

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC MỞ THÀNH PHỐ HỒ CHÍ MINH



VƯƠNG MINH KHÁNH

**NGHIÊN CỨU PHÁT TRIỂN CHATBOT TƯ VẤN MUA
HÀNG TRỰC TUYẾN DỰA TRÊN GPT-4O-MINI**

KHÓA LUẬN TỐT NGHIỆP
NGÀNH CÔNG NGHỆ THÔNG TIN

TP. HỒ CHÍ MINH, 2025

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC MỞ THÀNH PHỐ HỒ CHÍ MINH



VƯƠNG MINH KHÁNH

NGHIÊN CỨU PHÁT TRIỂN CHATBOT TƯ VẤN MUA
HÀNG TRỰC TUYẾN DỰA TRÊN GPT-4O-MINI

Mã số sinh viên: 2151050191

KHÓA LUẬN TỐT NGHIỆP
NGÀNH CÔNG NGHỆ THÔNG TIN

Giảng viên hướng dẫn: TS. LÊ VIẾT TUẤN

TP. HỒ CHÍ MINH, 2025

TRƯỜNG ĐẠI HỌC MỞ CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
THÀNH PHỐ HỒ CHÍ MINH **Độc lập – Tự do – Hạnh phúc**
KHOA CÔNG NGHỆ THÔNG TIN

GIẤY XÁC NHẬN

Tôi tên là: Vương Minh Khánh

Ngày sinh: 02/09/2003 Nơi sinh: TPHCM

Chuyên ngành: CNTT Mã sinh viên: 2151050191

Tôi đồng ý cung cấp toàn văn thông tin đồ án/ khóa luận tốt nghiệp hợp lệ về bản quyền cho Thư viện Trường Đại học Mở Thành phố Hồ Chí Minh. Thư viện Trường Đại học Mở Thành phố Hồ Chí Minh sẽ kết nối toàn văn thông tin đồ án/ khóa luận tốt nghiệp vào hệ thống thông tin khoa học của Sở Khoa học và Công nghệ Thành phố Hồ Chí Minh.

Ký tên

(Ghi rõ họ và tên)

Ý KIẾN CHO PHÉP BẢO VỆ ĐỒ ÁN/ KHÓA LUẬN TỐT NGHIỆP

CỦA GIẢNG VIÊN HƯỚNG DẪN

Giảng viên hướng dẫn:

Sinh viên thực hiện: Lớp:

Ngày sinh: Nơi sinh:

Tên đề tài:

.....
.....
.....
.....

Ý kiến của giảng viên hướng dẫn về việc cho phép sinh viên được bảo vệ đồ án/khoa luận trước Hội đồng:

Thành phố Hồ Chí Minh, ngày ... tháng ... năm

Người nhận xét

LỜI CẢM ƠN

NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

TÓM TẮT KHÓA LUẬN TỐT NGHIỆP

Nhu cầu tích hợp ChatBot vào trong hoạt động tư vấn bán hàng ngày càng được các doanh nghiệp quan tâm, và chúng thường được đưa vào trong nhiều website lớn nhỏ khác nhau. Thế nhưng, ChatBot hiện tại vẫn còn nhiều sai sót trong các tình huống, khi chúng thiếu tính linh hoạt và khả năng ứng biến với các trường hợp mới, hoặc trả lời kém hiệu quả trong các đoạn hội thoại dài và phức tạp. Một bất lợi lớn hơn không chỉ đối với các ChatBot, mà cũng như trong hoạt động tư vấn bán hàng của người thật, là bên tư vấn không thể chủ động dẫn dắt cuộc trò chuyện theo một quy trình bán hàng định sẵn. Người tư vấn/ChatBot đa phần chỉ phản hồi thụ động dựa trên câu hỏi của khách hàng, mà không đặt ra các câu hỏi khai thác thêm nhu cầu của họ, dẫn đến việc giảm mức độ tương tác của người dùng và giảm tỷ lệ chuyển đổi trong bán hàng.

