỏi được thiết kế sẵn, thông qua việc đối chiếu câu trả lời do được tạo ra với đáp án chuẩn, có thể đo lường khả năng tìm kiếm, độ bám sát tài liệu gốc và mức độ hiệu quả của mô hình.

5.1. Nghiên cứu tài liệu:

Đọc, tổng hợp và hệ thống các kiến thức nền tảng từ các nguồn tài liệu khoa học uy tín như sách chuyên khảo, báo cáo kỹ thuật và các bài báo trên các hội nghị và tạp chí. Nội dung nghiên cứu bao gồm: Nghiên cứu kiến thức nền tảng về các mô hình ngôn ngữ lớn LLMs, tập trung vào cơ chế hoạt động của kiến trúc Transformer, khả năng hiểu và sinh ngôn ngữ tự nhiên. Tìm hiểu nguyên lý hoạt động của mô hình Retrieval Augmented Generation bao gồm pipeline các bước: phân đoạn văn bản, tạo embedding, truy xuất thông tin và sinh phản hồi kết hợp với LLM. Phân tích các chiến lược phân đoạn tài liệu chunking, đặc biệt là các phương pháp duy trì ngữ nghĩa trong các hệ thống thực tế. Khảo sát các mô hình embedding mã nguồn mở và thương mại đang được sử dụng rộng rãi, đánh giá theo các tiêu chí như hiệu suất, khả năng mở rộng. Khảo sát các hệ thống ứng dụng LLM trong hỏi đáp và tóm tắt tài liệu văn bản.

5.2. Thiết kế hệ thống:

Kiến trúc tổng thể: Xây dựng mô hình tổng quát cho hệ thống AI Document Assistant với các thành phần chính:

- Phần xử lý tài liệu: tiếp nhận file từ người dùng, trích xuất văn bản, làm sạch và phân đoạn chunking theo ngữ nghĩa. Đây là một tác vụ nặng nên đề xuất sẽ xử lý bất đồng bộ.
- Phần phân đoạn và vector hóa: Lắng nghe các message từ module xử lý tài liệu, nhận các chunk văn bản, vector hóa chúng và lưu vào vector database.
- Phần suy luận và hỏi đáp: Cung cấp khả năng sinh văn bản từ LLM, điều phối toàn bộ quy trình trả lời câu hỏi của người dùng.

5.3. Phát triển và triển khai

Hệ thống được phát triển theo kiến trúc đã thiết kế, bao gồm các bước: Tiền xử lý và phân tích tài liệu văn bản, Phân đoạn và vector hóa, Phát triển RAG Pipeline. Hệ thống được xây dựng tăng dần, bắt đầu từ luồng cơ bản (upload PDF đến hỏi đáp) rồi mở rộng sang ingest đa định dạng, quản lý tài liệu và hỗ trợ nhiều tác vụ. Môi trường chạy là Python 3.11, backend dùng FastAPI kết hợp Uvicorn, giao diện web tĩnh dùng HTML/CSS/JavaScript. Các thư viện chính gồm sentence-transformers, faiss-cpu, langchain, langchain-google-genai, PyPDF2, python-docx, mammoth, olefile, striprtf và fpdf2. Hệ thống truy cập Gemini bằng `GOOGLE_API_KEY` lưu trong tệp env, phù hợp cho việc thử nghiệm và đánh giá trong phạm vi đề tài.

5.4. Đánh giá

Thiết kế bộ dữ liệu: Quá trình này tập trung xây dựng một tập dữ liệu hỏi đáp bám sát thực tế nhằm đánh giá toàn diện năng lực của hệ thống. Bộ dữ liệu sẽ bao phủ các dạng truy vấn đa dạng như câu hỏi trích xuất dữ kiện cụ thể, câu hỏi tóm tắt ý chính, và câu hỏi suy luận đòi hỏi kết nối logic nhiều luồng thông tin. Đi kèm với mỗi câu hỏi là một câu trả lời tiêu chuẩn. Đáp án mẫu này sẽ đóng vai trò như một hệ quy chiếu đáng tin cậy để đối chiếu, chấm điểm và đánh giá độ chính xác của AI, từ đó làm cơ sở vững chắc để liên tục tối ưu hóa mô hình.

II. NỘI DUNG

CHƯƠNG 1: TỔNG QUAN VỀ VẤN ĐỀ

Chương này trình bày các khái niệm và công nghệ nền tảng được sử dụng trong đề tài, đồng thời tổng hợp các nghiên cứu và hệ thống hiện có liên quan đến việc xây dựng trợ lý tài liệu thông minh dựa trên mô hình ngôn ngữ lớn, gồm: tổng quan về xử lý ngôn ngữ tự nhiên, mô hình ngôn ngữ lớn, Retrieval-Augmented Generation và các hệ thống tiêu biểu, từ đó rút ra những hàm ý thiết kế cho hệ thống đề xuất.

1.1 Tổng quan về xử lý ngôn ngữ tự nhiên

Natural Language Processing NLP là lĩnh vực kết hợp khoa học máy tính, trí tuệ nhân tạo và ngôn ngữ học. NLP giúp máy tính tiếp nhận, phân tích và tạo ra ngôn ngữ con người cho các bài toán thực tế. Khác với dữ liệu có cấu trúc, văn bản tự nhiên mơ hồ, đa nghĩa và phụ thuộc ngữ cảnh. Vì vậy NLP không chỉ là xử lý ký tự mà còn là mô hình hóa tri thức và ngữ nghĩa. NLP là công nghệ nền tảng cho các hệ thống tìm kiếm, trợ lý ảo và chatbot.

1.1.1 Các tác vụ cốt lõi của NLP

Trong hệ thống NLP, quy trình xử lý gồm nhiều tác vụ, cơ bản nhất là tokenization, tách văn bản thô thành từ, cụm từ hoặc subword. Tiếp theo là các tác vụ như POS tagging, phân tích cú pháp, NER và chuẩn hóa thực thể để hệ thống hiểu cấu trúc câu và các thành phần ý nghĩa. Với tiếng Việt các thách thức như từ ghép, ngắt từ không tường minh và cách viết linh hoạt đòi hỏi bước tiền xử lý chất lượng cao để đảm bảo hiệu quả cho các bước sau.

1.1.2 Vai trò của NLP trong các hệ thống trợ lý tài liệu

Trong đề tài AI Document Assistant, NLP là nền tảng cho mọi bước. Ở giai đoạn tiền xử lý, hệ thống lọc ký tự, chuẩn hóa khoảng trắng để giảm nhiễu khi tạo prompt. Các tác vụ hỏi đáp, tóm tắt và trích xuất đều cần hiểu và diễn giải yêu cầu của người dùng. Giao diện cũng được thiết kế để người dùng nhập câu hỏi hoặc yêu cầu bằng ngôn ngữ tự nhiên. Như vậy NLP không chỉ là thuật toán mà còn định hình cách người dùng tương tác với hệ thống.

1.1.3 Quy trình xử lý từ văn bản thô đến suy luận

Kiến trúc hệ thống trong đề tài thể hiện rõ luồng xử lý. Khi người dùng gửi yêu cầu qua API, lớp Preprocessor làm sạch dữ liệu, gán request_id và timestamp. Với tác vụ hỏi đáp, PromptEnricher lấy truy vấn đã làm sạch, truy

xuất các đoạn văn liên quan qua tầng truy xuất lai (kết hợp FAISS theo ngữ nghĩa và BM25 theo từ khóa, hợp nhất bằng RRF rồi tinh lọc bằng CrossEncoder và MMR), sau đó ghép ngữ cảnh truy xuất với hướng dẫn Chain-of-Thought và bộ ví dụ few-shot. Prompt hoàn chỉnh được gửi tới mô hình Gemini. Cuối cùng, Postprocessor trích xuất văn bản phản hồi từ đối tượng AIMessage, kiểm tra safety_check và cắt ngắn nếu vượt quá 10.000 ký tự.

1.1.4 Các mô hình tiêu biểu trong NLP hiện đại

Về ứng dụng, mỗi loại mô hình phù hợp với một nhóm nhiệm vụ khác nhau. Các mô hình encoder như BERT hoặc all-MiniLM-L6-v2 phù hợp cho bài toán tạo embedding vì sinh ra biểu diễn ngữ nghĩa ổn định và tối ưu cho so khớp tương đồng. Các mô hình decoder hoặc encoder-decoder lớn như GPT, Gemini hay T5 phù hợp hơn cho bài toán sinh câu trả lời, tóm tắt hoặc giải thích vì có năng lực diễn đạt ngôn ngữ mạnh. Hệ thống được xây dựng trong đề tài dùng SentenceTransformer all-MiniLM-L6-v2 để tạo vector cho văn bản và dùng Gemini 2.5 Flash để sinh đầu ra cuối cùng. Sự phân tách vai trò giữa mô hình biểu diễn và mô hình sinh chính là một thực hành phổ biến trong các kiến trúc AI tài liệu hiện đại.

1.2 Tổng quan về mô hình ngôn ngữ lớn

Mô hình ngôn ngữ lớn LLM là lớp mô hình được huấn luyện trên khối lượng dữ liệu văn bản rất lớn với số lượng tham số có thể lên tới hàng tỷ hoặc hàng trăm tỷ. Điểm đột phá của LLM nằm ở khả năng học các mẫu quan hệ thống kê và ngữ nghĩa ở quy mô rộng, từ đó thực hiện nhiều nhiệm vụ khác nhau chỉ thông qua prompt thay vì cần thiết kế mô hình riêng cho từng bài toán. Nếu trước đây mỗi tác vụ NLP như phân loại, dịch máy, tóm tắt hay hỏi đáp đòi hỏi một pipeline tương đối độc lập, thì LLM đưa ra khả năng thống nhất các nhiệm vụ đó vào một nền tảng ngôn ngữ chung. Điều này làm thay đổi mạnh mẽ cách phát triển ứng dụng AI, bởi nhà phát triển có thể xây dựng dịch vụ tri thức bằng cách kết hợp prompt, dữ liệu và cơ chế truy xuất thay vì huấn luyện lại mô hình từ đầu.

1.2.1 Kiến trúc Transformer

Một khối Transformer gồm các thành phần chính như self-attention nhiều đầu, mạng feed-forward, chuẩn hóa lớp và kết nối tắt. Self-attention giúp mô hình hiểu mức độ quan trọng của từng từ trong ngữ cảnh. Multi-head attention cho phép mô hình nhìn chuỗi dưới nhiều góc độ khác nhau, còn positional

encoding bổ sung thông tin vị trí.

1.2.2 Chu trình huấn luyện của LLM

Lợi ích của cách làm này là nhà phát triển ứng dụng không nhất thiết phải tham gia vào công đoạn huấn luyện tốn kém nhất. Đề tài hiện tại thay vì tự huấn luyện một LLM, hệ thống tận dụng Gemini làm mô hình nền đã được nhà cung cấp huấn luyện và căn chỉnh. Lớp ứng dụng sau đó tập trung vào việc cung cấp ngữ cảnh tài liệu, ví dụ few-shot và quy tắc suy luận để điều hướng mô hình vào bài toán hỏi đáp tài liệu. Cách làm này phản ánh hướng tiếp cận phổ biến trong phát triển sản phẩm AI hiện nay: tận dụng sức mạnh của mô hình nền tảng, đồng thời bù đắp giới hạn miền tri thức bằng RAG và thiết kế prompt có cấu trúc.

1.2.3 Đánh giá, hạn chế và các vấn đề cần kiểm soát

LLM rất mạnh nhưng vẫn còn tồn tại nhiều hạn chế. Hiện tượng hallucination khiến mô hình sinh ra thông tin nghe có vẻ hợp lý nhưng không dựa trên nguồn xác thực. Tri thức của mô hình bị giới hạn bởi mốc dữ liệu huấn luyện, dẫn tới tình trạng knowledge cutoff. Context window dù đã tăng mạnh nhưng vẫn là tài nguyên hữu hạn; nếu đưa toàn bộ tài liệu dài vào một prompt, chi phí suy luận sẽ cao và nguy cơ mô hình bỏ sót chi tiết quan trọng vẫn tồn tại. Ngoài ra còn có các vấn đề về độ ổn định phản hồi, độ nhạy với prompt, quyền riêng tư dữ liệu và chi phí gọi API theo số token. Để khắc phục, các kiến trúc bổ trợ như RAG, reranking, guardrail an toàn và kiểm soát hậu xử lý đã ra đời.

1.2.4 Ứng dụng của LLM và lý do lựa chọn Gemini

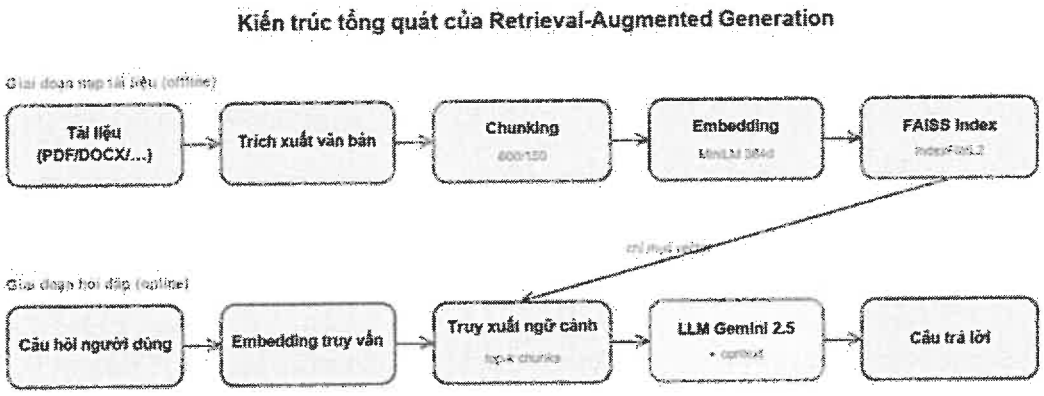
Hệ thống AI Document Assistant sử dụng mô hình gemini-2.5-flash. Việc lựa chọn biến thể Flash thể hiện định hướng cân bằng giữa chất lượng và độ trễ phản hồi, phù hợp với ứng dụng web tương tác gần thời gian thực. Gemini là dòng mô hình đa phương thức của Google, được tối ưu cho nhiều tác vụ và có khả năng xử lý ngữ cảnh dài. Với đề tài này ưu điểm đáng chú ý là mô hình có thể phản hồi tốt ở cả tiếng Anh và tiếng Việt, dễ tích hợp qua API, đồng thời cho phép cấu hình các tham số sinh như temperature, max_output_tokens và top_p thông qua biến môi trường. Điều này giúp hệ thống giữ được tính linh hoạt khi triển khai thử nghiệm các kịch bản khác nhau mà không làm phức tạp quá mức tầng ứng dụng.

1.3 Tổng quan về Retrieval-Augmented Generation

Retrieval-Augmented Generation RAG là cách tiếp cận kết hợp hai năng lực: truy xuất thông tin từ kho dữ liệu bên ngoài và sinh phản hồi bằng mô hình ngôn

ngữ lớn. Ý tưởng căn bản của RAG là thay vì yêu cầu LLM nhớ toàn bộ tri thức, hệ thống sẽ chủ động tìm các đoạn tài liệu liên quan tới truy vấn, sau đó chèn ngữ cảnh này vào prompt để mô hình dựa trên đó mà trả lời. Cấu trúc này tạo ra cầu nối giữa tính cập nhật của kho dữ liệu động với khả năng diễn đạt tự nhiên của LLM. Trong các ứng dụng tài liệu, đây là chiến lược gần như bắt buộc nếu muốn giảm ảo giác và tăng tính kiểm chứng của câu trả lời.

1.3.1 Quy trình hoạt động của RAG



Hình 1.1: Kiến trúc tổng quát của Retrieval-Augmented Generation

Một pipeline RAG điển hình gồm các bước: thu nhận tài liệu, chuyển đổi sang văn bản sạch, phân đoạn, vector hóa, lập chỉ mục, truy xuất và sinh phản hồi. Đầu tiên tài liệu thô phải được đưa về dạng văn bản xử lý được. Văn bản sau đó được chia thành các đoạn nhỏ, đủ ngắn để biểu diễn tốt nhưng vẫn giữ được ngữ cảnh. Mỗi đoạn được mã hóa thành embedding và lưu vào cơ sở dữ liệu vector. Khi có truy vấn hệ thống tính embedding của truy vấn, tìm các đoạn gần nhất trong không gian vector và ghép chúng thành ngữ cảnh cho LLM. Nếu có thêm bước reranking, các đoạn ứng viên sẽ được sắp xếp lại theo mức độ phù hợp trước khi chuyển vào prompt cuối cùng.

Hệ thống của đề tài triển khai luồng xử lý RAG như sau: Ở giai đoạn ingestion, các hàm extract_pages_from_file được gọi để trích xuất nội dung, ghép thành full_text, sau đó phân đoạn bằng chunk_text. Các chunk gần trùng được loại bỏ, khử trùng lặp dựa trên độ tương đồng Jaccard, rồi tạo embedding qua create_embeddings. Các vector này được lập chỉ mục bằng FAISS IndexFlatL2 và ghi xuống đĩa, kèm theo mapping giữa chỉ số và nội dung chunk. Khi có câu hỏi PromptEnricher gọi tầng truy xuất lai (kết hợp FAISS theo ngữ nghĩa và BM25 theo từ khóa, hợp nhất bằng RRF) để lấy các đoạn văn tiềm năng. Sau đó CrossEncoder và MMR được dùng để xếp hạng lại và chọn lọc trước khi đưa

vào prompt. Điều này cho thấy đề tài đã triển khai đầy đủ các bước của RAG trong mã nguồn, chứ không chỉ dừng ở mức khái niệm.

1.3.2 Lợi ích của RAG đối với bài toán hỏi đáp tài liệu

RAG mang lại hiệu quả vận hành đáng kể cho đề tài. Thay vì gửi toàn bộ tài liệu dài tới LLM cho mọi câu hỏi, hệ thống chỉ chuyển một tập con ngữ cảnh liên quan. Điều này giúp tiết kiệm token, giảm chi phí API và làm cho prompt tập trung hơn. Người dùng sẽ nhận được phản hồi ngắn gọn, đúng trọng tâm và ít lan man. Khi kết hợp với kỹ thuật Chain-of-Thought (yêu cầu mô hình "chỉ trả lời dựa trên context" và "nêu rõ khi tài liệu không chứa thông tin"), tầng RAG trở thành bộ lọc kiến thức, giúp hệ thống đáng tin cậ y hơn trong thực tế.

1.3.3 Những yếu tố quyết định chất lượng của RAG

Chất lượng của một hệ RAG không chỉ phụ thuộc vào mô hình ngôn ngữ mà còn phụ thuộc mạnh vào bốn nhóm yếu tố: chất lượng văn bản đầu vào, chiến lược chunking, chất lượng embedding và cơ chế xếp hạng kết quả truy xuất. Nếu văn bản trích xuất từ tài liệu bị lỗi OCR hoặc lẫn nhiều ký tự rác, embedding thu được sẽ thiếu ổn định. Nếu chunk quá ngắn thì thông tin ngữ cảnh bị chia cắt; nếu chunk quá dài thì embedding trở nên loãng và kết quả truy xuất dễ kém chính xác. Tương tự, một mô hình embedding nhỏ nhưng phù hợp có thể cho hiệu quả tốt hơn một mô hình lớn song không tương thích miền dữ liệu. Cuối cùng, reranking quan trọng vì đoạn văn gần nhất theo vector chưa chắc đã hữu ích nhất cho việc sinh câu trả lời.

