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

TÓM TẮT ĐỀ ÁN TỐT NGHIỆP THẠC SĨ

HÀ NỘI – NĂM 2025

Đề án tốt nghiệp được hoàn thành tại:
HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

Người hướng dẫn khoa học:
……

Phản biện 1: .....
Phản biện 2: .....

Đề án tốt nghiệp sẽ được bảo vệ trước Hội đồng chấm đề án tốt nghiệp thạc sĩ tại Học viện Công nghệ Bưu chính Viễn thông

Vào lúc: ..... giờ ..... ngày ..... tháng ..... năm .....

Có thể tìm hiểu đề án tốt nghiệp tại:

- Thư viện của Học viện Công nghệ Bưu chính Viễn thông

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.

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

Trên thế giới, lĩnh vực chatbot dựa trên LLMs và RAG là một trong những hướng nghiên cứu sôi động nhất hiện nay. Các tập đoàn công nghệ hàng đầu như Google, Meta, OpenAI, Microsoft không chỉ tập trung phát triển các LLMs ngày càng mạnh mẽ hơn mà còn đầu tư mạnh vào các framework hỗ trợ xây dựng ứng dụng RAG như LangChain, LlamaIndex.

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.

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.

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 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. 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. 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.

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. Large Language Models (LLM): Mô hình hiểu và tạo ngôn ngữ tự nhiên dùng cho tóm tắt, trả lời câu hỏi. 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.

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. 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. 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.

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.

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, phần phân đoạn và vector hoá, phần suy luận và hỏi đáp.

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.

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.

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, giúp máy tính tiếp nhận, phân tích và tạo ra ngôn ngữ con người cho các bài toán thực tế.

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.

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

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. Tóm lại 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. Sau quá trình xử lý prompt hoàn chỉnh được gửi tới mô hình Gemini. 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. 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.

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.

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.

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.

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.

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.

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.

1.3.1 Quy trình hoạt động của RAG

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.

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: 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.

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.

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; hỗ trợ nhiều định dạng tệp; thiết kế module tách rời các thành phần giúp dễ dàng thay thế công nghệ ở từng lớp mà không ảnh hưởng đến toàn bộ kiến trúc.

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.

1.4.1 Các hệ thống thương mại phổ biến

Điển hình là 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.

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.

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.

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.

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.

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. Toàn bộ thiết kế đều xoay quanh việc tích hợp các mô hình ngôn ngữ lớn LLM cùng RAG nhằm hỗ trợ người dùng trong việc đọc hiểu, tìm kiếm, tra cứu, hiểu sâu nội dung tài liệu và hỏi đáp.

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.

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.

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.

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.

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.

2.1.4 CrossEncoder reranker trong tầng truy xuất

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.

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 và Chain-of-Thought.

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.

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.

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

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.

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 ...”.

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.

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.

2.2.5 Luồng nạp tài liệu và lưu trữ theo từng document

Một phần cực kỳ quan trọng của kiến trúc là pipeline nạp tài liệu có luồng xử lý: xác định phần mở rộng tệp, 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 cụ thể với các tệp faiss_index.idx, mapping.npy, chunks.json, pages.json, metadata.json và original.*.

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.

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.

2.3.1 Thiết kế API REST

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 và xóa tài liệu.

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.

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.

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.

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: tính an toàn và khả năng tự phòng vệ của hệ thống; cấu trúc mã nguồn có tính tách lớp tương đối cao; cách hệ thống quản lý cache index.

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.

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.

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.

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.

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.

3.2 Bộ dữ liệu và kịch bản 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

3.2.1 Tập dữ liệu thực ngiệm

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

3.2.2 Tập câu hỏi đánh giá

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

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.

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.

3.3.1 Các chỉ số đánh giá chính

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.

3.4.1 Bảng tổng hợp kết quả

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

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.

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.

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, đóng góp về kiến trúc phần mềm và tối ưu hoá 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 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. 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. 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. 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. Hệ thống đã cho thấy khả năng vận hành với một kiến trúc tương đối gọn nhưng vẫn giữ được tính mở rộng. 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.

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. Một hướng phát triển khác là tăng cường khả năng giải thích và khả năng kiểm chứng của phản hồi.