Và để khắc phục vấn đề này, đề tài phát triển một ChatBot ứng dụng kiến trúc AI Agents — một xu hướng công nghệ mới nổi bật năm 2025 — nhằm phân chia nhiệm vụ theo từng tác tử (Agents) thông minh riêng biệt, giúp ChatBot không chỉ trả lời câu hỏi, mà còn chủ động đặt ra các câu hỏi mang tính chiến lược (followup), giúp dẫn dắt khách hàng qua từng giai đoạn trong quy trình. Cách tiếp cận này giúp tối ưu hóa việc thu thập dữ liệu, cải thiện tương tác của người dùng, và nâng cao tỷ lệ chuyển đổi cho phễu bán hàng của doanh nghiệp.

Hệ thống đã đạt được độ chính xác 80.58% trung bình, với thời gian trả lời và followup dao động từ 5-20 giây và dưới 5 giây, chỉ số hài lòng khách hàng (CSAT) đạt 4-5 điểm trong tổng 82.9% phiên hội thoại. Những thông số này không chỉ cho thấy khả năng trả lời đúng các yêu cầu tư vấn sản phẩm, mà còn thể hiện trải nghiệm người dùng tích cực với các câu hỏi của ChatBot. Kết quả khóa luận còn tạo tiền đề cho tính khả thi của kiến trúc AI Agents trong việc tối ưu hóa luồng nghiệp vụ của các Agents, nâng cao tính tự động hóa và mở rộng hệ thống — chỉ cần bổ sung Agent mà không cần can thiệp sau vào hệ thống hiện có.

ABSTRACT

ChatBots play a key role for customer–consultant interactions in modern business environments. Though widely used, conventional ChatBots are limited by their predetermined response protocols, making them ill-suited for more complex interactions. By doing so, these systems essentially respond to what a user has requested from them, rather than being able to direct conversations in a more strategic direction effectively; the result is that customer data collection is often fragmented, and subsequent conversion rates are much lower. Additionally, they fail to interact proactively resulting in user loss of interest in the long run.

This project proposes the development of an advanced chatbot based on the AI Agent architecture, a rising trend in the AI research community in 2025. By dividing the ChatBot into agents — each responsible for tasks such as orchestrating, product retrieval, follow-up questioning, and profile updating, the system not only responds to customer inquiries, but also proactively provides strategic follow-up questions to maintain the conversation flow and guides them through a robust sales workflow.

The Chatbot system achieved the average accuracy of 80.58%, with AI answers' response time fluctuating from 5-20 seconds and AI follow-ups' time latency under 5 seconds, and Customer Satisfaction (CSAT) ratings of 4–5 stars in 82.9% of 70 sessions. These statistics not only infer the ability to accurately satisfy general queries from customers, but also reflect a positive user experience throughout the AI chat session. What is more, the thesis establishes the viability of an AI Agent architecture of agents' orchestration, enhancing automation, and enabling seamless system scalability, where new functionalities can be simply integrated into production just by implementing additional agents without any major modifications.

MỤC LỤC

LỜI CẢM ƠN	3
NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN.....	4
TÓM TẮT KHÓA LUẬN TỐT NGHIỆP	5
ABSTRACT	6
DANH MỤC TỪ VIẾT TẮT.....	10
DANH MỤC HÌNH VẼ	11
DANH MỤC BẢNG	13
MỞ ĐẦU.....	14
Chương 1. TỔNG QUAN ĐỀ TÀI	15
1.1. Giới thiệu đề tài.....	15
1.2. Mục tiêu & Phạm vi đề tài	16
1.2.1. Mục tiêu.....	16
1.2.2. Phạm vi đề tài	16
1.3. Phương pháp nghiên cứu.....	16
1.4. Bố cục báo cáo	17
Chương 2. CƠ SỞ LÝ THUYẾT.....	18
2.1. LangChain	18
2.1.1. Giới thiệu	18
2.1.2. Lý do lựa chọn kiến trúc.....	18
2.1.3. Cách thức sử dụng	19
2.1.4. Ứng dụng trong đề tài.....	20
2.2. OpenAI Models	20
2.2.1. Giới thiệu	20
2.2.2. Lý do lựa chọn mô hình	21