Mã nguồn của đề tài đã tính đến các yếu tố này. Chunker sử dụng kích thước mặc định 600 ký tự với overlap 150 ký tự, kèm căn chỉnh theo ranh giới câu để vừa giữ ngữ cảnh vừa tránh cắt cụt ý. Sau đó, bước khử trùng lặp chunk (dựa trên độ tương đồng Jaccard giữa các n-gram) loại bỏ các đoạn gần trùng trước khi vector hóa. Vector store dùng all-MiniLM-L6-v2 với không gian 384 chiều, tạo chỉ mục FAISS dạng IndexFlatL2 ưu tiên độ chính xác cho corpora quy mô vừa. Tầng truy xuất được thiết kế theo hướng lai (hybrid): kết hợp truy xuất ngữ nghĩa bằng FAISS với truy xuất từ khóa bằng BM25, hợp nhất hai bảng xếp hạng bằng thuật toán RRF; sau đó CrossEncoder đánh giá lại từng cặp (query, chunk) và thuật toán MMR chọn ra tập đoạn vừa liên quan vừa đa dạng, tránh đưa vào prompt nhiều đoạn trùng lặp nội dung. Sự kết hợp nhiều tầng tối ưu như vậy là dấu hiệu của một thiết kế RAG có chủ đích, không dừng ở mức truy xuất rồi ghép vào prompt một cách cơ học.

1.3.4 Ứng dụng điển hình của RAG trong các hệ trợ lý tri thức

Đề tài AI Document Assistant là ứng dụng RAG hướng tài liệu với vài điểm khác biệt. Hệ thống không chỉ hỏi đáp mà còn tóm tắt và trích xuất thông tin, dùng chung một pipeline bốn giai đoạn. Lớp ingestion hỗ trợ nhiều định dạng tệp như PDF, DOCX, DOC, TXT, RTF, phù hợp với dữ liệu thực tế. Thiết kế module tách rời các thành phần ingestion, chunking, embedding và pipeline giúp dễ dàng thay thế công nghệ ở từng lớp mà không ảnh hưởng đến toàn bộ kiến trúc. Điều này cho thấy RAG trong đề tài là một nguyên lý tổ chức hệ thống, không chỉ là giải pháp thuật toán.

1.4 Các công trình và hệ thống liên quan

Thị trường công cụ AI đọc hiểu tài liệu đã phát triển nhanh chóng với nhiều giải pháp thương mại và mã nguồn mở. Việc tìm hiểu các hệ thống này giúp đề tài xác định những điểm còn thiếu để từ đó định hướng thiết kế. Phần này sẽ phân tích hai nhóm chính: sản phẩm thương mại và dự án mã nguồn mở.

1.4.1 Các hệ thống thương mại phổ biến

ChatGPT với khả năng phân tích tệp đã mở ra mô hình tương tác tài liệu dựa trên đối thoại cho người dùng phổ thông. Điểm mạnh của hệ thống này là giao diện đơn giản, chất lượng sinh ngôn ngữ tốt và khả năng thực hiện nhiều thao tác trên cùng một tài liệu như tóm tắt, hỏi đáp, viết lại hoặc giải thích biểu đồ. Tuy nhiên khi nhìn từ góc độ nghiên cứu ứng dụng, hệ thống thương mại thường là hộp đen: người dùng khó kiểm soát cách tài liệu được phân đoạn, mô hình nào dùng cho embedding, cơ chế truy xuất ra sao hoặc liệu câu trả lời có bám sát tài liệu nguồn tới mức nào, điều này khiến các giải pháp thương mại mạnh về tiện dụng nhưng hạn chế ở tính minh bạch và khả năng tùy biến theo miền nghiệp vụ riêng.

1.4.2 Các dự án nguồn mở tiêu biểu

Trong các dự án mã nguồn mở, PrivateGPT là ví dụ tiêu biểu cho hệ thống hỏi đáp tài liệu chạy cục bộ. Dự án này chú trọng quyền riêng tư dữ liệu, cho phép người dùng ingest tài liệu vào vector store và truy vấn bằng mô hình chạy tại chỗ hoặc qua API cấu hình. AnythingLLM lại là một nền tảng workspace, hỗ trợ nhiều backend LLM và vector database, cùng với giao diện quản lý bộ sưu tập tài liệu khá linh hoạt. Quivr thì tập trung vào trải nghiệm "bộ não thứ hai", nơi người dùng tải tài liệu và khai thác thông tin qua giao diện hội thoại. Nhìn chung, các dự án này cho thấy cộng đồng nguồn mở đã chuẩn hóa nhiều cách

thiết kế cho bài toán trợ lý tài liệu, nhất là mô hình ingest – embed – retrieve – answer.

1.4.3 So sánh và định vị đề tài nghiên cứu

So với các công cụ thương mại, hệ thống của đề tài có ưu điểm là kiến trúc minh bạch. Người phát triển có thể xem trực tiếp cách tài liệu được đọc, cách chunks được tạo và khử trùng lặp, cách embedding được sinh ra và cách các kết quả truy xuất lai (FAISS + BM25, hợp nhất bằng RRF) được rerank bằng CrossEncoder, sau đó chọn lọc bằng MMR trước khi gửi cho LLM. So với các dự án nguồn mở tổng quát, hệ thống có ưu điểm là gọn nhẹ và giải thích rõ luồng xử lý qua bốn giai đoạn: tiền xử lý, làm giàu prompt, gọi mô hình và hậu xử lý. Điều này phù hợp với mục tiêu của một đề án thạc sĩ định hướng ứng dụng, không phải là tích hợp mọi tính năng, mà là xây dựng một sản phẩm có kiến trúc rõ ràng, dễ vận hành và có thể phân tích từ góc độ khoa học.

Đề tài còn có một điểm khác biệt là hỗ trợ nhiều tác vụ trên cùng một hạ tầng tài liệu. Thay vì chỉ hỏi đáp tài liệu như các hệ RAG cơ bản, hệ thống còn hỗ trợ tóm tắt và trích xuất thông tin với các prompt few-shot riêng. Ngoài ra giao diện người dùng được xây dựng bằng HTML/CSS/JavaScript thuần, giúp giảm sự phụ thuộc vào các framework nặng và minh họa rõ cách frontend làm việc với backend FastAPI. Trong bối cảnh nghiên cứu ứng dụng ở Việt Nam, việc hỗ trợ tiếng Việt và xử lý nhiều định dạng tệp là một lợi thế đáng kể. Vì thế đề tài này có thể coi là một công trình tích hợp thực tế: không nhằm cạnh tranh trực tiếp với các nền tảng thương mại lớn, mà cung cấp một mô hình kiến trúc rõ ràng để xây dựng trợ lý tài liệu nội bộ.

Qua việc khảo sát các công trình và hệ thống liên quan có thể rút ra rằng xu hướng chủ đạo của trợ lý tài liệu thông minh hiện nay là kết hợp LLM với truy xuất ngữ nghĩa và thiết kế prompt có cấu trúc. Điều còn khác biệt giữa các hệ thống không nằm ở ý tưởng tổng quát, mà ở cách tổ chức pipeline, cách kiểm soát chất lượng truy xuất, mức độ minh bạch triển khai và mức độ phù hợp với nhu cầu người dùng cụ thể.

Khảo sát gợi ý một số điểm thiết kế quan trọng. Một là trợ lý tài liệu thông minh cần dựa vào tài liệu nguồn, không chỉ riêng LLM để tránh hallucination và knowledge cutoff. Hai là mục tiêu chính là mô hình trả lời đúng theo tài liệu, yêu cầu tác vụ và định dạng mong muốn chứ không phải chỉ nói hay. Do đó đề tài chọn kiến trúc kết hợp nhiều lớp kỹ thuật thay vì chỉ phụ thuộc vào mô hình nền tảng.

1.5 Ý nghĩa của đề tài

Từ hạn chế đã xác định, đề tài đặt ra các tiêu chí định hướng: Một là phản hồi phải đúng theo tài liệu, ưu tiên bám sát nguồn. Hai là hệ thống phải triển khai được trên máy trạm phổ thông, dùng thư viện mã nguồn mở thông dụng. Ba là thiết kế module để các thành phần dễ thay thế hoặc mở rộng mà không ảnh hưởng cấu trúc chung. Bốn là dễ sử dụng, người dùng có thể khai thác hệ thống qua giao diện đơn giản mà không cần hiểu về vector database hay prompt engineering.

Dù đã có nhiều thành tựu các hệ trợ lý tài liệu vẫn còn hạn chế ở sự cân bằng giữa tính chính xác theo tài liệu, tính minh bạch triển khai và khả năng thích nghi với dữ liệu bản địa. Nhiều nền tảng thương mại tốt nhưng khó xem cơ chế truy xuất, tùy biến kém và không phù hợp yêu cầu quản lý dữ liệu nội bộ. Ngược lại nhiều dự án nguồn mở tự chủ cao nhưng lại phức tạp khi cấu hình hoặc thiếu giao diện hoàn chỉnh cho người dùng không chuyên. Điều này cho thấy cần phát triển các hệ thống gọn, dễ triển khai và có cấu trúc kỹ thuật rõ ràng.

1.6 Định hướng giải pháp

Sau khi khảo sát các hệ thống hiện có, có thể nhận thấy một số khoảng trống chung mà hầu hết các giải pháp hiện tại chưa giải quyết triệt để. Đây chính là cơ sở để đề tài xác định tiêu chí thiết kế và định hướng giải pháp của mình. Phần này tổng hợp những hàm ý rút ra từ phần tổng quan và nêu rõ các tiêu chí mà hệ thống AI Document Assistant hướng tới.

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT VÀ THIẾT KẾ KIẾN TRÚC

Phần này tập trung mô tả cụ thể cơ sở lý thuyết và kiến trúc tổng thể của hệ thống trợ lý đọc hiểu tài liệu sử dụng mô hình ngôn ngữ lớn, đồng thời phân tích vai trò và cơ chế hoạt động của từng bộ phận của hệ thống. Nội dung bao gồm:

- Cơ sở lý thuyết: Trong phần cơ sở lý thuyết sẽ phân tích những khái niệm, kỹ thuật và công nghệ nền tảng cần thiết để xây dựng hệ thống trợ lý đọc hiểu tài liệu:

+ Mô hình Embedding: Giới thiệu khái quát về vai trò của embedding trong việc chuyển đổi nội dung văn bản sang dạng vector số, giúp phản ánh đặc trưng ngữ nghĩa và so sánh mức độ tương đồng giữa các đoạn. Trình bày kỹ thuật Chunking để chia văn bản dài thành các đoạn nhỏ; các phương pháp tạo embedding phổ biến như BERT, Sentence-BERT, OpenAI Embedding.

+ Vector Database: Trình bày vai trò của cơ sở dữ liệu vector trong việc lưu trữ và truy xuất embedding.

+ Truy xuất thông tin: Giới thiệu khái quát về mục tiêu và vai trò của IR trong việc tìm các đoạn văn bản liên quan đến câu hỏi. Trình bày các phương pháp tiêu biểu như Lexical Search, Vector Search, Hybrid Search, Re-ranking.

+ Kỹ thuật Prompt Engineering: Trình bày vai trò của kỹ thuật thiết kế prompt trong việc định hướng phản hồi của mô hình, giúp LLM trả lời chính xác dựa trên ngữ cảnh tài liệu. Giới thiệu các cách tiếp cận phổ biến như Zero-shot, Few-shot prompting.

- Kiến trúc tổng thể của hệ thống: Kiến trúc hệ thống được xây dựng theo hướng phân tách thành các phần độc lập, mỗi thành phần đảm nhiệm một vai trò riêng biệt. Cách tiếp cận này không chỉ giúp dễ dàng mở rộng, mà còn tối ưu hóa hiệu suất xử lý trong những tình huống phức tạp, đặc biệt là khi phải làm việc với khối lượng dữ liệu văn bản lớn.

+ Tiền xử lý: Tiếp nhận tệp đầu vào (PDF, DOCX, DOC, TXT). Trích xuất văn bản và làm sạch dữ liệu.

+ Phân đoạn và vector hóa: Chuyển các đoạn văn bản thành vector bằng mô hình Embedding. Lưu trữ vector trong vector database để truy xuất nhanh.

+ Truy xuất thông tin: Nhận câu hỏi của người dùng, sinh vector cho câu hỏi. Tìm kiếm các đoạn văn bản liên quan trong cơ sở dữ liệu, có thể sử dụng thuật toán Hybrid Search. Sử dụng Re-ranking để sắp xếp kết quả theo mức độ phù hợp.

+ Phản hồi: Vector hóa câu hỏi của người dung để truy xuất tìm ra các đoạn nội dung văn bản liên quan, biến đổi thành 1 promt (có thể áp dụng kỹ thuật Prompt Engineering) gửi lên mô hình ngôn ngữ lớn LLM và gửi phản hồi.

Luồng dữ liệu và tương tác giữa các thành phần: Hệ thống hoạt động theo các bước sau: Người dùng tải tài liệu lên, hệ thống tiền xử lý và phân đoạn văn bản. Các đoạn văn bản này được embedding và lưu trữ trong vector database. Khi người dùng đặt câu hỏi, hệ thống tìm các đoạn liên quan. LLM sẽ tạo câu trả lời dựa trên dữ liệu đã tìm được. Cuối cùng, câu trả lời hiển thị tại giao diện người dùng.

2.1 Cơ sở lý thuyết cho hệ thống

Chương này trình bày các kỹ thuật chính dùng để xây dựng hệ thống AI Document Assistant. Hệ thống được phát triển dựa trên sự kết hợp của nhiều kỹ thuật khác nhau, bao gồm: đọc và làm sạch tài liệu, phân đoạn văn bản, biểu diễn ngữ nghĩa, truy xuất bằng vector, xếp hạng lại, thiết kế prompt, gọi mô hình ngôn ngữ lớn và hậu xử lý phản hồi.

Thiết kế hệ thống này cho thấy chất lượng đầu ra phụ thuộc vào toàn bộ pipeline, không riêng gì LLM. Nếu tài liệu bị đọc sai, chia chunk kém, hay truy xuất nhầm, mô hình ngôn ngữ mạnh cũng khó trả lời đúng. Vì vậ phần lý thuyết của đề tài được xây dựng theo từng lớp thành phần, từ biểu diễn dữ liệu đến điều phối suy luận. Cách này giúp hệ thống dễ bảo trì và mở rộng, do mỗi module có trách nhiệm rõ ràng.

2.1.1 Mô hình embedding all-MiniLM-L6-v2

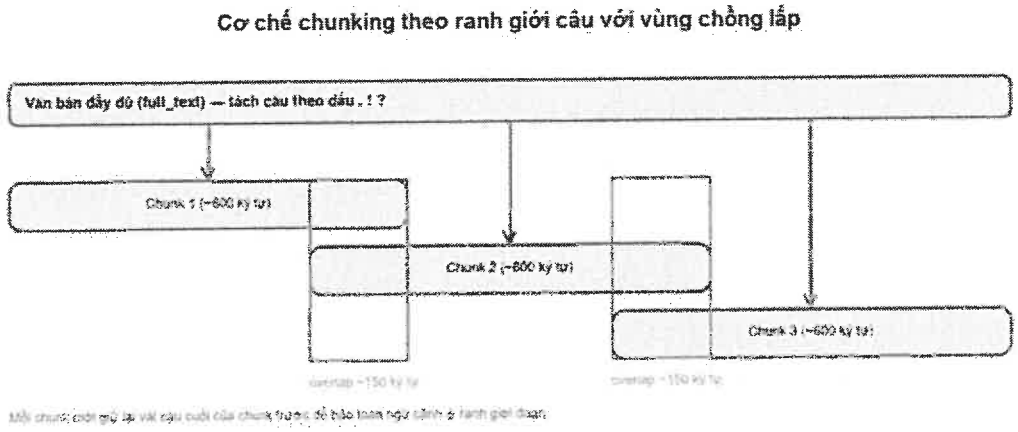
Văn bản trong hệ thống được chuyển đổi thành các vector thông qua kỹ thuật embedding, giúp máy tính hiểu và tìm kiếm tài liệu theo ý nghĩa thay vì chỉ so khớp từng từ khóa. Đề án sử dụng mô hình all-MiniLM-L6-v2 vì đặc tính gọn nhẹ, tốc độ xử lý nhanh và độ chính xác cao. Để lưu trữ và tìm kiếm các vector này, hệ thống sử dụng thư viện FAISS với cấu trúc IndexFlatL2. Đây là phương pháp tìm kiếm duyệt cạn, đảm bảo kết quả truy xuất ngữ cảnh luôn chính xác tuyệt, hoàn toàn phù hợp với quy mô tài liệu thực tế của đề án.

2.1.2 Cơ sở dữ liệu vector FAISS và chỉ mục IndexFlatL2

FAISS đóng vai trò như bộ nhớ của hệ thống, chuyên dùng để lưu trữ và tìm kiếm các đoạn tài liệu dưới dạng vector. Đề án sử dụng cấu trúc IndexFlatL2 để thực hiện tìm kiếm BF thay vì dùng các thuật toán xấp xỉ phức tạp. Lý do là vì với quy mô tài liệu cỡ vừa của đề án, phương pháp này đảm bảo tốc độ phản hồi cực nhanh trên CPU mà không cần tốn công huấn luyện mô hình. Nó mang lại độ chính xác truy xuất ngữ cảnh tuyệt đối, giúp hệ thống hoạt động ổn định, dễ triển khai và bảo trì.

2.1.3 Thuật toán chunking theo ranh giới câu

Chunking là kỹ thuật cắt một tài liệu dài thành nhiều đoạn văn nhỏ để hệ thống dễ dàng nhúng và tìm kiếm thông tin. Việc chọn kích thước đoạn văn đòi hỏi sự cân bằng: nếu cắt quá dài sẽ làm loãng thông tin, nhưng nếu cắt quá ngắn sẽ làm mất đi ngữ cảnh của câu. Để giải quyết vấn đề này hệ thống áp dụng thuật toán chia đoạn dựa trên dấu kết thúc câu kết hợp với cơ chế vùng chồng lấp (overlap). Thay vì cắt khúc văn bản một cách cứng nhắc, khi một đoạn văn đạt đến giới hạn số lượng chữ cho phép, hệ thống sẽ chốt đoạn đó lại, sau đó sẽ tự động lấy lại 1-2 câu cuối cùng của đoạn văn trước để làm phần mở đầu cho đoạn kế tiếp. Cơ chế chồng lấp này đảm bảo luồng ngữ cảnh luôn được duy trì liên tục, giúp LLM không bị mất mát các thông tin quan trọng nằm ở ranh giới giữa hai đoạn văn.



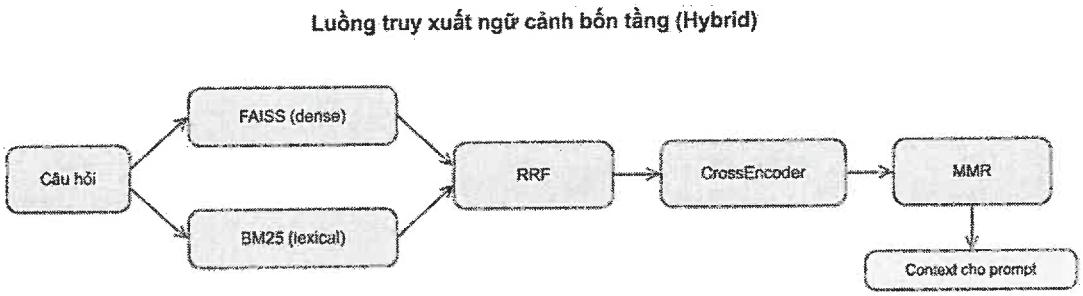
Hình 2.1: Cơ chế chunking theo ranh giới câu với vùng chồng lấp (overlap)

Với tài liệu tiếng Việt và tiếng Anh thông thường, 600 ký tự thường tương đương vài câu đến một đoạn văn ngắn, đủ để giữ mạch ý chính nhưng không quá dài khiến embedding bị loãng. Overlap 150 ký tự giúp tăng khả năng truy xuất đúng khi truy vấn liên quan tới phần chuyển tiếp giữa hai đoạn, chẳng hạn một định nghĩa nằm ở cuối đoạn này nhưng ví dụ minh họa lại ở đầu đoạn kế tiếp. Việc căn chỉnh theo câu thay vì cắt cứng theo số ký tự làm cho chunk tự nhiên

hơn, từ đó cả mô hình embedding lẫn mô hình sinh đều dễ xử lý hơn.

Chunking còn tác động tới chi phí lưu trữ và thời gian truy xuất. Nếu chunk quá lớn, số lượng vector giảm nhưng mỗi vector kém đặc hiệu; nếu chunk quá nhỏ, số lượng vector tăng làm chỉ mục phình to và truy vấn trở nên đắt hơn. Giải pháp trong đề tài có thể xem là cấu hình cân bằng cho kho tài liệu vừa và không yêu cầu phần cứng quá mạnh.

2.1.4 CrossEncoder reranker trong tầng truy xuất



Hình 2.2: Luồng truy xuất ngữ cảnh bốn tầng (Hybrid BM25 + FAISS → RRF → CrossEncoder → MMR)

Để đảm bảo chatbot nhận được thông tin đầu vào chính xác nhất, hệ thống xây dựng cơ chế tìm kiếm nhiều lớp chặt chẽ: Đầu tiên tài liệu được tìm kiếm bằng kỹ thuật Hybrid Search: kết hợp tìm kiếm FAISS và tìm kiếm BM25. Phương pháp này giúp hệ thống không bị bỏ sót bất kỳ từ khóa hiếm hoặc tên riêng đặc thù nào.

Sau khi có danh sách kết quả thô ban đầu, hệ thống tiếp tục đưa qua bước Xếp hạng lại Reranking bằng mô hình CrossEncoder. Nếu tìm kiếm FAISS chỉ phát hiện được các đoạn văn cùng chủ đề thì CrossEncoder đóng vai trò như một bộ lọc tinh, chấm điểm lại độ sát nghĩa của từng đoạn văn để lọc ra đúng đoạn chứa câu trả lời trực tiếp.

Cuối cùng hệ thống dùng thuật toán MMR để loại bỏ các đoạn văn có nội dung bị trùng lặp. Nhờ chuỗi xử lý này 5 đoạn ngữ cảnh cuối cùng được cấp cho LLM luôn đạt chất lượng cao nhất, vừa sát với câu hỏi, vừa đa dạng thông tin, giúp câu trả lời sinh ra luôn sắc bén và không bị lan man.

2.1.5 In-Context Learning, few-shot và Chain-of-Thought

Để đảm bảo chatbot trả lời đúng trọng tâm và tuyệt đối không bịa ra thông tin, hệ thống không để cho mô hình ngôn ngữ suy nghĩ tự do. Thay vào đó hành vi của AI được định hướng chặt chẽ qua hai kỹ thuật:

Few-shot: Hệ thống cung cấp sẵn 24 ví dụ gồm câu hỏi và câu trả lời chuẩn mực cho các tác vụ hỏi đáp, tóm tắt và trích xuất.

Chain-of-Thought: Bắt buộc mô hình phải lập luận theo một quy trình các bước nghiêm ngặt. Nhờ đó luồng suy luận của chatbot trở nên logic, minh bạch và có thể truy vết được.

Bộ câu lệnh Prompt còn được thiết kế để tự động nhận diện ngôn ngữ. Người dùng có thể đặt câu hỏi bằng tiếng Việt hay tiếng Anh tùy ý, hệ thống sẽ tự động hiểu và phản hồi bằng chính ngôn ngữ đó.

2.1.6 LangChain

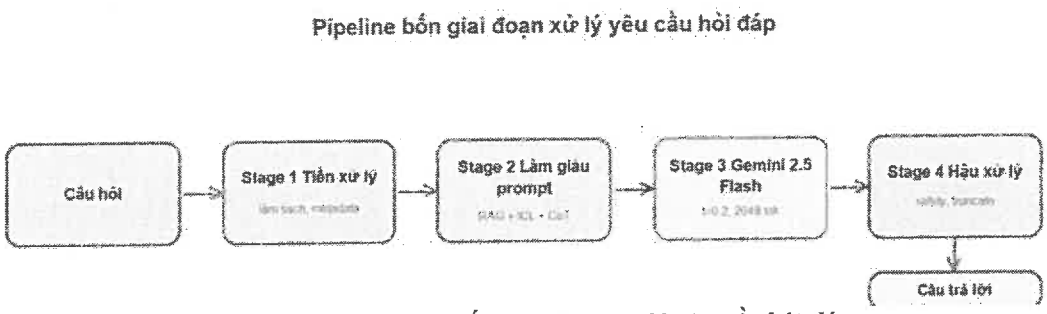
LangChain giúp hệ thống kết nối ổn định với mô hình ngôn ngữ Gemini 2.5 Flash, đồng thời cung cấp các công cụ định dạng để đóng gói câu hỏi một cách gọn gàng, có cấu trúc. Quan điểm thiết kế này giúp hệ thống đạt được hai mục tiêu: tiết kiệm thời gian phát triển nhờ tận dụng các bộ thư viện chuẩn hóa có sẵn, nhưng vẫn giữ nguyên được sự minh bạch của mã nguồn. Lập trình viên hoàn toàn làm chủ được luồng dữ liệu chứ không bị lệ thuộc vào một nền tảng duy nhất, giúp việc bảo trì và mở rộng hệ thống về sau trở nên thuận tiện.

2.2 Kiến trúc tổng thể và pipeline bốn giai đoạn

Kiến trúc của AI Document Assistant được tổ chức theo hướng module hóa, phản ánh rất rõ trong cấu trúc thư mục src gồm ingestion, chunking, embedding, pipeline và conversion. Mỗi thư mục đại diện cho một trách nhiệm kỹ thuật riêng. ingestion phụ trách đọc tài liệu và lưu trữ; chunking phụ trách chia đoạn văn bản; embedding chịu trách nhiệm vector hóa và truy xuất; pipeline điều phối luồng suy luận AI. Cách chia module này giúp nhà phát triển có thể thay thế từng lớp công nghệ mà không cần viết lại toàn bộ hệ thống.

Trung tâm của hệ thống là lớp InContextLearningModul, lớp này điều phối bốn stage: Preprocess, Enrich, LLM Call và Postprocess. Việc đặt tên tường minh cho từng stage có ý nghĩa rất quan trọng đối với khả năng quan sát và phân tích hệ thống. Khi có lỗi hoặc cần tối ưu, người phát triển có thể xác định vấn đề xuất hiện ở stage nào thay vì debug toàn bộ pipeline như một khối mờ. Đây là tư duy kiến trúc đáng chú ý trong một sản phẩm AI ứng dụng.

2.2.1 Stage 1 - Tiền xử lý yêu cầu và văn bản



Hình 2.3: Pipeline bốn giai đoạn xử lý yêu cầu hỏi đáp

Stage đầu tiên được hiện thực bởi lớp Preprocessor trong source code. Thành phần này thực hiện ba nhiệm vụ chính: làm sạch văn bản, chuẩn hóa truy vấn và gắn metadata cho mỗi lần gọi. Các biểu thức chính quy _CTRL_CHARS, _EXTRA_SPACES và _EXTRA_NEWLINES lần lượt loại bỏ ký tự điều khiển không in được, gom nhiều khoảng trắng hoặc tab thành một khoảng trắng và chuẩn hóa từ ba dòng trống trở lên về hai dòng trống. Cách làm này tuy đơn giản nhưng rất cần thiết, bởi prompt gửi tới LLM càng ít nhiễu thì xác suất mô hình hiểu đúng càng cao. Với dữ liệu tài liệu thật, đặc biệt khi xuất phát từ PDF hoặc DOC cũ, lớp tiền xử lý thường quyết định trực tiếp đến chất lượng các bước phía sau.

Preprocessor còn sinh request_id bằng UUID và timestamp theo chuẩn ISO với múi giờ UTC. Metadata này được trả về cùng conversation đã làm sạch và được ghi log ở nhiều endpoint. Đây là lựa chọn thiết kế đúng đắn vì hệ thống AI ứng dụng không thể chỉ quan tâm tới đầu vào và đầu ra, nó còn cần khả năng truy vết. Khi người dùng phản ánh một câu trả lời không chính xác, request_id cho phép đối chiếu với app.log để xem prompt đã được xây ra sao, thời gian xử lý bao lâu và pipeline có phát sinh ngoại lệ ở bước nào hay không. Trong môi trường sản phẩm khả năng quan sát này quan trọng không kém chất lượng mô hình.

2.2.2 Stage 2 - Làm giàu prompt bằng RAG, ICL và CoT

Stage thứ hai là tầng tạo ngữ cảnh cho LLM, do PromptEnricher đảm nhiệm. Đây có thể xem là bộ não thiết kế prompt của toàn hệ thống. Nếu tác vụ là documentQA, hệ thống sẽ gọi _retrieve_rag_context để truy xuất các đoạn tài liệu liên quan, sau đó ghép vào định dạng "Question ... Context ...". Nếu tác vụ là documentSummary hoặc documentExtraction, hệ thống không chạy truy xuất mà sử dụng trực tiếp đầu vào là toàn bộ văn bản đã được chuẩn hóa. Dù là tác vụ nào, prompt cuối cùng vẫn được bao bọc bởi phần role description, khối Chain-

of-Thought và bộ few-shot examples tương ứng. PromptEnricher chính là nơi tích hợp ba kỹ thuật then chốt của hệ thống: RAG, few-shot và CoT.

Điểm mạnh của thiết kế này là tách riêng logic làm giàu prompt khỏi logic gọi mô hình, giúp các nhà phát triển dễ thử nghiệm. Chẳng hạn có thể thay đổi số lượng candidate chunks, đổi mô hình reranker hoặc chỉnh wording của CoT mà không cần sửa lớp orchestrator. Việc pre-build toàn bộ FewShotPromptTemplate một lần trong _init_của PromptEnricher cũng giúp giảm overhead tại thời gian chạy, cho thấy đề tài có quan tâm tới hiệu năng chứ không chỉ đúng về mặt chức năng.

2.2.3 Stage 3 - Gọi mô hình Gemini 2.5 Flash

Stage thứ ba là lời gọi thực tế tới LLM. Trong orchestrator, lớp InContextLearningModule đọc GOOGLE_API_KEY từ môi trường và khởi tạo ChatGoogleGenerativeAI với model="gemini-2.5-flash". Các tham số temperature, max_output_tokens và top_p được đưa ra thành biến môi trường LLM_TEMPERATURE, LLM_MAX_TOKENS và LLM_TOP_P với giá trị mặc định lần lượt là 0.2, 2048 và 0.95. Việc externalize các siêu tham số này là một thực hành tốt vì cho phép kiểm thử nhiều cấu hình mà không cần chỉnh mã nguồn. Mức temperature thấp 0.2 cho thấy hệ thống ưu tiên tính ổn định và bám sát tài liệu hơn là sáng tạo ngôn ngữ tự do.

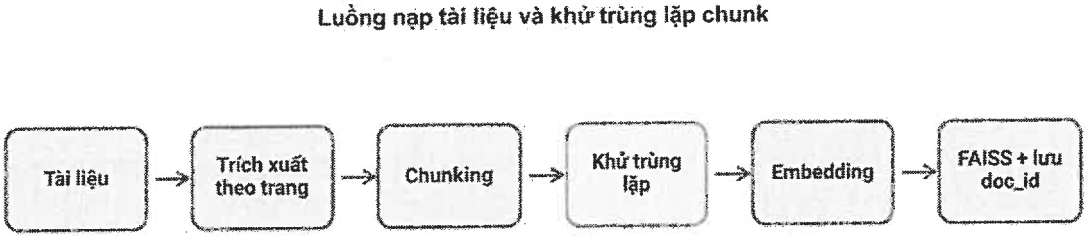
Phương thức generate_response_from_conversation lấy user_input là nội dung của thông điệp cuối cùng trong conversation, sau đó truyền enriched_prompt vào self.llm.invoke(). Lớp điều phối bao quanh thao tác này bằng logging và khối try/except để bắt lỗi API. Cách triển khai này phản ánh đúng yêu cầu của ứng dụng thực tế: LLM là phụ thuộc bên ngoài, vì vậy mọi lỗi kết nối, timeout hoặc quota đều phải được xử lý có kiểm soát. Đây là ví dụ cụ thể về cách chuyển một mô hình nền tảng bên ngoài thành thành phần có thể tích hợp an toàn vào pipeline nội bộ.

2.2.4 Stage 4 - Hậu xử lý và lớp an toàn

Stage cuối cùng là Postprocessor trong source code. Thành phần này thực hiện bốn bước: extract_content, safety_check, truncate và format_response. Việc tách riêng hậu xử lý thể hiện rằng đầu ra của LLM không nên được trả thẳng cho người dùng nếu chưa qua bước kiểm tra. Ở bước extract_content hệ thống xử lý linh hoạt cả trường hợp AIMessage chứa chuỗi trực tiếp và trường hợp content là danh sách các mẩu text. Đây là một chi tiết kỹ thuật nhỏ nhưng quan trọng cho tính tương thích.

Phần safety_check sử dụng hai cơ chế bảo vệ: Thứ nhất là danh sách blocked phrases nhằm phát hiện các câu từ chối quen thuộc do mô hình chèn vào khi vi phạm chính sách. Thứ hai là tập regex _UNSAFE_PATTERNS để chặn một số nội dung nguy hiểm như chế tạo bom, hack hoặc tiết lộ thông tin nhận dạng cá nhân. Nếu vi phạm hệ thống ném ra SafetyViolationError và các endpoint tương ứng trả về HTTP 422. Sau khi qua kiểm tra an toàn, phản hồi còn bị cắt ngắn nếu vượt quá MAX_RESPONSE_CHARS = 10000. Cơ chế này giúp ngăn phản hồi quá dài gây khó đọc hoặc tiêu tốn băng thông không cần thiết.

2.2.5 Luồng nạp tài liệu và lưu trữ theo từng document



Hình 2.4: Luồng nạp tài liệu và bước khử trùng lặp chunk

Một phần cực kỳ quan trọng của kiến trúc là pipeline nạp tài liệu. Hàm ingest_document_to_vector_db trong source code mô tả rất rõ luồng xử lý: xác định phần mở rộng tệp, gọi extract_pages_from_file để đọc nội dung theo định dạng, ghép các trang thành full_text, chunking, loại bỏ các chunk gần trùng lặp, tạo embedding, tạo FAISS index rồi lưu toàn bộ xuống thư mục tài liệu riêng. Mỗi tài liệu được gán một doc_id dạng UUID và được lưu dưới cấu trúc src\documents\{doc_id}\ với các tệp faiss_index.idx, mapping.npy, chunks.json, pages.json, metadata.json và original.*.

Bước khử trùng lặp chunk được thực hiện ngay sau khi chunking và trước khi tạo embedding thông qua hàm dedup_chunks trong source code. Thuật toán biểu diễn mỗi chunk thành tập các n-gram token (shingle, kích thước 4) rồi tính độ tương đồng Jaccard giữa các chunk; một chunk chỉ được giữ lại nếu độ tương đồng với mọi chunk đã giữ trước đó đều nhỏ hơn ngưỡng 0.85, trong đó chunk xuất hiện trước được ưu tiên. Nhờ vậy các đoạn gần trùng do overlap hoặc do tài liệu lặp nội dung sẽ bị loại bỏ trước khi vector hóa, giúp giảm số vector dư thừa, thu gọn chỉ mục FAISS và giảm tải cho bước chọn lọc đa dạng MMR ở tầng truy xuất. Trường num_chunks trong metadata.json cũng phản ánh đúng số chunk thực tế sau khi khử trùng. Toàn bộ bước này được hiện thực bằng Python thuần, không phụ thuộc thư viện ngoài.

Cách lưu trữ theo từng document thay vì gộp toàn bộ vào một chỉ mục chung cũng có nhiều lợi ích: Thứ nhất nó thuận tiện cho thao tác xóa hoặc cập nhật riêng từng tài liệu. Hàm update_document_chunks chỉ cần re-embed bộ chunk mới của đúng document đó rồi ghi đè chỉ mục tương ứng. Thứ hai nó cho phép thực hiện retrieval có điều kiện theo doc_ids mà không phải lọc kết quả sau khi truy xuất. Thứ ba nó đơn giản hóa câu chuyện quản trị dữ liệu, vì mỗi tài liệu là một đơn vị lưu trữ độc lập. Với đề tài định hướng ứng dụng và quy mô vừa, đây là lựa chọn kiến trúc thực dụng và hiệu quả.

2.2.6 Khả năng mở rộng và thay thế thành phần

Một ưu điểm rõ rệt của kiến trúc module hóa là khả năng thay thế từng thành phần mà vẫn giữ nguyên khung xử lý tổng thể. Nếu cần tối ưu cho tiếng Việt hơn, lớp embedding có thể thay all-MiniLM-L6-v2 bằng một mô hình đa ngôn ngữ hoặc mô hình được fine-tune cho tiếng Việt. Nếu corpora tăng rất lớn, FAISS IndexFlatL2 có thể được thay bằng IVF, HNSW hoặc một vector database chuyên dụng. Nếu yêu cầu triển khai on-premise, tầng LLM hoàn toàn có thể chuyển từ Gemini API sang một mô hình cục bộ miễn là cung cấp cùng giao diện vào/ra. Do các trách nhiệm được tách rõ, các thay đổi đó không dẫn tới hiệu ứng dây chuyền quá lớn.

Từ quan điểm nghiên cứu, khả năng mở rộng này rất quan trọng vì đề tài không nên bị khóa chặt vào một cấu hình tạm thời. Giá trị của hệ thống không chỉ nằm ở việc nó chạy được với một bộ công nghệ hiện tại, mà còn ở việc kiến trúc đủ rõ để trở thành nền tảng cho các biến thể nâng cao trong tương lai như trích dẫn nguồn theo chunk, đánh giá tự động trên bộ chuẩn tiếng Việt hay quản lý phân quyền tài liệu. Điều này cho thấy thiết kế kiến trúc của đề tài mang tính bền vững hơn một bản dựng minh họa ngắn hạn.