2.2.3.	Cách thức hoạt động	21
2.2.4.	Ứng dụng trong đề tài.....	22
2.3.	AI Agents	22
2.3.1.	Giới thiệu	22
2.3.2.	Lý do lựa chọn kiến trúc.....	24
2.3.3.	Cách thức sử dụng	24
2.3.4.	Ứng dụng trong đề tài.....	26
2.4.	Weaviate.....	27
2.4.1.	Giới thiệu	27
2.4.2.	Lý do lựa chọn kiến trúc.....	28
2.4.3.	Cách thức sử dụng	29
2.4.4.	Ứng dụng trong đề tài.....	30
2.5.	Qdrant.....	30
2.5.1.	Giới thiệu	30
2.5.2.	Lý do lựa chọn kiến trúc.....	32
2.5.3.	Cách thức hoạt động	32
2.5.4.	Ứng dụng trong đề tài.....	33
2.6.	Các công nghệ phụ	33
2.6.1.	Redis - Giải pháp cache lịch sử chat	33
2.6.2.	Chatwoot – Nền tảng quản lý hội thoại của khách hàng cho ChatBot	
	34	
Chương 3.	CHATBOT TƯ VẤN BÁN HÀNG.....	34
3.1.	Giới thiệu bài toán.....	34
3.2.	Kiến trúc hệ thống	35
3.3.	Phân tích hệ thống	37

3.3.1.	Lược đồ Use Case hệ thống ChatBot	37
3.3.2.	Đặc tả Use Case:.....	39
3.4.	Thiết kế hệ thống.....	42
3.4.1.	Tổng quan kiến trúc AI Agents	42
3.4.2.	Xây dựng các AI Agents	46
3.4.3.	Xây dựng CRUD vectorstore	58
3.5.	Kết quả đề tài	60
3.5.1.	Dữ liệu	60
3.5.2.	Sơ lược về các độ đo kết quả của mô hình.....	62
3.5.3.	Đánh giá tầng mô hình (Task-Level Evaluation)	63
3.5.4.	Đánh giá tầng phiên hội thoại (Session-Level Evaluation)	65
3.5.5.	Giải thích về mất cân bằng lớp (Class Imbalance) giữa các loại câu hỏi khi kiểm tra hệ thống Chatbot	69
3.5.6.	Các chức năng hoàn chỉnh của hệ thống	71
Chương 4.	KẾT LUẬN & HƯỚNG PHÁT TRIỂN	74
4.1.	Kết luận	74
4.2.	Hướng phát triển	75
TÀI LIỆU THAM KHẢO		77
PHỤ LỤC		79

DANH MỤC TỪ VIẾT TẮT

Chữ viết tắt	Nghĩa
LLM(s)	Large Language Model(s)
GPT	Generative Pre-trained Transformer
RL	Reinforcement Learning
RAG	Retrieval-Augmented Generation
CRUD	Create, Read, Update, Delete