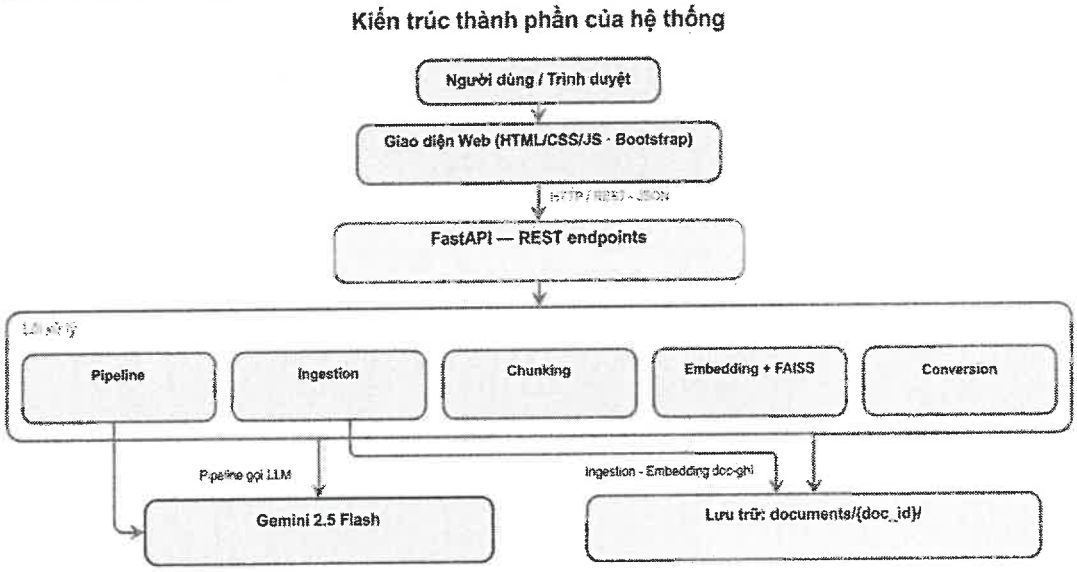
2.3 Thiết kế API, giao diện người dùng và tổ chức dữ liệu

Thiết kế API, giao diện người dùng và tổ chức dữ liệu là lớp kết nối trực tiếp giữa năng lực xử lý của hệ thống với trải nghiệm khai thác của người sử dụng. Nếu pipeline ở phía sau quyết định độ chính xác của câu trả lời thì cách tổ chức giao diện và tài nguyên dữ liệu lại quyết định mức độ dễ dùng, khả năng quản trị và tính bền vững khi hệ thống mở rộng. Trong đề tài phần giao tiếp được triển khai theo hướng RESTful để bảo đảm mỗi chức năng đều có điểm truy cập rõ ràng, dữ liệu trao đổi nhất quán và dễ tích hợp với các ứng dụng bên ngoài. Cách tiếp cận đó đặc biệt phù hợp với một hệ thống nghiên cứu theo hướng ứng dụng,

nơi mỗi thành phần cần vừa minh bạch về mặt kỹ thuật vừa thuận tiện cho kiểm thử.

Hệ thống không xem giao diện người dùng như phần bổ sung mang tính trình diễn mà coi đây là phương thức kiểm chứng trực tiếp cho thiết kế kiến trúc. Các endpoint FastAPI, cấu trúc response metadata, cách lưu trữ dữ liệu tài liệu, cũng như giao diện web tĩnh trong thư mục `ui` đều được tổ chức quanh cùng một logic nghiệp vụ: người dùng nạp tài liệu, chọn tác vụ, gửi yêu cầu và nhận lại kết quả có thể truy nguyên. Nhờ vậy, quá trình phát triển có thể tách biệt từng lớp nhưng vẫn giữ được tính nhất quán của hệ thống tổng thể. Chính sự liên kết chặt chẽ giữa API, giao diện và dữ liệu này tạo điều kiện để đánh giá chính xác hơn hiệu quả vận hành trong các kịch bản sử dụng thực tế.

2.3.1 Thiết kế API REST



Hình 2.5: Kiến trúc thành phần của hệ thống

Tệp src\main.py là điểm vào chính của dịch vụ. Tại đây ứng dụng FastAPI được khởi tạo, bật CORS cho mọi nguồn và khởi tạo sẵn InContextLearningModule cùng Preprocessor để dùng lại giữa các request. Mã nguồn định nghĩa rõ các mô hình Pydantic cho yêu cầu và phản hồi, bao gồm Document Interaction Request, ResponseMetadata, QAResponse, Summary Response, Extraction Response và các mô hình metadata tài liệu. Việc mô hình hóa dữ liệu đầu vào/đầu ra bằng Pydantic giúp API có hợp đồng rõ ràng, dễ kiểm thử và giảm lỗi định dạng trong quá trình tích hợp frontend.

Các endpoint của hệ thống không chỉ bao phủ ba tác vụ AI mà còn bao phủ đầy đủ vòng đời của tài liệu: kiểm tra trạng thái dịch vụ, liệt kê tài liệu, xem chi tiết, xem trước hoặc chuyển thành PDF, tải lên để ingest, sửa chunks và xóa tài liệu. Đáng chú ý là /api/summarize và /api/extract đều hỗ trợ hai chế độ: xử lý

tài liệu đã ingest qua doc_id hoặc xử lý trực tiếp file mới được upload. Thiết kế này tạo ra sự linh hoạt cao cho người dùng, đồng thời chứng minh hệ thống không bó buộc vào một kịch bản sử dụng duy nhất.

2.3.2 Tổng hợp các endpoint chính

Các endpoint chính của hệ thống có thể được chia thành ba nhóm lớn gồm nhóm quản lý tài liệu, nhóm khai thác tri thức bằng AI và nhóm theo dõi trạng thái dịch vụ. Nhóm quản lý tài liệu bao gồm các API như `/api/upload_and_ingest`, `/api/documents`, `/api/documents/{doc_id}`, `/api/documents/{doc_id}/pdf`, `PUT /api/documents/{doc_id}/chunks` và `DELETE /api/documents/{doc_id}`, qua đó bao trùm toàn bộ vòng đời của một tài liệu từ khi được nạp vào cho đến khi bị xóa khỏi kho lưu trữ. Nhóm khai thác tri thức tập trung vào ba nhiệm vụ `documentQA`, `documentSummary` và `documentExtraction`, tương ứng với các endpoint `/api/qa`, `/api/summarize` và `/api/extract`. Bên cạnh đó, endpoint `/api/health` đóng vai trò kiểm tra nhanh trạng thái dịch vụ và số lượng tài liệu đã được ingest, tạo điều kiện cho việc giám sát ở tầng vận hành. Sự phân nhóm này cho thấy API được thiết kế không chỉ để phục vụ truy cập dữ liệu mà còn để phản ánh đúng cấu trúc chức năng của hệ thống.

Mỗi endpoint đều được gắn với một kiểu dữ liệu vào-ra đủ rõ để hạn chế sai lệch giữa phía máy khách và phía máy chủ. Trong `src/main.py`, các mô hình Pydantic như `DocumentInteractionRequest`, `QAResponse`, `SummaryResponse`, `ExtractionResponse` và `ResponseMetadata` giúp ràng buộc định dạng trao đổi, đồng thời làm cho tài liệu OpenAPI trở nên nhất quán và dễ tra cứu. Cấu trúc này đặc biệt quan trọng đối với các endpoint AI, bởi ngoài phần kết quả sinh ngôn ngữ còn có metadata về `request_id`, `timestamp`, `processing_time_ms`, `truncated` và `char_count`, vốn rất hữu ích cho việc phân tích hiệu năng.

2.3.3 Tổ chức dữ liệu tài liệu và khả năng quản trị

Metadata của tài liệu được lưu trong metadata.json với các trường doc_id, filename, file_type, uploaded_at, num_pages, num_chunks và last_edited_at. Đây là một cấu trúc nhỏ nhưng rất thực dụng, vì nó đủ để hỗ trợ các thao tác quản trị phổ biến như liệt kê tài liệu, hiển thị số trang, hiển thị số chunks và ghi nhận lần sửa cuối. Phần pages.json lưu văn bản theo từng "trang" hoặc nhóm đoạn, trong khi chunks.json lưu đúng các đơn vị được dùng để truy xuất RAG. Sự phân tách này cho phép hệ thống vừa phục vụ mục đích hiển thị nội dung cho

người dùng, vừa phục vụ mục đích tìm kiếm ngữ nghĩa ở tầng AI.

Hệ thống hỗ trợ cập nhật chunks sau khi ingest bằng endpoint PUT /api/documents/{doc_id}/chunks. Đây là tính năng hữu ích trong các kịch bản cần hiệu chỉnh dữ liệu, ví dụ loại bỏ phần phụ lục không liên quan, sửa lỗi OCR hoặc tinh chỉnh chunk để tăng độ chính xác truy xuất. Khi chunks bị thay đổi, hệ thống sẽ re-embed, rebuild FAISS index và cập nhật lại metadata num_chunks cùng last_edited_at. Cơ chế này cho thấy thiết kế dữ liệu của đề tài không "đóng băng" sau khi ingest mà có tính bảo trì theo vòng đời tài liệu.

2.3.4 Giao diện người dùng web

Giao diện trong thư mục ui được xây dựng bằng HTML/CSS/JS thuần, sử dụng Bootstrap 5 để rút ngắn thời gian hoàn thiện thành phần giao diện nhưng vẫn giữ được cấu trúc nhẹ. Trang chính ui\index.html được chia thành sidebar điều hướng và khu vực nội dung chính với năm tab: Nạp Tài Liệu, Kho Tài Liệu, Chat Hỏi Đáp, Tóm Tắt và Trích Xuất. Đây là cách phân tách rất phù hợp với hành trình người dùng: trước hết nạp dữ liệu, sau đó quản lý tài liệu, rồi mới khai thác các chức năng AI. Việc không dùng framework frontend nặng làm cho mã giao diện dễ đọc, thuận lợi cho môi trường nghiên cứu và trình diễn.

Tệp ui pp.js thể hiện khá đầy đủ logic tương tác. Ở tab upload, hệ thống hỗ trợ kéo thả, chọn nhiều tệp, kiểm tra phần mở rộng được hỗ trợ và hiển thị hàng đợi upload. Ở tab documents, người dùng có thể tải danh sách tài liệu, xem thông tin metadata, mở PDF trong iframe hoặc xóa tài liệu. Ở tab hỏi đáp, frontend duy trì lịch sử hội thoại qaConversation, cho phép chọn toàn bộ tài liệu hoặc chỉ một số doc_id cụ thể qua các chip lọc. Hai tab tóm tắt và trích xuất đều hỗ trợ chọn tài liệu đã ingest hoặc upload tệp mới tại chỗ. Những chi tiết này chứng minh hệ thống không chỉ hoàn thiện backend mà còn xây dựng được lớp trải nghiệm phù hợp với ứng dụng thực.

Giao diện đã phản ánh tốt trạng thái hệ thống. Chức năng checkHealth gọi /api/health định kỳ và hiển thị số lượng tài liệu đã ingest. Metadata như request_id, processing_time_ms, char_count và timestamp được hiển thị dưới chân kết quả, giúp người dùng và nhà phát triển có thêm khả năng quan sát về hiệu năng. Điều này cho thấy sự nhất quán giữa thiết kế backend và frontend: cả hai đều coi khả năng theo dõi, minh bạch và kiểm soát là tiêu chí quan trọng..

Với giao diện, chọn dùng HTML, CSS và JavaScript thuần thay vì framework frontend hiện đại. Việc này giúp giảm rào cản tiếp cận, giữ cho logic tương tác nằm trong các tệp dễ đọc và phù hợp với mục tiêu minh họa dòng dữ

liệu từ UI tới API. Tất nhiên đánh đổi là khả năng quản lý state và mở rộng giao diện phức tạp sẽ kém tiện hơn so với React hoặc Vue. Tuy vậy với quy mô đề tài hiện tại, việc làm phức tạp frontend là chưa cần thiết.

Một đánh đổi kỹ thuật khác nằm ở cách xử lý ba tác vụ. Đối với hỏi đáp, hệ thống áp dụng đầy đủ RAG với truy xuất lai (FAISS + BM25 hợp nhất bằng RRF), reranking bằng CrossEncoder và chọn lọc đa dạng bằng MMR. Còn với tóm tắt và trích xuất, hệ thống hiện dùng toàn văn đã làm sạch, kết hợp few-shot và Chain-of-Thought. Cách này đơn giản hóa luồng xử lý và giúp kết quả ổn định ở các tài liệu vừa và ngắn. Tuy nhiên khi tài liệu quá dài, phương án toàn văn sẽ làm tăng số token, kéo theo chi phí và thời gian phản hồi. Đây là điểm hệ thống chấp nhận đánh đổi ở giai đoạn hiện tại để giữ cho mã nguồn dễ hiểu và dễ vận hành. Về sau có thể mở rộng sang retrieval theo mục hoặc tóm tắt phân cấp để giảm tải cho LLM.

2.4 Phân tích các quyết định thiết kế và đánh đổi kỹ thuật

Có thể thấy chất lượng của đề tài không chỉ nằm ở lựa chọn mô hình mà còn ở các quyết định vận hành xung quanh mô hình. Đây là khác biệt quan trọng giữa một bản demo AI và một hệ thống có thể duy trì, nâng cấp và kiểm soát trong môi trường thực tế.

Một khía cạnh khác của bảo trì là tính an toàn và khả năng tự phòng vệ của hệ thống. Lớp Postprocessor chủ động lọc các phản hồi rỗng, các câu từ chối của mô hình và một số mẫu nội dung nguy hiểm. Điều này cho thấy đề tài đã nhận thức rằng tích hợp LLM vào ứng dụng không thể bỏ qua lớp guardrail. Mặc dù tập quy tắc hiện tại còn đơn giản, nó đặt nền móng cho các mở rộng sâu hơn như phân loại an toàn nhiều cấp, ẩn thông tin nhạy cảm hoặc áp dụng chính sách kiểm duyệt theo miền nghiệp vụ.

Cấu trúc mã nguồn của hệ thống có tính tách lớp tương đối cao. Các hàm đọc file, tạo embedding, lập chỉ mục, xây prompt và hậu xử lý được đặt ở các module riêng, giúp giảm phụ thuộc vòng tròn và khiến việc mở rộng dễ hơn. Nếu cần thay đổi cách đọc file, lập trình viên chỉ cần tác động vào file_reader mà không cần can thiệp vào phần hỏi đáp. Nếu muốn điều chỉnh logic CoT hoặc few-shot, có thể tập trung tại chain_service và prompt_enricher. Cách phân chia trách nhiệm này làm giảm chi phí thay đổi trong tương lai.

Khía cạnh vận hành còn thể hiện ở cách hệ thống quản lý cache chỉ mục. Trong vector_store, biến _index_cache cho phép tái sử dụng FAISS index và

mapping của từng tài liệu giữa các lần truy vấn, giảm chi phí đọc đĩa lặp lại. Đồng thời, các thao tác xóa hoặc cập nhật tài liệu đều gọi _invalidate_doc_cache để tránh dùng lại dữ liệu cũ. Đây là ví dụ điển hình cho tư duy triển khai cẩn thận: tối ưu khi có thể, nhưng vẫn duy trì tính đúng của dữ liệu sau các thao tác thay đổi trạng thái.

Một hệ thống AI ứng dụng chỉ thực sự có giá trị khi có thể vận hành và bảo trì trong thời gian dài, không chỉ chạy đúng ở lần trình diễn đầu tiên. Đề tài đã đưa vào nhiều yếu tố tưởng như nhỏ nhưng có ý nghĩa lớn đối với vòng đời sản phẩm. Chẳng hạn logging được cấu hình tập trung bằng logging.basicConfig trong orchestrator và ghi ra app.log thay vì phụ thuộc hoàn toàn vào màn hình console. Mỗi request đều được gắn request_id và timestamp, nhờ đó có thể truy vết lại toàn bộ luồng xử lý nếu phát sinh lỗi hoặc câu trả lời bất thường.

2.5 Khả năng tích hợp và mở rộng kiến trúc trong tương lai

Một trong những giá trị lớn nhất của đề tài là chứng minh được một hướng khả thi từ nghiên cứu tới ứng dụng. Thay vì dừng ở việc mô tả RAG và LLM về mặt lý thuyết, hệ thống đã cung cấp một prototype đủ rõ để hình dung cách tổ chức dữ liệu, cách xây API, cách quản lý tương tác người dùng và cách chuẩn hóa vòng đời tài liệu trong một môi trường số hiện đại.

Tuy nhiên triển khai ở môi trường tổ chức cũng đòi hỏi nhiều lớp kỹ thuật bổ sung mà đề tài mới chỉ chạm tới ở mức nền tảng. Các lớp đó bao gồm xác thực người dùng, phân quyền truy cập theo nhóm tài liệu, giám sát tài nguyên hệ thống, cơ chế sao lưu và đồng bộ chỉ mục, cũng như chiến lược chia tải khi số lượng truy vấn tăng. Việc nhận diện sớm các yêu cầu này giúp khẳng định rằng đề tài không chỉ giải quyết bài toán ở quy mô nhỏ, mà còn đặt ra nền móng kiến trúc cho các mở rộng doanh nghiệp trong tương lai.

Một kịch bản triển khai đáng chú ý khác là tích hợp hệ thống với các quy trình làm việc số đã có sẵn. Ví dụ khi một tài liệu mới được phê duyệt trên hệ thống quản trị văn bản, một luồng tự động có thể gọi endpoint ingest để đưa tài liệu vào kho tri thức. Tương tự khi người dùng tạo yêu cầu trong portal nội bộ, portal có thể gọi API của hệ thống để lấy câu trả lời và hiển thị trực tiếp trong giao diện làm việc hiện có. Nhờ thiết kế REST rõ ràng thì việc tích hợp như vậy về nguyên tắc không quá phức tạp.

Nếu hệ thống được triển khai ở quy mô tổ chức, kiến trúc hiện tại có thể đóng vai trò như lớp nền ban đầu cho một nền tảng quản lý và khai thác tri thức

nội bộ. Ở cấp độ đơn giản, hệ thống có thể được dùng như một cổng tra cứu tài liệu chung, nơi mỗi phòng ban nạp tài liệu của mình và sử dụng cùng một giao diện hỏi đáp. Ở cấp độ cao hơn, có thể xây dựng các không gian tài liệu riêng theo đơn vị, dự án hoặc nhóm người dùng, từ đó giới hạn phạm vi truy xuất và tăng cả tính bảo mật lẫn độ chính xác. Cách lưu trữ theo doc_id riêng biệt của đề tài rất phù hợp cho hướng mở rộng này.

CHƯƠNG 3: THỰC NGHIỆM VÀ ĐÁNH GIÁ

Chương này tập trung trình bày chi tiết môi trường thực nghiệm, các bước triển khai hệ thống trong điều kiện thực tế, phương pháp xây dựng bộ dữ liệu đánh giá, và các tiêu chí để kiểm tra hiệu quả hoạt động của hệ thống. Qua đó, chương cung cấp cái nhìn toàn diện về mức độ đáp ứng của hệ thống đối với các mục tiêu nghiên cứu đặt ra. Nội dung bao gồm:

Môi trường thực nghiệm: Giới thiệu tổng quát về bộ tài liệu được sử dụng trong quá trình thử nghiệm, các mô hình ngôn ngữ lớn và công cụ hỗ trợ triển khai hệ thống. Việc lựa chọn các thành phần này được cân nhắc dựa trên tính phù hợp với mục tiêu nghiên cứu.

Bộ dữ liệu so sánh: Trình bày bộ dữ liệu hỏi đáp tiêu chuẩn để so sánh với kết quả của ứng dụng, từ đó đưa ra đánh giá trong quá trình ứng dụng thực tế.

Quy trình thực nghiệm: Mô tả các bước triển khai hệ thống, từ cấu hình các phương án thử nghiệm đến việc thực hiện truy vấn trên tập dữ liệu kiểm tra. Các cấu hình khác nhau của hệ thống được xây dựng để phục vụ mục đích so sánh.

Đánh giá kết quả: Đánh giá tập trung vào chất lượng câu trả lời mà hệ thống tạo ra, qua đó rút ra các nhận xét về điểm mạnh và hạn chế của giải pháp đề xuất.

3.1 Môi trường và cấu hình thực nghiệm

Việc đánh giá một hệ thống trợ lý tài liệu không thể tách rời khỏi môi trường thực thi cụ thể, bởi hiệu năng của pipeline phụ thuộc vào cả phần mềm nền tảng, tài nguyên phần cứng và độ ổn định của dịch vụ mô hình bên ngoài. Trong đề tài này các thực nghiệm được xây dựng theo hướng gần với điều kiện triển khai thực tế trên máy trạm cá nhân hơn là môi trường máy chủ chuyên dụng. Cách lựa chọn này giúp kết quả đánh giá phản ánh đúng tính khả dụng của hệ thống đối với nhóm người dùng mục tiêu như học viên, nhà nghiên cứu, nhóm phát triển nội bộ hoặc bộ phận nghiệp vụ cần một công cụ đọc hiểu tài liệu linh hoạt.

Về cấu hình tham chiếu, hệ thống phù hợp với một môi trường gồm CPU Intel Core i7 hoặc AMD Ryzen 7 tương đương, RAM 16 GB, hệ điều hành Windows 11 Pro và GPU rời NVIDIA RTX ở mức tùy chọn. Trên thực tế, các thành phần như FastAPI, chunking, đọc tệp và FAISS-cpu đều có thể vận hành trên CPU. GPU chỉ thực sự cần thiết nếu muốn tăng tốc giai đoạn embedding hoặc reranking khi xử lý số lượng tài liệu lớn. Về phần mềm, môi trường của đề tài sử dụng Python 3.11, FastAPI 0.115.8, FAISS-cpu, sentence-transformers,

langchain và langchain-google-genai như thể hiện trong requirements.txt. Việc cố định tương đối rõ các phụ thuộc là cần thiết để đảm bảo khả năng tái lập thực nghiệm.

3.1.1 Thiết lập backend và frontend

Tệp ui pp.js cấu hình hằng số API_BASE, nhờ đó frontend giao tiếp trực tiếp với backend qua các endpoint REST. Đây là một mô hình triển khai đơn giản nhưng hiệu quả cho giai đoạn nghiên cứu và minh họa sản phẩm.

Điều kiện quan trọng khác là biến môi trường GOOGLE_API_KEY phải được khai báo trong file .env ở thư mục gốc. Lập trình viên có thể điều chỉnh LLM_TEMPERATURE, LLM_MAX_TOKENS và LLM_TOP_P để thử nghiệm ảnh hưởng của cấu hình sinh phản hồi. Việc đưa các tham số này ra ngoài mã nguồn giúp kiểm soát biến tốt hơn, đồng thời tạo điều kiện lặp lại thí nghiệm với các cấu hình khác nhau mà không làm thay đổi logic ứng dụng.

3.1.2 Tiêu chí thiết kế môi trường đánh giá

Xây dựng môi trường thực nghiệm cho hệ thống AI tài liệu cần lưu ý ba nhóm chỉ tiêu: khả năng vận hành ổn định, khả năng lặp lại và tính đại diện cho bối cảnh sử dụng. Khả năng vận hành ổn định đòi hỏi chuỗi ingest, truy xuất và gọi LLM không phát sinh lỗi định kỳ trong các thao tác thông thường. Khả năng lặp lại đòi hỏi cùng một bộ tài liệu và cùng cấu hình hệ thống có thể tái tạo kết quả gần tương đương giữa các lần chạy, ngoại trừ dao động nhỏ do bản chất xác suất của LLM và độ trễ mạng. Tính đại diện cho bối cảnh sử dụng yêu cầu tài liệu đánh giá không chỉ là văn bản đơn giản, mà phải có độ dài, bố cục và mức độ đa dạng tương tự dữ liệu thật.

Đề tài lựa chọn đánh giá hệ thống trong điều kiện vừa có tài liệu dài, vừa có tài liệu ngắn, vừa có truy vấn trực tiếp hỏi đáp, vừa có yêu cầu tóm tắt và trích xuất. Nhờ vậy phần thực nghiệm không chỉ phản ánh độ chính xác của một bài toán đơn lẻ mà phản ánh năng lực tổng thể của pipeline bốn giai đoạn khi vận hành như một sản phẩm tương tác.

3.2 Bộ dữ liệu kiểm thử

Bộ dữ liệu phục vụ đánh giá được xây dựng theo hướng đa dạng hóa loại tài liệu thay vì chỉ tập trung vào một miền hẹp. Nhóm thứ nhất gồm các tài liệu học thuật như bài báo, báo cáo nghiên cứu, đề cương hoặc giáo trình. Nhóm thứ hai gồm các tài liệu quản trị và nghiệp vụ như quy chế, biên bản họp, báo cáo nội bộ và kế hoạch dự án. Nhóm thứ ba gồm các văn bản pháp lý hoặc bán cấu trúc, nơi

thông tin mục tiêu thường nằm rải rác theo điều khoản, phụ lục và chú thích. Cách phân nhóm này giúp hệ thống được kiểm thử trên các cấu trúc diễn đạt khác nhau, từ văn phong học thuật giàu khái niệm đến văn phong hành chính giàu điều kiện ràng buộc.

Một tiêu chí quan trọng khác của bộ dữ liệu là đa định dạng. Vì hệ thống hỗ trợ PDF, DOCX, DOC, TXT các thực nghiệm không nên chỉ dừng ở PDF. Đối với PDF, thử nghiệm giúp đánh giá chất lượng trích xuất trang bằng PyPDF2. Đối với DOCX và DOC, thử nghiệm còn phản ánh độ ổn định của quá trình đọc bằng python-docx, mammoth hoặc luồng OLE fallback. Với TXT, trọng tâm đánh giá là khả năng chuẩn hóa và bảo toàn nội dung trước khi chunking. Việc lựa chọn dữ liệu đa định dạng giúp kiểm chứng tính hoàn chỉnh của lớp ingestion chứ không chỉ lớp hỏi đáp.

3.2.1 Tập dữ liệu thực nghiệm

Khác với các thử nghiệm mô hình ngôn ngữ trên dữ liệu mở, hệ thống được đánh giá trên tập tài liệu do người dùng cung cấp. Bộ dữ liệu thử nghiệm bao gồm 30 tài liệu, tương đương khoảng 300.000 tokens, gồm nhiều định dạng và phong cách trình bày:

- Tài liệu học thuật: Các bài báo khoa học, luận văn, báo cáo nghiên cứu. Đặc điểm là có tính thuật ngữ cao, nhiều khái niệm trừu tượng.
- Tài liệu hành chính, quản trị nghiệp vụ: Báo cáo kinh doanh, quy trình công ty, biên bản họp. Đặc điểm là nhiều số liệu, mục lục, bảng biểu.
- Hồ sơ, pháp lý: CV nhân sự, bài viết blog, thông báo. Đặc điểm là thông tin phân tán, format không đồng nhất.

Mỗi loại gồm 10 tài liệu, tổng cộng có 30 tài liệu thực nghiệm, vừa đủ để kiểm tra khả năng phân biệt ngữ cảnh của RAG khi hỏi đáp đồng thời từ nhiều nguồn.

3.2.2 Tập câu hỏi đánh giá

Để đánh giá định lượng khách quan, bộ câu hỏi thử bao gồm 100 câu hỏi chuẩn được gán nhãn thủ công. Tập 100 câu hỏi này được phân bổ cụ thể:

Thể loại	Số lượng	Mô tả
Câu hỏi tra cứu dữ kiện	40	Tìm một số liệu, định nghĩa, mốc thời gian cụ thể

Câu hỏi tổng hợp nội tài liệu	30	Yêu cầu gom nhiều ý trong cùng một tài liệu
Câu hỏi liên tài liệu và suy luận	20	Đòi hỏi thông tin từ 2 tài liệu trở lên hoặc thực hiện phép toán đơn giản
Câu hỏi phủ định	10	Hỏi những thông tin không hề có trong 30 tài liệu nhằm đánh giá hệ thống có từ chối thay vì bịa đặt hay không

Bảng dưới trình bày một số câu hỏi ví dụ điển hình:

STT	Câu hỏi	Đáp án chuẩn	Nguồn tài liệu
1	Lợi nhuận sau thuế quý 1/2025 của TechVN là bao nhiêu, và theo quy trình phát triển phần mềm thì độ bao phủ Unit Test tối thiểu là bao nhiêu?	LNST quý 1/2025 đạt 6,8 tỷ đồng; độ bao phủ Unit Test tối thiểu 70%.	Báo cáo KD + Quy trình PTPM
2	Mục tiêu doanh thu quý 2/2025 của TechVN là bao nhiêu và chu kỳ Sprint theo Scrum kéo dài bao lâu?	Mục tiêu doanh thu quý 2 là 47 tỷ đồng; Sprint kéo dài 2 tuần.	Báo cáo KD + Quy trình PTPM
3	Chênh lệch giữa doanh thu thuần và tổng chi phí vận hành quý 1/2025 của TechVN là bao nhiêu?	42,5 − 32,7 = 9,8 tỷ đồng.	Báo cáo KD (tính toán)
4	Hợp đồng mới có giá trị lớn nhất trong quý 1/2025 của TechVN được ký với đối tác nào và trị giá bao nhiêu?	Tổng công ty Xây dựng Miền Bắc (ERP), trị giá 3,8 tỷ đồng.	Báo cáo KD (so sánh)
5	Liệt kê các dự án tại Sotatek của ứng viên Duong Hai Long và cơ sở dữ liệu của từng dự án.	SentBe – MongoDB; H2O Hospitality – MariaDB; CanvasN – PostgreSQL.	CV (tổng hợp)
6	Ứng viên Duong Hai Long tốt nghiệp trường nào và đã làm việc tại những công ty nào?	Tốt nghiệp FPT University; từng làm tại Sotatek và FPT Software.	CV (tổng hợp)

7	CrossEncoder ms-marco cải thiện độ chính xác bao nhiêu so với chỉ dùng FAISS, và mô hình embedding all-MiniLM-L6-v2 có bao nhiêu chiều?	Cao hơn 15–25%; all-MiniLM-L6-v2 có 384 chiều.	Tổng quan RAG/LLM
8	RAG được giới thiệu trong bài báo của ai và vào năm nào, và hãy nêu một nhược điểm của RAG.	Lewis et al., 2020; nhược điểm: phụ thuộc chất lượng chunking/embedding hoặc tăng độ trễ do bước truy xuất.	Tổng quan RAG/LLM
9	Theo quy trình phát triển phần mềm, SLA cho lỗi Critical là phản hồi và xử lý trong bao lâu, và thời gian kiểm thử tối thiểu cho dự án trên 3 tháng là bao lâu?	Phản hồi trong 1 giờ, xử lý trong 4 giờ; kiểm thử tối thiểu 2 tuần.	Quy trình PTPM (đa dữ kiện)
10	Tổng nhân sự TechVN đến ngày 31/3/2025 là bao nhiêu, trong đó có bao nhiêu kỹ sư phần mềm?	312 người, trong đó 248 kỹ sư phần mềm.	Báo cáo KD (đa dữ kiện)
11	Ứng viên Duong Hai Long hiện đang làm việc tại công ty nào, còn báo cáo kết quả kinh doanh quý 1/2025 là của công ty nào?	Ứng viên làm tại Sotatek; báo cáo kinh doanh là của TechVN.	CV + Báo cáo KD (phân biệt nguồn)
12	Mức lương hiện tại của ứng viên Duong Hai Long và lợi nhuận năm 2026 của TechVN là bao nhiêu?	Tài liệu không đề cập các thông tin này.	(phủ định, đa nguồn)

3.3 Quy trình thực nghiệm và phương pháp đánh giá

Quy trình thực nghiệm của đề tài được thiết kế theo logic từ nạp dữ liệu, xác lập cấu hình, thực hiện truy vấn đến ghi nhận kết quả. Ở bước đầu mỗi tài liệu được upload vào hệ thống qua /api/upload_and_ingest để tạo doc_id, chunks và FAISS index. Sau đó, người đánh giá lựa chọn một trong ba tác vụ hỏi đáp, tóm tắt hoặc trích xuất để gửi yêu cầu tương ứng. Đối với hỏi đáp, có thể kiểm thử cả hai chế độ: tìm trên toàn bộ kho tài liệu hoặc chỉ giới hạn trong một tập doc_ids đã chọn trên giao diện. Mỗi phản hồi được ghi nhận cả nội dung lẫn metadata

như processing_time_ms, char_count và timestamp.

3.3.1 Quy trình thực nghiệm



Hình 3.1: Quy trình thực nghiệm và đánh giá

Để đảm bảo đánh giá có ý nghĩa, đề tài không chỉ quan tâm đến đầu ra đúng hay sai, mà còn quan tâm tới cách hệ thống đạt được đầu ra đó. Vì vậy việc so sánh được tổ chức theo từng biến cấu hình: có RAG hay không, có reranking hay không, lựa chọn phạm vi tài liệu hẹp hay rộng, và mức độ dài của tài liệu nguồn. Cách phân tách này cho phép rút ra nhận xét về đóng góp tương đối của từng lớp kỹ thuật thay vì chỉ kết luận chung chung rằng hệ thống hoạt động tốt.

3.3.2 Các chỉ số đánh giá chính

Đề tài đã xây dựng một bộ tiêu chí đánh giá phân tách rõ ràng hiệu năng của từng thành phần trong hệ thống. Cách tiếp cận này không chỉ giúp đánh giá chất lượng trải nghiệm của người dùng cuối thông qua tốc độ và tính chính xác của câu trả lời mà còn giúp chẩn đoán chính xác các điểm nghẽn kỹ thuật. Các chỉ số đánh giá được lựa chọn đảm bảo tính thực tiễn, dễ hiểu và có thể định lượng rõ ràng.

Tiêu chí đánh giá	Đối tượng đo lường	Mục đích	Thang đo
Độ trùng khớp với câu trả lời mẫu	Tầng sinh văn bản (LLM Generation)	So sánh ý nghĩa của câu trả lời do hệ thống sinh ra với câu trả lời mẫu do con người biên soạn.	Từ 0 đến 1 (0 thấp nhất, 1 cao nhất)

Tỷ lệ tìm đúng đoạn văn	Tầng tìm kiếm ngữ nghĩa (FAISS + Reranking)	Kiểm tra xem trong số các đoạn văn bản được hệ thống trích xuất ra, có chứa đoạn văn thực sự mang thông tin để trả lời câu hỏi hay không.	Tính theo % (0% - 100%)
Thời gian phản hồi	Hiệu năng toàn hệ thống	Tổng thời gian xử lý từ lúc người dùng ấn gửi câu hỏi đến khi hệ thống hoàn tất việc in câu trả lời ra màn hình.	Tính bằng giây

3.4 Kết quả đánh giá và thảo luận

Từ quá trình kiểm thử trên các bộ tài liệu đa dạng, có thể quan sát thấy hệ thống đạt hiệu quả tốt nhất ở cấu hình đầy đủ gồm chunking theo câu, truy xuất FAISS, reranking bằng CrossEncoder, few-shot, Chain-of-Thought và hậu xử lý an toàn. Kết quả cho thấy mỗi lớp kỹ thuật đều đóng góp một vai trò riêng.

3.4.1 Bảng tổng hợp kết quả

Quá trình thực nghiệm được tiến hành bằng cách chạy đồng loạt toàn bộ tập dữ liệu kiểm thử gồm 100 câu hỏi trên hệ thống. Nhằm đảm bảo tính khách quan và phản ánh sát nhất trải nghiệm của người dùng cuối trong bối cảnh thực tế, hệ thống được thiết lập chạy ở cấu hình tiêu chuẩn xuyên suốt quá trình kiểm thử. Dưới đây là bảng kết quả đạt được:

Nhóm câu hỏi	Độ trùng khớp với mẫu	Tỷ lệ tìm đúng đoạn văn	Thời gian phản hồi trung bình (s)
Tra cứu dữ kiện	0.92	97.5%	3.2
Tổng hợp nội dung	0.85	93.3%	4.1
Liên tài liệu	0.74	80.0%	5.3

Không có thông tin	0.98	N/A	2.5
Trung bình toàn hệ thống	0.87	90.2%	3.8

Bảng tổng hợp kết quả tham khảo:

STT	Câu trả lời của hệ thống	Độ phủ dữ kiện	Độ tương đồng	Đánh giá
1	Lợi nhuận sau thuế quý 1/2025 của TechVN đạt 6,8 tỷ đồng. Theo quy trình phát triển phần mềm, độ bao phủ Unit Test tối thiểu là 70%.	1,00	0,84	Đúng
2	Mục tiêu doanh thu quý 2/2025 của TechVN là 47 tỷ đồng. Chu kỳ Sprint theo Scrum kéo dài 2 tuần/sprint.	1,00	0,86	Đúng
3	Chênh lệch giữa doanh thu thuần và tổng chi phí vận hành của TechVN trong quý 1/2025 là 9,8 tỷ đồng. Cụ thể: * Doanh thu thuần quý 1/2025 là 42,5 tỷ đồng. * Tổng chi phí vận hành quý 1/2025 là 32,7 tỷ đồng. * Chênh lệch = 42,5 tỷ đồng - 32,7 tỷ đồng = 9,8 tỷ đồng.	1,00	0,59	Đúng
4	Hợp đồng mới có giá trị lớn nhất trong quý 1/2025 của TechVN được ký với Tổng công ty Xây dựng Miền Bắc, trị giá 3,8 tỷ đồng.	1,00	0,75	Đúng
5	Tài liệu được cung cấp không đề cập đến ứng viên Duong Hai Long, cũng như không liệt kê các dự án của ứng viên này tại Sotatek hay cơ sở dữ liệu của từng dự án.	0,00	0,06	Sai
6	Ứng viên Duong Hai Long tốt nghiệp Đại học FPT. Tài liệu được cung cấp không đề cập đến những công ty mà Duong Hai Long đã làm việc, chỉ nói rằng anh ấy có hơn 3 năm kinh nghiệm.	0,50	0,37	Một phần
7	Dựa trên ngữ cảnh được cung cấp: 1. Mô hình CrossEncoder ms-marco-MiniLM-L-6-v2 cải thiện độ chính xác cao hơn 15–25% so với chỉ dùng FAISS. 2. Tài liệu được cung cấp không nêu rõ mô hình embedding all-MiniLM-L6-v2 có bao nhiêu chiều. Nó chỉ đề cập rằng mô hình embedding chuyển đổi	0,67	0,52	Một phần

	văn bản thành "vector số chiều cao".			
8	RAG được giới thiệu trong bài báo "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks" của Lewis et al. vào năm 2020. Tài liệu được cung cấp không nêu rõ nhược điểm của RAG. Nó chỉ đề cập rằng RAG "Cân bằng giữa chi phí, độ chính xác và khả năng mở rộng" khi so sánh với các phương pháp khác như Fine-tuning và Prompt stuffing.	1,00	0,62	Đúng
9	Theo quy trình phát triển phần mềm: * SLA cho lỗi Critical là phản hồi trong 1 giờ và xử lý trong 4 giờ. * Thời gian kiểm thử tối thiểu cho dự án trên 3 tháng là 2 tuần.	1,00	0,73	Đúng
10	Tính đến ngày 31/3/2025, tổng nhân sự của công ty TechVN là 312 người, trong đó có 248 kỹ sư phần mềm.	1,00	0,76	Đúng
11	Dựa trên ngữ cảnh được cung cấp: Ứng viên Duong Hai Long không được đề cập là đang làm việc tại công ty nào trong tài liệu. Tài liệu chỉ cung cấp thông tin cá nhân và tóm tắt kinh nghiệm của anh ấy. Báo cáo kết quả kinh doanh quý 1/2025 là của Công ty Cổ phần TechVN.	0,50	0,62	Một phần
12	Dựa trên ngữ cảnh được cung cấp: 1. **Mức lương hiện tại của ứng viên Duong Hai Long:** Tài liệu không đề cập đến ứng viên tên Duong Hai Long, cũng như không cung cấp thông tin về mức lương của bất kỳ cá nhân nào. 2. **Lợi nhuận năm 2026 của TechVN:** Tài liệu có nhắc đến Công ty TechVN và kế hoạch doanh thu quý 2/2025 (47 tỷ đồng), nhưng không có thông tin về lợi nhuận của TechVN cho năm 2026.	—	—	Đúng (từ chối)

Từ kết quả tham khảo có thể rút ra một nhận định: giá trị của hệ thống đến từ sự phối hợp giữa nhiều lớp xử lý thay vì từ riêng một mô hình sinh ngôn ngữ. Nói cách khác, bảng tổng hợp không chỉ phản ánh hiệu năng mà còn minh chứng cho triết lý thiết kế của đề tài, trong đó mỗi thành phần đều được chọn vì đóng góp vào độ tin cậy, khả năng giải thích và trải nghiệm sử dụng cuối cùng.