DANH MỤC HÌNH VẼ

Hình 2.1: Quá trình tải dữ liệu và phân mảnh dữ liệu (Nguồn: https://python.langchain.com/docs/tutorials/rag)	19
Hình 2.2: Kiến trúc hoạt động của Langchain RAG với lịch sử hội thoại (Nguồn: https://python.langchain.com/docs/tutorials/qa_chat_history).....	20
Hình 2.3: Mô hình RAG của OpenAI Models (Nguồn: https://platform.openai.com/docs/guides/optimizing-lm-accuracy/understanding-the-tools)	22
Hình 2.4: Kiến trúc cơ bản của một AI Agent (Nguồn: https://blog.ori.co/ai-agent-introduction)	25
Hình 2.5: Kiến trúc Weaviate Cloud sử dụng dữ liệu sản phẩm để xây dựng vectorstore bằng OpenAI Embeddings (Nguồn: https://weaviate.io/developers/weaviate/model-providers/openai/embeddings).....	29
Hình 2.6: Mô hình truy xuất vectorstore bằng Weaviate RESTful API và GraphQL API.....	29
Hình 2.7: Biểu đồ thể hiện độ trễ trung bình (ms) so với độ chính xác của Redis, Qdrant, Weaviate, Elasticsearch và Milvus trên tập dbpedia-openai-1M-1536-angular (cập nhật 06/2024) [13]	31
Hình 2.8: Tổng quan kiến trúc của Qdrant vectorstore (Nguồn: https://qdrant.tech/documentation/overview/).....	33
Hình 2.9: Hệ thống lưu trữ cache Redis (Nguồn: https://iconduck.com/icons/13186/redis-original-wordmark)	34
Hình 2.10: Nền tảng quản lý tin nhắn Chatwoot (Nguồn: https://dribbble.com/shots/10077118-Chatwoot-Logo-Animation)	34
Hình 3.1: Kiến trúc tổng thể của ChatBot tư vấn bán hàng	37
Hình 3.2: Sơ đồ Use case hệ thống ChatBot	38
Hình 3.3: Biểu đồ phân tán biểu diễn thời gian phản hồi của Agent 1 theo lượt.....	65
Hình 3.4: Biểu đồ phân tán biểu diễn thời gian phản hồi của Agent 1 và Agent 2 theo lượt.....	66
Hình 3.5: Biểu đồ phân tán biểu diễn thời gian phản hồi của Agent 3 theo lượt.....	66

Hình 3.6: Thang điểm cho CSAT (Nguồn: https://base.vn/blog/csat-la-gi/)	68
Hình 3.7: Biểu đồ cột thể hiện sự hài lòng của khách hàng qua cuộc hội thoại.....	68
Hình 3.8: Biểu đồ cột đôi thể hiện Số lượt truy vấn và tỷ lệ chính xác theo nhóm câu hỏi	70
Hình 3.9: Hộp hội thoại giữa khách hàng và ChatBot	71
Hình 3.10: Khi xóa cuộc hội thoại, thông báo sẽ hiện ra để người dùng bắt đầu cuộc trò chuyện mới	72
Hình 3.11: Cuộc trò chuyện với Chatbot sẽ không được gán (unassigned) cho tư vấn viên nào	73
Hình 3.12: Khi người dùng nhập từ khóa “tuvanvien”, cuộc hội thoại được chuyển cho tư vấn viên	73
Hình 3.13: Khi này, cuộc hội thoại chuyển cho tư vấn viên (gán cho Mine – hộp thoại của tôi)	74

DANH MỤC BẢNG

Bảng 3.1: Bảng đặc tả Use Case Chat tư vấn bán hàng	39
Bảng 3.2: Bảng đặc tả Use Case Chuyển phiên chat cho tư vấn viên.....	40
Bảng 3.3: Bảng đặc tả Use Case CRUD vectorstore.....	41
Bảng 3.4: Các thông tin đầu vào (input) dùng chung cho các AI Agents.....	42
Bảng 3.5: Các thông tin đầu vào (input) dùng chung cho các AI Agents.....	44
Bảng 3.6: Giải thích ý nghĩa của đối tượng JSON được trả về của Agent 1.....	47
Bảng 3.7: Một vài dòng đầu trong CSDL.....	60
Bảng 3.8: Một vài dòng đầu trong nội dung kiểm tra hiệu quả ChatBot	61
Bảng 3.9: Độ chính xác Precision Soft theo Agent.....	63
Bảng 3.10: Ngẫu nhiên 8 câu hỏi thuộc nhóm Câu hỏi tổng quát.....	69