3.4.2 Đánh giá kết quả

Có thể thấy chất lượng câu trả lời của hệ thống phụ thuộc nhiều vào dạng câu hỏi. Cụ thể, ở nhóm câu hỏi tra cứu dữ kiện, hệ thống đạt điểm trùng khớp rất cao (0.92) do sự kết hợp xuất sắc giữa FAISS và mô hình Gemini 2.5 Flash. Khi chuyển sang nhóm câu hỏi tổng hợp nội dung, điểm trùng khớp giảm nhẹ xuống mức 0.85, chủ yếu do giới hạn giữ lại 3 đoạn văn (top_n=3) của Reranker khiến hệ thống bỏ sót một vài ý nhỏ so với đáp án mẫu biên soạn thủ công. Thử thách lớn nhất nằm ở nhóm câu hỏi liên tài liệu với điểm số giảm còn 0.74 và tỷ lệ Hit Rate chỉ đạt 80%, cho thấy có khoảng 20% trường hợp hệ thống không lấy đủ ngữ cảnh từ nhiều tài liệu khác nhau, dẫn đến việc thiếu hụt thông tin đầu vào cho quá trình sinh câu trả lời. Ngược lại, đối với nhóm câu hỏi không có thông tin, hệ thống ghi nhận điểm số gần như tuyệt đối (0.98), thể hiện khả năng thực thi cực kỳ tốt lệnh từ chối trả lời, qua đó giúp xây dựng lòng tin cho người dùng và ngăn chặn hiệu quả rủi ro phát sinh ảo giác thông tin.

3.4.3 Nhận xét tổng hợp và hướng cải thiện

Có thể nhận định hệ thống đã đạt được mục tiêu cốt lõi của đề tài là xây dựng một trợ lý tài liệu có khả năng hỏi đáp, tóm tắt và trích xuất trên nền tảng LLM kết hợp RAG. Điểm mạnh rõ nhất của giải pháp là kiến trúc mạch lạc, tính minh bạch tương đối cao và khả năng triển khai bằng các thư viện mã nguồn mở phổ biến. Chất lượng phản hồi nhìn chung đáp ứng tốt nhu cầu khai thác tài liệu ở mức tác nghiệp và nghiên cứu cơ bản, đặc biệt khi người dùng đặt câu hỏi cụ thể và tài liệu đã được ingest sạch.

Hệ thống vẫn còn một số giới hạn. Chất lượng đọc PDF phụ thuộc vào khả năng trích xuất văn bản của PyPDF2, nên với tài liệu scan hoặc bố cục quá phức tạp, kết quả có thể chưa tối ưu. Tác vụ tóm tắt và trích xuất hiện chưa áp dụng retrieval cục bộ như hỏi đáp, nên với tài liệu quá dài có thể làm tăng chi phí token. Ngoài ra hệ thống chưa hiển thị trích dẫn nguồn ở mức câu trả lời cuối. Các hướng cải thiện tự nhiên bao gồm bổ sung OCR cho tài liệu scan, áp dụng hierarchical RAG cho tóm tắt dài, gắn nguồn chunk vào từng phản hồi và phát

triển thêm cơ chế đánh giá tự động trên bộ chuẩn tiếng Việt.

Giá trị thời gian chỉ mang tính tham khảo trong môi trường máy trạm nội bộ; chất lượng trả lời được tổng hợp từ đánh giá thủ công trên các bộ câu hỏi đa dạng.

Kết quả thực nghiệm xác nhận rằng hướng tiếp cận kết hợp LLM với RAG, reranking và prompt engineering là phù hợp cho bài toán trợ lý đọc hiểu tài liệu. Hệ thống đạt được mức cân bằng hợp lý giữa độ chính xác, tốc độ và tính khả dụng triển khai. Những hạn chế còn lại chủ yếu thuộc về chất lượng dữ liệu đầu vào, sự phụ thuộc vào dịch vụ LLM bên ngoài và việc chưa khai thác hết các kỹ thuật nâng cao cho tài liệu rất dài. Tuy nhiên, trong phạm vi một đề án thạc sĩ định hướng ứng dụng, kết quả đạt được là có ý nghĩa cả về mặt kỹ thuật lẫn khả năng mở rộng thành sản phẩm hoàn chỉnh hơn.

Thực nghiệm trong tương lai nên xem xét cả yếu tố người dùng cuối. Các tiêu chí như mức độ hài lòng, mức độ tin tưởng, thời gian hoàn thành tác vụ và tần suất phải đọc lại tài liệu gốc sẽ bổ sung một góc nhìn rất thực tiễn cho hệ thống. Nhiều khi một cải tiến kỹ thuật làm tăng vài phần trăm độ chính xác nhưng không cải thiện trải nghiệm thực tế; ngược lại, một thay đổi nhỏ về giao diện hoặc cách trình bày kết quả có thể nâng cao đáng kể hiệu quả sử dụng. Vì vậy, mở rộng thực nghiệm theo hướng lấy người dùng làm trung tâm sẽ là bước đi tự nhiên để hoàn thiện giá trị ứng dụng của đề tài.

Một hướng mở rộng thực nghiệm khác là đánh giá khả năng chịu tải và độ ổn định khi số lượng tài liệu tăng lên đáng kể. Khi corpora mở rộng từ vài chục lên hàng trăm hoặc hàng nghìn tài liệu, các câu hỏi về chiến lược đánh chỉ mục, quản lý bộ nhớ đệm, tốc độ reranking và chính sách chọn candidate_k sẽ trở nên rõ rệt hơn. Đây là cơ hội để kiểm chứng sâu hơn các quyết định thiết kế hiện tại, đồng thời xem xét các giải pháp nâng cao như hybrid search, caching hoặc tiền xử lý theo miền.

Nếu mở rộng thực nghiệm trong giai đoạn tiếp theo, một hướng cần thiết là chuẩn hóa bộ benchmark tiếng Việt cho trợ lý tài liệu. Bộ benchmark này có thể bao gồm tập tài liệu nhiều miền, tập câu hỏi có đáp án chuẩn, tập yêu cầu tóm tắt với bản tham chiếu và tập truy vấn trích xuất có nhãn. Khi đó, hệ thống không chỉ được đánh giá bằng nhận xét thủ công mà còn có thể dùng các chỉ số định lượng nhất quán hơn. Việc xây dựng benchmark như vậy sẽ có ý nghĩa vượt ra ngoài phạm vi riêng của đề tài, vì nó tạo nền tảng chung cho các nghiên cứu tiếp theo về RAG tiếng Việt.

Khả năng ứng dụng thực tế còn đến từ việc hệ thống đã giải quyết khá đầy đủ các nhu cầu vòng đời tài liệu. Người dùng không chỉ hỏi đáp mà còn có thể nạp tài liệu, quản lý kho dữ liệu, xóa tài liệu, xem trước nội dung, tóm tắt nhanh hoặc trích xuất dữ kiện có cấu trúc. Nói cách khác, hệ thống không coi tài liệu chỉ là đầu vào cho một truy vấn đơn lẻ, mà coi tài liệu là một đối tượng dữ liệu có vòng đời riêng. Đây là một đặc điểm quan trọng nếu muốn triển khai thật, bởi các tổ chức luôn cần khả năng cập nhật, chỉnh sửa và kiểm soát tài liệu.

Kết quả thực nghiệm cho phép nhận định rằng hệ thống phù hợp với nhiều bối cảnh ứng dụng thực tế khác nhau. Trong môi trường học thuật, công cụ có thể hỗ trợ học viên đọc nhanh bài báo, tóm tắt luận điểm chính, truy xuất khái niệm và đối chiếu các mốc thời gian hoặc kết quả thực nghiệm quan trọng. Trong môi trường doanh nghiệp, công cụ có thể đóng vai trò giao diện hỏi đáp trên báo cáo nội bộ, biên bản họp, quy trình vận hành hoặc văn bản pháp lý. Đối với cơ quan quản lý, giá trị tiềm năng nằm ở khả năng giảm thời gian đọc thủ công và chuẩn hóa cách khai thác thông tin từ kho tài liệu ngày càng lớn. Như vậy, dù đề tài được phát triển ở quy mô nghiên cứu, logic sản phẩm của nó có thể chuyển tiếp tương đối thuận lợi sang bối cảnh sử dụng thực.

3.5 Khả năng ứng dụng thực tế và định hướng mở rộng thực nghiệm

Khả năng ứng dụng thực tế của hệ thống thể hiện rõ nhất ở những bối cảnh mà người dùng phải làm việc với khối lượng tài liệu lớn nhưng thời gian đọc hiểu lại hạn chế. Với tài liệu học thuật, hệ thống có thể hỗ trợ người học rà soát luận điểm chính, truy tìm khái niệm then chốt và đối chiếu nhanh nhiều nguồn trong cùng một chủ đề. Với môi trường doanh nghiệp, các tác vụ như tra cứu hợp đồng, tổng hợp biên bản họp hay truy xuất quy trình nội bộ đều có thể được rút ngắn đáng kể khi người dùng chuyển từ thao tác đọc tuyến tính sang đặt câu hỏi trực tiếp bằng ngôn ngữ tự nhiên. Chính tính chất đa dụng này cho thấy hệ thống không bị bó hẹp trong một miền ứng dụng đơn lẻ.

Tuy nhiên ứng dụng thực tế chỉ bền vững khi đi cùng với một chiến lược thực nghiệm mở rộng đủ nghiêm ngặt. Trong các vòng đánh giá tiếp theo hệ thống nên được thử trên những tập tài liệu có cấu trúc khác nhau, chẳng hạn giáo trình, báo cáo quản trị, văn bản pháp quy và hồ sơ kỹ thuật, nhằm kiểm tra mức độ ổn định của cơ chế chunking và truy xuất ngữ nghĩa. Bên cạnh thước đo chính xác nội dung, cần bổ sung các quan sát về khả năng chấp nhận của người dùng như mức độ dễ hiểu, thời gian hoàn thành tác vụ và mức tin cậy khi đối

chiếu với tài liệu gốc. Việc mở rộng thực nghiệm theo hướng này sẽ giúp kết nối chặt chẽ hơn giữa kết quả trong phòng thí nghiệm và giá trị triển khai ngoài thực tế.

Một định hướng quan trọng khác là tổ chức thực nghiệm theo kiểu ablation có kiểm soát, tức thay đổi từng thành phần của pipeline để xác định chính xác mức đóng góp của nó. Chẳng hạn, có thể so sánh biến thể dùng `all-MiniLM-L6-v2` với các mô hình embedding đa ngôn ngữ khác, hoặc đánh giá tác động của việc tăng giảm `chunk_size`, `overlap`, `top_k` và số ví dụ few-shot đối với từng loại nhiệm vụ. Khi kết hợp các kết quả này với nhật ký thời gian xử lý đã có sẵn trong metadata, báo cáo đánh giá sẽ phản ánh đầy đủ hơn quan hệ đánh đổi giữa chất lượng phản hồi và chi phí vận hành.

III. KẾT LUẬN

1. Kết quả và đóng góp của đề án

Đề tài đã triển khai thành công một hệ thống chatbot trợ lý đọc hiểu tài liệu văn bản thông qua ngôn ngữ tự nhiên. Hệ thống sử dụng mô hình ngôn ngữ lớn LLM kết hợp với RAG nhằm cải thiện khả năng hiểu và phản hồi thông tin từ tài liệu. Có thể khẳng định đề án đã hoàn thành các mục tiêu cốt lõi là nghiên cứu nền tảng lý thuyết, lựa chọn công nghệ, thiết kế kiến trúc module hóa và triển khai được một hệ thống vận hành thực tế. Kết quả của đề tài thể hiện qua những đóng góp nổi bật sau:

Đóng góp về mặt giải pháp kỹ thuật:

- Xây dựng pipeline hoàn chỉnh và minh bạch: Hệ thống hiện thực một pipeline gồm 4 giai đoạn gồm tiền xử lý dữ liệu, làm giàu prompt, gọi mô hình LLM và hậu xử lý. Việc phân tách rõ ràng này giúp dòng xử lý vừa minh bạch về mặt học thuật vừa thuận tiện khi triển khai thực tế.

- Hybrid Search 4 tầng chuyên sâu: Ở tác vụ hỏi đáp documentQA, đề tài thiết lập một quy trình truy xuất khắt khe nhằm cải thiện chất lượng ngữ cảnh và loại bỏ ảo giác, bao gồm: FAISS truy xuất theo ngữ nghĩa, BM25 truy xuất theo từ khóa, hợp nhất tập ứng viên bằng thuật toán RRF, xếp hạng lại bằng mô hình CrossEncoder và dùng thuật toán MMR để chọn ra 5 đoạn ngữ cảnh sát với câu hỏi và đa dạng nhất.

- Prompt Engineering: Đề tài xây dựng một lớp prompt tương đối hoàn chỉnh với 24 ví dụ few-shot phân bố đều cho 3 tác vụ, kết hợp khối Chain-of-Thought gồm 4 bước suy luận nghiêm ngặt cho từng loại yêu cầu.

Đóng góp về kiến trúc phần mềm và tối ưu hóa hệ thống:

- Xử lý đa định dạng và thuật toán khử trùng lặp vector: Ở tầng dữ liệu hệ thống mở rộng phạm vi ingest ra đa định dạng như PDF, DOCX, DOC, TXT. Đặc biệt khâu phân đoạn được bổ sung bước khử trùng lặp chunk dựa trên độ tương đồng Jaccard giữa các n-gram trước khi vector hóa. Cải tiến này giúp loại bỏ các đoạn gần trùng, giảm số vector dư thừa và thu gọn đáng kể chỉ mục FAISS.

- Kiến trúc module và tối ưu tài nguyên: Hệ thống chia thành 5 subpackage độc lập kết nối qua 12 endpoint REST và giao diện web. Đặc biệt tối ưu hóa thông qua cơ chế cache ở mức tài nguyên (bộ nhớ đệm FAISS).

- Giá trị học thuật theo hướng ứng dụng: Thông qua việc hiện thực trọn vẹn từ lý thuyết đến source code, đề tài làm rõ mối quan hệ giữa các khái niệm đang được sử dụng rộng rãi như NLP, LLM, RAG, reranking, few-shot, CoT, tạo điều kiện thuận lợi cho người đọc muốn tiếp tục nghiên cứu hoặc tái sử dụng hệ thống.

2. Đánh giá đóng góp ứng dụng của đề án

Hệ thống được đưa vào ứng dụng thực tế như tra cứu thông tin trong tài liệu, tóm tắt nội dung hoặc trích xuất các trường dữ kiện. Hệ thống đạt độ chính xác cao và biết từ chối đúng khi thông tin không có trong tài liệu, qua đó khẳng định khả năng ứng dụng thực tế của hệ thống.

Hệ thống đặc biệt phù hợp với các nhu cầu như tra cứu giáo trình, hỏi đáp trên tài liệu môn học hoặc tóm tắt nhanh bài báo khoa học để hỗ trợ đọc hiểu. Với khả năng truy xuất theo ngữ cảnh và tóm lược nội dung dài, công cụ có thể giúp sinh viên và học viên sau đại học rút ngắn đáng kể thời gian tìm ý chính, đối chiếu khái niệm và tổng hợp tài liệu tham khảo.

Trong doanh nghiệp, ứng dụng có thể phát huy ở các tác vụ tra cứu thông tin trong hợp đồng, báo cáo nội bộ, quy trình vận hành hoặc biên bản họp. Chức năng trích xuất có thể hỗ trợ lấy ra các thực thể có cấu trúc như ngày tháng, tên dự án, khoản mục chi phí hoặc nghĩa vụ thực hiện, từ đó giảm khối lượng đọc thủ công đối với các tài liệu dài và lặp lại.

Đối với lĩnh vực pháp lý và hành chính, hệ thống có tiềm năng hỗ trợ tra cứu văn bản quy phạm pháp luật, tóm tắt các điều khoản quan trọng hoặc xác định nhanh các quy định liên quan đến một chủ đề cụ thể. Mặc dù chưa thay thế được chuyên gia pháp lý, công cụ có thể đóng vai trò như một lớp hỗ trợ ban đầu nhằm thu hẹp phạm vi tìm kiếm và tăng tốc quá trình rà soát tài liệu.

Kết quả hệ thống có thể hỗ trợ người dùng nhanh chóng tìm ra thông tin cần thiết từ các tài liệu lớn, giảm đáng kể thời gian đọc hiểu và phân tích thủ công. Các câu trả lời do hệ thống sinh ra có độ chính xác cao, dễ hiểu và bám sát ngữ cảnh tài liệu gốc. Điều này cho thấy việc ứng dụng mô hình ngôn ngữ lớn kết hợp với truy xuất tri thức không chỉ khả thi về mặt kỹ thuật, mà còn thực sự mang lại giá trị trong thực tiễn ứng dụng.

Đề án khẳng định một định hướng quan trọng trong phát triển ứng dụng AI hiện nay: giá trị thực tiễn không chỉ đến từ việc lựa chọn một mô hình ngôn ngữ mạnh, mà đến từ khả năng tổ chức dữ liệu, kiểm soát ngữ cảnh và thiết kế

pipeline phù hợp với mục tiêu nghiệp vụ. Hệ thống AI Document Assistant là minh chứng cho cách tiếp cận đó. Thay vì xây dựng một chatbot chung chung, đề tài tập trung vào một bài toán cụ thể là đọc hiểu tài liệu và từng bước giải quyết các nhu cầu thực tế như ingest đa định dạng, quản lý kho tài liệu, hỏi đáp theo ngữ cảnh, tóm tắt nội dung và trích xuất thông tin có cấu trúc.

Hệ thống đã cho thấy khả năng vận hành với một kiến trúc tương đối gọn nhưng vẫn giữ được tính mở rộng. Các module ingestion, chunking, embedding, pipeline và frontend được tách bạch, tạo tiền đề cho việc thay thế từng thành phần công nghệ trong tương lai. Nếu tiếp tục phát triển có thể bổ sung các năng lực như hiển thị nguồn trích dẫn theo chunk, hỗ trợ OCR cho tài liệu scan, đánh giá tự động trên bộ chuẩn tiếng Việt, tích hợp cơ chế phân quyền truy cập tài liệu hoặc triển khai trên môi trường doanh nghiệp với kho dữ liệu lớn hơn. Những hướng đi này cho thấy đề án không dừng ở mức mô hình thử nghiệm mà có tiềm năng phát triển tiếp thành nền tảng hỗ trợ tri thức số.

Đề án có giá trị học thuật theo hướng ứng dụng. Thông qua việc hiện thực hóa trọn vẹn từ lý thuyết đến mã nguồn, đề tài giúp làm rõ mối quan hệ giữa các khái niệm đang được sử dụng rộng rãi trong cộng đồng AI như NLP, LLM, RAG, reranking, few-shot và Chain-of-Thought. Việc đối chiếu trực tiếp giữa cấu trúc chương báo cáo với cấu trúc mô-đun trong mã nguồn cũng tạo điều kiện thuận lợi cho người đọc muốn tiếp tục nghiên cứu hoặc tái sử dụng hệ thống vào bài toán gần tương tự.

Có thể kết luận đề án đã hoàn thành tốt mục tiêu đặt ra: xây dựng được một hệ thống trợ lý đọc hiểu tài liệu thông minh có kiến trúc rõ ràng, có khả năng vận hành, có cơ sở lý thuyết đầy đủ và có kết quả thực nghiệm thuyết phục ở mức ứng dụng. Những kết quả này không chỉ có ý nghĩa trong phạm vi môn học hay luận văn mà còn gợi mở một hướng phát triển khả thi cho các công cụ khai thác tài liệu tiếng Việt trong bối hiện nay.

3. Nhận xét đề án

Đề tài không tập trung vào việc đề xuất phương pháp mới mà hướng đến việc ứng dụng hiệu quả các kỹ thuật và mô hình đã được nghiên cứu rộng rãi. Điểm mạnh của hệ thống là kiến trúc module hóa rõ ràng gồm ingestion, chunking, embedding, pipeline, conversion; không chỉ dễ đọc mà còn cực kỳ thuận lợi khi cần thay thế từng thành phần mà không phải viết lại toàn bộ ứng dụng.

Tuy nhiên đề tài vẫn còn những điểm chưa hoàn thiện: chất lượng suy luận

hiện vẫn phụ thuộc hoàn toàn vào dịch vụ Gemini API bên ngoài, đầu ra hỏi đáp chưa hiển thị trích dẫn chunk theo dạng citation tường minh, và hệ thống chưa tích hợp OCR nên chưa xử lý tốt tài liệu scan hoặc tài liệu ảnh.

4. Hướng phát triển tiếp theo

Hướng phát triển tiếp theo trước hết nên tập trung vào việc nâng chất lượng của tầng ingestion, bởi đây là điểm khởi đầu quyết định độ đầy đủ của toàn bộ pipeline RAG. Đối với tài liệu quét hoặc tài liệu có bố cục nhiều cột, việc bổ sung OCR, nhận diện cấu trúc trang và trích xuất bảng biểu sẽ giúp hệ thống tiếp cận được nhiều nguồn dữ liệu hơn thay vì chỉ giới hạn ở văn bản có sẵn lớp text. Khi chất lượng dữ liệu đầu vào được cải thiện, các bước chunking, embedding và retrieval phía sau cũng sẽ ổn định hơn. Đây là hướng nâng cấp có tác động lan tỏa tới hầu hết thành phần của hệ thống.

Một hướng phát triển khác là tăng cường khả năng giải thích và khả năng kiểm chứng của phản hồi. Hệ thống hiện đã có cấu trúc đủ để biết những chunk nào được truy xuất, song chưa hiển thị rõ cho người dùng mối liên hệ giữa câu trả lời và nguồn trích dẫn. Việc bổ sung chức năng hiển thị đoạn ngữ cảnh, đánh dấu vị trí trong tài liệu, hoặc thậm chí chấm điểm độ tin cậy cho từng phản hồi sẽ làm tăng đáng kể giá trị sử dụng trong môi trường học thuật và chuyên môn. Khi đó, hệ thống không chỉ trả lời mà còn hỗ trợ người dùng tự đánh giá độ chắc chắn của câu trả lời.

Nghiên cứu tương lai nên mở rộng theo hướng chuẩn hóa đánh giá trên bộ dữ liệu tiếng Việt có nhãn và thử nghiệm thêm các biến thể kiến trúc. Những phương án như tóm tắt phân cấp cho ngữ cảnh dài (long-context summarization), triển khai mô hình ngôn ngữ cục bộ qua Ollama, hoặc bổ sung lớp bộ nhớ dài hạn cho hội thoại đa lượt đều là các hướng đáng khảo sát. Sự kết hợp giữa benchmark định lượng và khảo sát trải nghiệm người dùng sẽ giúp xác định rõ đâu là cải tiến mang lại giá trị thực tiễn cao nhất. Trên cơ sở đó, hệ thống AI Document Assistant có thể tiến gần hơn tới một nền tảng trợ lý tài liệu hoàn chỉnh, đáng tin cậy và có khả năng phục vụ trong nhiều bối cảnh ứng dụng khác nhau.

IV. DANH MỤC CÁC TÀI LIỆU THAM KHẢO

[1] Lewis, P., Perez, E., Piktus, A., et al. (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. Advances in Neural Information Processing Systems.

[2] Jurafsky, D., và Martin, J. H. (2023). Speech and Language Processing (bản thảo lần xuất bản thứ 3). Đại học Stanford.

[3] Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. Proceedings of NAACL-HLT.

[4] Johnson, J., Douze, M., và Jégou, H. (2019). Billion-Scale Similarity Search with GPUs. Tạp chí IEEE Transactions on Big Data.

[5] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention Is All You Need. Advances in Neural Information Processing Systems.

[6] Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. Proceedings of EMNLP-IJCNLP.

[7] LangChain. (2025). LangChain Python Documentation. Truy cập tại: https://python.langchain.com/

[8] Brown, T. B., Mann, B., Ryder, N., et al. (2020). Language Models are Few-Shot Learners. Advances in Neural Information Processing Systems.

[9] Wei, J., Wang, X., Schuurmans, D., và cộng sự (2022). Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. Kỷ yếu Advances in Neural Information Processing Systems.

[10] Karpukhin, V., Oguz, B., Min, S., et al. (2020). Dense Passage Retrieval for Open-Domain Question Answering. Proceedings of EMNLP.

[11] Touvron, H., Martin, L., Stone, K., cùng các tác giả khác (2023). Llama 2: Open Foundation and Fine-Tuned Chat Models. Meta AI.

[12] Nogueira, R., & Cho, K. (2019). Passage Re-ranking with BERT. arXiv preprint arXiv:1901.04085.

[13] R. Jay, Generative AI Apps with LangChain and Python, Apress, 2024.

[14] D. Polzer, RAG with Python Cookbook: Practical Recipes from Data Preprocessing to LLM Agents, O'Reilly Media, 2025.

[15] D. Rothman, RAG-Driven Generative AI: Build Custom RetrievalAugmented Generation Pipelines with LlamaIndex, Deep Lake, and Pinecone, Packt Publishing, 2024

BỘ KHOA HỌC VÀ CÔNG NGHỆ
HỌC VIỆN CÔNG NGHỆ
BƯU CHÍNH VIỄN THÔNG

CỘNG HOÀ XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập - Tự do - Hạnh phúc

BIÊN BẢN
HỌP HỘI ĐỒNG CHẤM ĐỀ ÁN TỐT NGHIỆP THẠC SĨ

Căn cứ quyết định số Quyết định số 1949/QĐ-HV ngày 07 tháng 07 năm 2026 của Giám đốc Học viện Công nghệ Bưu chính Viễn thông về việc thành lập Hội đồng chấm đề án tốt nghiệp thạc sĩ. Hội đồng đã họp vào hồi 10 giờ 15 phút, ngày 22 tháng 07 năm 2026 tại Học viện Công nghệ Bưu chính Viễn thông để chấm đề án tốt nghiệp thạc sĩ cho:

Học viên: **Nguyễn Quang Huy**

Tên đề án tốt nghiệp: **Nghiên cứu xây dựng chatbot trợ lý đọc hiểu tài liệu**

Chuyên ngành: **Hệ thống thông tin** Mã số: **8480104**

Các thành viên của Hội đồng chấm đề án tốt nghiệp có mặt: 05 / 05

TT	HỌ VÀ TÊN	TRÁCH NHIỆM TRONG HĐ	GHI CHÚ
1	**PGS.TS. Nguyễn Trọng Khánh**	Chủ tịch	
2	**TS. Đặng Ngọc Hùng**	Thư ký	
3	**TS. Vũ Văn Thỏa**	Phản biện 1	
4	**TS. Nguyễn Văn Vinh**	Phản biện 2	
5	**TS. Trịnh Bảo Ngọc**	Uỷ viên	

Các nội dung thực hiện:

1. Chủ tịch Hội đồng điều khiển buổi họp. Công bố quyết định của Giám đốc Học viện Công nghệ Bưu chính Viễn thông về việc thành lập Hội đồng chấm đề án tốt nghiệp thạc sĩ.
2. Người hướng dẫn khoa học hoặc thư ký đọc lý lịch khoa học và các điều kiện bảo vệ đề án tốt nghiệp của học viên. (có bản lý lịch khoa học và kết quả các môn học cao học của học viên kèm theo).
3. Học viên trình bày tóm tắt đề án tốt nghiệp.
4. Phản biện 1 đọc nhận xét (có văn bản kèm theo)
5. Phản biện 2 đọc nhận xét (có văn bản kèm theo)
6. Các câu hỏi của thành viên Hội đồng:

1, Giải thích rõ hơn về kiến trúc tổng thể hệ thống?
2, Số lượng tài liệu và nguồn tài liệu thử nghiệm?
3, Làm rõ lý thuyết Embedding được áp dụng
4, Tại sao không đánh giá thành phần RAG trong pipeline?

7. Trả lời của học viên:

1. Học viên có giải thích về kiến trúc 02 thành phần FE và BE
2. Đề nghị sử dụng 03 nhóm tài liệu: thư viện, pháp lý và nghiệp vụ. Tổng 10 tài liệu.
3. Học viên có trả lời về các lý thuyết Embbeding được sử dụng.
4. Đề nghị viết thêm phần giới thiệu về RAG.

8. Thư ký đọc nhận xét về quá trình thực hiện đề án tốt nghiệp của học viên *(có văn bản kèm theo).*

9. Hội đồng họp riêng:

- Bầu Ban kiểm phiếu:
  1. Trưởng Ban kiểm phiếu: PGS. TS. Nguyễn Trọng Chiến
  2. Uỷ viên Ban kiểm phiếu: TS. Vũ Văn Thoả
  3. Uỷ viên Ban kiểm phiếu: TS. Đặng Ngọc Hưng
- Hội đồng chấm đề án tốt nghiệp bằng bỏ phiếu kín.
- Ban kiểm phiếu làm việc:
- Trưởng Ban kiểm phiếu báo cáo kết quả kiểm phiếu *(có Biên bản họp Ban kiểm phiếu kèm theo)*
- **Điểm trung bình của đề án tốt nghiệp:** 7,0

Kết luận:

1. Các nội dung cần chỉnh sửa, hoàn thiện sau bảo vệ đề án tốt nghiệp:
   - Rà soát nghiên cứu văn phong, câu, chữ, ngữ không có tính học thuật. (theo góp ý của phản biện)
   - Làm rõ và mô tả bổ sung tài liệu phù nghiên trong đề án
   - Thêm chỉ bố cục để viết cho phù hợp nội dung, tên đề án.

2. Đề nghị Học viện công nhận (hoặc không) và cấp bằng (hoặc không) thạc sĩ cho học viên: Đề nghị công nhận

3. Đề án tốt nghiệp có thể phát triển thành đề tài nghiên cứu cho NCS.....Không.....

Buổi làm việc kết thúc vào..... cùng ngày.

Chủ tịch	**Thư ký**
Nguyễn Trọng Chiến	Đặng Ngọc Hưng

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập – Tự do – Hạnh phúc

BẢN NHẬN XÉT ĐỀ ÁN TỐT NGHIỆP THẠC SĨ

(Dùng cho người phản biện)

Tên đề tài đề án tốt nghiệp: NGHIÊN CỨU XÂY DỰNG CHATBOT TRỢ LÝ ĐỌC HIỂU TÀI LIỆU

Chuyên ngành: Hệ thống thông tin

Mã chuyên ngành: 8.48.01.04

Họ và tên học viên: Nguyễn Quang Huy .....

Họ và tên người nhận xét: Nguyễn Văn Vinh

Học hàm, học vị: Tiến sỹ .....

Chuyên ngành: Khoa học máy tính .....

Cơ quan công tác: Trường Đại học Công Nghệ, ĐHQG Hà Nội .....

Số điện thoại:.....E-mail:.....

NỘI DUNG NHẬN XÉT

I/ Cơ sở khoa học và thực tiễn, tính cấp thiết của đề tài:
Hiện nay việc đọc hiểu tài liệu với khối lượng lớn là vấn đề quan trọng trong thời đại công nghệ số. Chatbot ngày càng được ứng dụng rộng rãi trong nhiều lĩnh vực thực tế. Vì vậy việc nghiên cứu và xây dựng ChatBot trợ giúp đọc hiểu tài liệu là bài toán có ý nghĩa về mặt khoa học cũng như thực tế.
II/ Nội dung của đề án tốt nghiệp, các kết quả đã đạt được:

- Tìm hiểu tổng quan về bài toán Chatbot
- Nghiên cứu và xây dựng Chatbot dựa vào mô hình ngôn ngữ lớn và công nghệ RAG (Retrieve Augumentation Generation)
- Thử nghiệm trên các bộ dữ liệu học thuật, quản trị và nghiệp vụ qui chế và cho kết quả khả quan.

III/ Những vấn đề cần giải thích thêm:

- Bố cục của đề án chưa hợp lý. Nội dung tổng quan và kiến thức nền tảng nên tập trung vào chương 1. Bỏ II. NỘI DUNG (trang 15), Tên chương 1: Cần thay đổi tên cho phù hợp với nội dung của bài toán
- Đề án viết chưa được tốt, các chương khá lộn xộn, chưa rõ vấn đề và phạm vi cần giải quyết.

- Chương 1 trình bày khá lan man. Nội dung chương 1 nên tập trung vào bài toán Chatbot và các kỹ thuật để xây dựng Chatbot hiệu quả.
- Chương 2 nên tập trung vào nghiên cứu và thiết kế xây dựng Chatbot bao gồm phân tích hệ thống ChatBot và các modul ứng dụng LLM và RAG trong đó. Hình vẽ cần rõ ràng (nên vẽ lại).
- Chương 3 thực nghiệm và đánh giá khá sơ sài? Chưa rõ là thử nghiệm với bao nhiêu tài liệu (bao nhiêu tokens?), các loại tài liệu như thế nào? Các công cụ sử dụng, các siêu tham số như thế nào, và tập đánh giá với 12 câu hỏi thì quá nhỏ? Không có kịch bản đánh giá? Phương pháp đánh giá (ví dụ độ tương đồng đánh giá như thế nào)? Có đánh giá phần RAG riêng không?
- Tài liệu tham khảo cần được tham chiếu trong nội dung của đề án.
- Một số lỗi chính tả và cách tham chiếu tài liệu tham khảo cần được chỉnh sửa.

IV/ Kết luận:

Đề án đáp ứng cơ bản đề án bậc thạc sỹ ngành Hệ thống thông tin sau khi chỉnh sửa theo các góp ý của các thành viên hội đồng.

Đồng ý cho phép học viên bảo vệ đề án tốt nghiệp.

Ngày...21...tháng...7.....năm .2026

NGƯỜI NHẬN XÉT

Nguyễn Văn Vinh

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập – Tự do – Hạnh phúc

BẢN NHẬN XÉT ĐỀ ÁN TỐT NGHIỆP THẠC SĨ

(Dùng cho người phản biện)

Tên đề tài đề án tốt nghiệp: Nghiên cứu xây dựng chatbot trợ lý đọc hiểu tài liệu

Chuyên ngành: Hệ thống thông tin
Mã chuyên ngành: 8.48.01.04
Họ và tên học viên: Nguyễn Quang Huy
Họ và tên người nhận xét: Trịnh Bảo Ngọc
Học hàm, học vị: Tiến sĩ
Chuyên ngành: Kỹ thuật phần mềm
Cơ quan công tác: Trường đại học Hà Nội
Số điện thoại: 0987 355 799 E-mail: ngoctb@hanu.edu.vn

NỘI DUNG NHẬN XÉT

I/ Cơ sở khoa học và thực tiễn, tính cấp thiết của đề tài:

Cơ sở khoa học: Đề tài được xây dựng trên nền tảng khoa học vững chắc của lĩnh vực xử lý ngôn ngữ tự nhiên (NLP), mô hình ngôn ngữ lớn (LLM) và kiến trúc Retrieval-Augmented Generation (RAG). Tác giả đã phân tích cơ chế hoạt động của Transformer, embedding vector, cơ sở dữ liệu FAISS và các kỹ thuật truy xuất như BM25, CrossEncoder, RRF. Việc kết hợp lý thuyết từ các nghiên cứu quốc tế và ứng dụng thực tế của các framework như LangChain, LlamaIndex giúp đề tài có cơ sở khoa học rõ ràng, đảm bảo tính hệ thống và phù hợp với xu hướng phát triển AI hiện nay.

Cơ sở thực tiễn: Đề tài xuất phát từ nhu cầu thực tế của người dùng trong việc đọc hiểu, tóm tắt và tra cứu tài liệu văn bản như PDF, DOCX, TXT. Các hệ thống hiện có như ChatGPT Plus, Perplexity AI đã chứng minh tính khả thi, nhưng tại Việt Nam ứng dụng RAG cho trợ lý tài liệu vẫn còn hạn chế. Tác giả đã xây dựng hệ thống chatbot với pipeline bốn giai đoạn, tích hợp FastAPI, FAISS, SentenceTransformers và Gemini, cho phép người dùng tải tài liệu, đặt câu hỏi và nhận phản hồi chính xác. Điều này cho thấy đề tài có giá trị ứng dụng thực tiễn cao, đáp ứng nhu cầu học tập và nghiên cứu.

Tính cấp thiết: Trong bối cảnh dữ liệu văn bản ngày càng nhiều và phức tạp, việc xây dựng hệ thống trợ lý đọc hiểu tài liệu là cấp thiết. Người dùng cá nhân, sinh viên, nhà nghiên cứu và doanh nghiệp đều cần công cụ hỗ trợ tóm tắt, trích xuất và phân tích nhanh chóng. Các mô hình LLM mạnh mẽ như GPT-4, Gemini kết hợp RAG mở ra khả năng ứng dụng rộng rãi. Tại Việt Nam, lĩnh vực này còn mới mẻ, ít công trình

nghiên cứu bài bản. Do đó, đề tài không chỉ có ý nghĩa học thuật mà còn mang tính cấp thiết trong việc tạo ra sản phẩm AI hỗ trợ người dùng hiệu quả.

II/ Nội dung của đề án tốt nghiệp, các kết quả đã đạt được:

Kết cấu đề án:

Đề án gồm ba phần chính: mở đầu với lý do, mục tiêu và phạm vi nghiên cứu; nội dung gồm cơ sở lý thuyết, thiết kế kiến trúc, thực nghiệm và đánh giá; kết luận nêu kết quả, đóng góp, hạn chế và hướng phát triển. Cấu trúc rõ ràng, logic và khoa học được trình bày trong tổng cộng 57 trang, với số trang như thế này là khá ngắn và nên tập trung vào phần mình làm, tuy nhiên phần cơ sở lý thuyết cũng khá dài so với tổng số khá ít trang.

Tóm tắt các kết quả đạt được:

Đề tài đã đạt được nhiều kết quả quan trọng. Về lý luận, tác giả đã hệ thống hóa kiến thức nền tảng về NLP, LLM và RAG, phân tích các công trình liên quan và rút ra định hướng thiết kế cho hệ thống. Về thực tiễn, tác giả đã xây dựng một hệ thống chatbot trợ lý tài liệu với pipeline bốn giai đoạn: ingestion, chunking, embedding và pipeline xử lý. Hệ thống sử dụng mô hình nhúng all-MiniLM-L6-v2, FAISS IndexFlatL2 để lưu trữ vector, CrossEncoder để rerank và Gemini cho sinh ngôn ngữ.

Các chức năng chính gồm hỏi đáp (documentQA), tóm tắt (documentSummary) và trích xuất thông tin (documentExtraction). Hệ thống hỗ trợ nhiều định dạng tài liệu phổ biến như PDF, DOCX, TXT, cho phép người dùng tải lên và nhận phản hồi chính xác. Các kịch bản kiểm thử cho thấy hệ thống hoạt động ổn định, trả lời đúng ngữ cảnh, có khả năng ứng dụng thực tế. Đề tài cũng đánh giá hiệu suất theo độ chính xác và trải nghiệm người dùng, chứng minh tính khả thi và giá trị ứng dụng cao.

III/ Những vấn đề cần giải thích thêm:

Về hình thức trình bày:

Đề án trình bày rõ ràng, có mục lục chi tiết, danh mục thuật ngữ và hình vẽ đầy đủ, giúp người đọc dễ theo dõi và nắm bắt nội dung.

Một số lỗi định dạng xuất hiện trong tài liệu. Trang iii, mục lục có lỗi căn chỉnh khi số trang không đồng bộ, ví dụ "CHƯƠNG 2: CƠ SỞ LÝ THUYẾT..." bị lệch dòng. Trang v, bảng "Danh mục các thuật ngữ" có lỗi xuống dòng ở mục "IndexFlatL2" khiến nội dung bị chia tách không hợp lý. Trang vii, danh sách hình vẽ có lỗi căn chỉnh khi số trang không thẳng hàng, gây khó đọc.

Trang 13, phần tiêu đề "Khảo sát các hệ thống ứng dụng LLM..." xuất hiện trong header nhưng không đồng bộ với nội dung chính. Ngoài ra, một số bảng có phần mô tả dài gây mất cân đối giữa các cột, ví dụ bảng thuật ngữ ở trang v. Các lỗi này tuy không

ảnh hưởng lớn đến nội dung nhưng làm giảm tính chuyên nghiệp của hình thức trình bày.

Về nội dung:

Một số vấn đề về kiến thức và logic cần lưu ý. Trang 8, phần tổng quan nghiên cứu quốc tế liệt kê nhiều công trình nhưng chưa phân tích sâu sự khác biệt giữa các framework như LangChain và LlamaIndex, khiến người đọc khó thấy rõ ưu nhược điểm.
Trang 9, phần mục tiêu cụ thể mô tả pipeline nhưng chưa giải thích rõ lý do chọn tham số chunk_size=600 và overlap=150, gây thiếu cơ sở khoa học.
Trang 11, phần phạm vi nghiên cứu chưa đề cập đến khả năng xử lý tài liệu đa ngôn ngữ, trong khi đây là nhu cầu thực tế.
Trang 12, phần phương pháp nghiên cứu chưa phân tích chi tiết cách đánh giá hiệu suất, chỉ dừng ở mức khái quát.
Trang 26, hình 2.1 minh họa chunking nhưng chưa giải thích rõ cách duy trì ngữ nghĩa khi cắt đoạn, gây thiếu trực quan.
Trang 39, phần bộ dữ liệu kiểm thử chưa nêu rõ nguồn dữ liệu, khiến độ tin cậy chưa cao. Trang 43, kết quả đánh giá mới dừng ở mức mô tả, chưa có so sánh định lượng với các hệ thống khác.
Trang 49, phần khả năng ứng dụng thực tế chưa phân tích chi phí triển khai và yêu cầu hạ tầng, yếu tố quan trọng trong thực tế.
Cuối cùng, phần kết luận ở trang 51 chưa đưa ra định hướng nghiên cứu tiếp theo cụ thể, khiến đề tài thiếu tính mở rộng.

Câu hỏi:

1. Vì sao hệ thống chọn chunk_size=600 và overlap=150 cho quá trình chunking?
2. Cơ chế FAISS IndexFlatL2 hoạt động thế nào trong truy xuất vector?
3. CrossEncoder được tích hợp vào pipeline để cải thiện bước nào?

IV/ Kết luận:
Đồng ý.

Ngày 21 tháng 07 năm 2026
NGƯỜI NHẬN XÉT

TS Trịnh Bảo Ngọc

BỘ THÔNG TIN VÀ TRUYỀN THÔNG
HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập - Tự do - Hạnh phúc

BẢN NHẬN XÉT ĐỀ ÁN TỐT NGHIỆP THẠC SĨ

Tên đề tài đề án:	Nghiên cứu xây dựng Chatbot trợ lý đọc hiểu tài liệu
Chuyên ngành:	Hệ thống thông tin **Mã số:** 8.48.01.04
Tên học viên:	Nguyễn Quang Huy
Họ tên người nhận xét:	Vũ Văn Thỏa
Học hàm, học vị:	Tiến sĩ **Chuyên ngành:** Toán học tính toán
Cơ quan công tác:	Học viện Công nghệ Bưu chính Viễn thông

NỘI DUNG NHẬN XÉT

I/ Cơ sở khoa học và thực tiễn, tính cấp thiết của đề tài

Hiện nay, tại Việt Nam, nhu cầu khai thác, đọc hiểu và xử lý khối lượng lớn tài liệu số như văn bản hành chính, báo cáo, hợp đồng, tài liệu nghiên cứu và tài liệu học tập ngày càng gia tăng. Tuy nhiên, việc tìm kiếm, tổng hợp và khai thác thông tin từ các tài liệu này vẫn thường được thực hiện thủ công, tốn nhiều thời gian và phụ thuộc vào kỹ năng của người sử dụng. Sự phát triển mạnh mẽ của các mô hình ngôn ngữ lớn (Large Language Models - LLMs), kỹ thuật RAG (Retrieval-Augmented Generation), Embedding và Vector Database đã tạo nền tảng khoa học để xây dựng các chatbot có khả năng đọc hiểu, truy xuất tri thức và trả lời câu hỏi dựa trên nội dung tài liệu với độ chính xác và tính linh hoạt cao.

Đề án với đề tài "Nghiên cứu xây dựng Chatbot trợ lý đọc hiểu tài liệu" của học viên Nguyễn Quang Huy hướng tới mục tiêu xây dựng một hệ thống chatbot trợ lý tài liệu thông minh có khả năng tiếp nhận tài liệu văn bản từ người dùng, hiểu nội dung của tài liệu và phản hồi các yêu cầu hỏi đáp, tóm tắt, trích xuất thông tin. Vì vậy, đề tài có tính cấp thiết, dựa trên cơ sở khoa học của các công nghệ AI và mang ý nghĩa thực tiễn cao, góp phần nâng cao hiệu quả khai thác tài liệu số và hỗ trợ người dùng trong quá trình tìm kiếm, đọc hiểu và trích xuất thông tin.

II/ Về nội dung chất lượng của đề án, những kết quả đạt được

Đề án được trình bày trong 55 trang bao gồm phần mở đầu, 3 chương nội dung, phần kết luận và danh mục các tài liệu tham khảo.

Chương 1 của đề án gồm 8 trang, trình bày tổng quan về Chatbot, xử lý ngôn ngữ tự nhiên (NLP), mô hình ngôn ngữ lớn (LLM), kỹ thuật Retrieval-Augmented Generation (RAG), Embedding và Vector Database. Các nội dung trong chương được sắp xếp tương đối hợp lý đã giới thiệu tổng quan về các công nghệ được sử dụng trong đề án. Tuy nhiên, chương này chưa nêu rõ bài toán nghiên cứu cần giải quyết, chưa phân tích đầy đủ nhu cầu thực tiễn và những khó khăn của việc đọc hiểu tài liệu bằng các chatbot thông thường. Đề án cũng chưa chỉ ra rõ các hạn chế của các hệ thống chatbot hiện nay, từ đó làm nổi bật lý do lựa chọn kiến trúc RAG và phạm vi nghiên cứu của đề án. Ngoài ra, phần tổng quan mới dừng ở việc giới thiệu các công

nghệ, chưa có sự so sánh giữa các mô hình LLM, mô hình Embedding hoặc các Vector Database để làm cơ sở cho việc lựa chọn giải pháp triển khai ở các chương tiếp theo.

Nội dung chính của chương 2 là phân tích yêu cầu hệ thống, xây dựng kiến trúc tổng thể và mô tả quy trình xử lý của chatbot trợ lý đọc hiểu tài liệu được trình bày trong 14 trang. Kiến trúc đề xuất phù hợp với mục tiêu xây dựng hệ thống và thể hiện được mối liên hệ giữa các thành phần trong quá trình xử lý dữ liệu. Tuy nhiên, nội dung chương còn mang tính khái quát. Đề án chưa trình bày cụ thể các công nghệ, thư viện và nền tảng sẽ được sử dụng để hiện thực từng thành phần của hệ thống sẽ được thực nghiệm trong chương 3. Chẳng hạn, chưa mô tả rõ mô hình LLM được lựa chọn, mô hình Embedding, Vector Database, Framework xây dựng chatbot, mô hình RAG, cơ chế Chunking, chiến lược truy xuất dữ liệu cũng như tiêu chí lựa chọn các công nghệ này. Việc thiếu sự liên kết giữa phần thiết kế và phần triển khai làm giảm tính logic của đề án.

Chương 3 của đề án gồm 13 trang, trình bày các nội dung triển khai thử nghiệm xây dựng hệ thống chatbot trợ lý đọc hiểu tài liệu và minh họa các chức năng như tải tài liệu, hỏi đáp, tóm tắt và trích xuất thông tin. Kết quả cho thấy hệ thống hoạt động ổn định và đáp ứng được các yêu cầu chức năng cơ bản đặt ra của đề án. Tuy nhiên, phần thực nghiệm và đánh giá còn là điểm hạn chế của đề án. Đề án chưa mô tả rõ tập dữ liệu sử dụng để thử nghiệm như nguồn dữ liệu, số lượng tài liệu, lĩnh vực, kích thước và đặc điểm của dữ liệu đầu vào. Các kịch bản thử nghiệm chưa được trình bày rõ ràng. Các tiêu chí đánh giá cũng chưa được xác định rõ ràng; việc đánh giá chủ yếu dựa trên các ví dụ minh họa mà chưa sử dụng các chỉ số định lượng như độ chính xác câu trả lời, Answer Relevancy, Faithfulness, Context Precision, Context Recall, thời gian phản hồi hoặc mức độ hài lòng của người dùng. Đề án cũng chưa có phần so sánh kết quả với các chatbot hoặc phương pháp khác để chứng minh hiệu quả của giải pháp đề xuất, đồng thời chưa phân tích các trường hợp trả lời sai, hiện tượng sinh thông tin không chính xác và những hạn chế của hệ thống khi xử lý các tập tài liệu lớn hoặc nhiều định dạng khác nhau. Các tham số Chunk size = 600 và overlap = 150 được lựa chọn nhưng chưa có phân tích lý do phù hợp.

Đề án được chia thành các chương mục rõ ràng và cơ bản bám sát các nội dung theo đề cương đã được phê duyệt. Đề án được trình bày có logic khá chặt chẽ. Tuy nhiên, cần hiệu chỉnh lại nội dung và hình thức của báo cáo đề tài theo các góp ý sau đây:

- Báo cáo đề án cần đánh số trang 1 từ phần Mở đầu. Cần trình bày báo cáo đề án thành các chương mục theo đúng quy định của đề án thạc sĩ.

- Nội dung phần mở đầu cần trình bày ngắn gọn và theo đúng quy định.

- Phần giới thiệu các chương nên in nghiêng sẽ phù hợp hơn.

- Cần bổ sung các nội dung kết luận chương của mỗi chương nhằm tóm tắt các kết quả đạt được trong chương và liên kết đến nội dung sẽ nghiên cứu trong chương tiếp theo.

- Trong đề án còn có một số nhận định mang tính tuyệt đối cần hiệu chỉnh phù hợp.

Trong đề án còn có khá nhiều lỗi in ấn phải sửa chữa tại các trang vi, 15, 18, 25, ... Một số hình vẽ còn mờ cần hiệu chỉnh hoặc in màu.

III/ Những vấn đề cần giải thích thêm

(1) Giải thích rõ hơn về kiến trúc tổng thể và chuỗi xử lý tự động (Pipeline) trình bày trong mục 2.2. Các giai đoạn xử lý có quan tâm đến định dạng khác nhau của tài liệu không?

(2) Giải thích thêm về tập dữ liệu văn bản sử dụng trong thực nghiệm đã triển khai trong chương 3 của đề án bao gồm: số lượng tài liệu và các định dạng văn bản, nguồn dữ liệu và phương pháp tiền xử lý dữ liệu?

(3) Giải thích rõ hơn về các kết quả thực nghiệm: kết quả sinh văn bản tóm tắt tài liệu, kết quả trích xuất thông tin từ văn bản và thời gian đưa ra kết quả trả lời các truy vấn?

IV/ Kết luận

Tôi đồng ý cho học viên Nguyễn Quang Huy được bảo vệ đề án với đề tài "Nghiên cứu xây dựng Chatbot trợ lý đọc hiểu tài liệu" trước Hội đồng chấm đề án tốt nghiệp thạc sĩ của Học viện. Tuy nhiên, để đảm bảo chất lượng đề án tốt nghiệp thạc sĩ chuyên ngành Hệ thống thông tin, học viên cần chỉnh sửa đề án và giải trình rõ trước Hội đồng các vấn đề được góp ý trên đây.

Ngày 20 tháng 07 năm 2026

Người nhận xét

Vũ Văn Thỏa

BỘ KHOA HỌC VÀ CÔNG NGHỆ
HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

CỘNG HOÀ XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập - Tự do - Hạnh phúc

BÁO CÁO GIẢI TRÌNH
SỬA CHỮA, HOÀN THIỆN ĐỀ ÁN TỐT NGHIỆP

Họ và tên học viên: Nguyễn Quang Huy

Chuyên ngành: Hệ thống thông tin

Khóa: 2023 đợt 1

Tên đề tài: Nghiên cứu xây dựng chatbot trợ lý đọc hiểu tài liệu

Người hướng dẫn khoa học: TS. Dương Trần Đức.

Ngày bảo vệ: 22/07/2025

Các nội dung học viên đã sửa chữa, bổ sung trong đề án tốt nghiệp theo ý kiến đóng góp của Hội đồng chấm đề án tốt nghiệp:

TT	Ý kiến hội đồng	Sửa chữa của học viên
1	Chỉnh sửa lỗi soạn thảo, lỗi ngữ pháp, chính tả	Học viên đã rà soát, chỉnh sửa các lỗi soạn thảo, các lỗi ngữ pháp
2	Lược bỏ bớt nội dung của phần mở đầu, tập trung cho phần nội dung	Tiếp thu góp ý của Hội đồng, học viên đã lược bỏ bớt nội dung phần mở đầu, tập trung cho phần nội dung chính
3	Bổ sung mô tả nguồn dữ liệu thực nghiệm, đáng giá kết quả qua các tiêu chí cụ thể	Tiếp thu góp ý của Hội đồng, học viên đã bổ sung nguồn dữ liệu thực nghiệm và đánh giá kết quả qua các tiêu chí cụ thể.

Hà Nội, ngày tháng năm 2026

Ký xác nhận của

CHỦ TỊCH HỘI ĐỒNG CHẤM ĐỀ ÁN	THƯ KÝ HỘI ĐỒNG	NGƯỜI HƯỚNG DẪN KHOA HỌC	HỌC VIÊN
PGS.TS.Nguyễn Trọng Khánh	TS. Đặng Ngọc Hùng	TS. Dương Trần Đức	Nguyễn Quang Huy

Quang Huy Nguyễn

2025_DeAn_HTTT_NguyenQuangHuy_0702026_v2.pdf

Đề án tốt nghiệp khóa 2024 đợt 1 kỹ thuật

Khoa Đào tạo Sau đại học

Posts and Telecommunications Institute of Technology

Thông tin chi tiết về tài liệu

ID bài nộp
trn:oid:::1:3603294110

Ngày nộp
09:15 GMT+7 29 thg 6, 2026

Ngày tải xuống
10:01 GMT+7 29 thg 6, 2026

Tên tệp
2025_DeAn_HTTT_NguyenQuangHuy_0702026_v2.pdf

Kích cỡ tệp
1.1 MB

57 Trang

22.630 Từ

88.107 Ký tự

18% Tính tương đồng nói chung

Tổng cộng của tất cả các kết quả trùng khớp, bao gồm cả các nguồn trùng lặp, cho mỗi c...

Đã lọc khỏi Báo cáo

- Mục lục tham khảo
- Văn bản được trích dẫn
- Văn bản được trích dẫn
- Kết quả trùng khớp nhỏ (ít hơn 8 từ)

Nguồn hàng đầu

15% Nguồn Internet
5% Ấn bản
8% Bài tập được nộp (bài của học sinh)