Chương 1. TỔNG QUAN ĐỀ TÀI

1.1. Giới thiệu đề tài

Một trong những ứng dụng nổi bật nhất trong thời đại AI hiện nay, là sự tích hợp các công cụ Chatbot vào trong hoạt động tương tác giữa người bán và người mua. Chatbot được nhiều công ty tin dùng, vì chúng giúp tự động hóa công việc tư vấn và chăm sóc khách hàng. Các nền tảng Chatbot cũng liên tục được phát triển và cải tiến, giúp chúng xử lý đa dạng các loại câu hỏi, cung cấp thông tin sản phẩm nhanh chóng và hỗ trợ phần nào khói lượng công việc cho tư vấn viên. Tuy nhiên, không khó để thấy rằng, Chatbot không hoàn toàn hữu hiệu, mà chúng vẫn bộc lộ nhiều bất cập khi trao đổi với con người, đặc biệt là trong các cuộc hội thoại dài, phức tạp, không đầy đủ ngữ cảnh [1].

Ngoài ra, không chỉ ở Chatbot, mà bản thân các tư vấn viên, thường chỉ phản hồi một cách thụ động, chủ yếu “hỏi gì - đáp nấy”, ít đặt lại câu hỏi cho khách hàng để khai thác thông tin thêm, và không có khả năng dẫn dắt họ theo một quy trình bán hàng cụ thể [2]. Điều này khiến việc xác định nhu cầu thực sự của người dùng hạn chế hơn, gây gián đoạn luồng trò chuyện giữa hai bên, từ đó làm suy giảm khả năng bán hàng [1] [2].

Từ những khó khăn trên, đề tài tập trung phát triển ChatBot theo thiên hướng chủ động cung cấp những câu hỏi dẫn dắt (followup) theo quy trình bán hàng của doanh nghiệp, cho phép hệ thống xác định người dùng đang ở trong giai đoạn nào của quy trình và khai thác sâu hơn chân dung khách hàng, từ đó gợi ý sản phẩm phù hợp hơn, dễ chốt đơn hơn.

Bên cạnh đó, khóa luận còn tích hợp kiến trúc AI Agent — một trong những xu hướng nổi bật nhất năm 2025 trong cộng đồng nghiên cứu về AI [3], nhằm tăng khả năng lập kế hoạch, điều phối, phối hợp tác vụ giữa các thành phần (Agents), giúp cá nhân hóa phản hồi của hệ thống dành cho người dùng, cũng như linh hoạt hơn trong mở rộng Chatbot với những chức năng bên ngoài.

1.2. Mục tiêu & Phạm vi đề tài

1.2.1. Mục tiêu

Dự án hướng đến việc khắc phục các vấn đề đã nêu ở trên, nhằm phát triển một ChatBot không chỉ phản hồi các câu truy vấn như thường lệ, mà còn có khả năng chủ động đưa ra những câu hỏi dẫn dắt người dùng theo quy trình bán hàng được thiết kế sẵn, từ đó thu thập thông tin khách hàng và đưa khách hàng đi theo đúng trình tự cần thiết để gia tăng khả năng chốt đơn hàng.

Ngoài ra, các thành phần của ChatBot sẽ được thiết kế như những AI Agents - những chức năng hoạt động cho từng tác vụ riêng biệt, và có thể học hỏi dựa vào phản hồi của người dùng.

1.2.2. Phạm vi đề tài

Dự án chỉ tập trung phát triển khả năng đưa ra các câu hỏi dẫn dắt khách hàng dựa theo quy trình, cách thức thiết kế prompt cho LLM cho từng nghiệp vụ trong một chu trình hỏi đáp, và khả năng thiết kế kiến trúc AI Agents.

ChatBot sẽ được ứng dụng và thử nghiệm cho các doanh nghiệp vừa và nhỏ tại Việt Nam về các thiết bị âm thanh như loa, micro, amplifier,... đang có nhu cầu sử dụng ChatBot cho nghiệp vụ tư vấn sản phẩm.

1.3. Phương pháp nghiên cứu

Đề tài bắt đầu bằng việc nghiên cứu các tài liệu liên quan đến Trí tuệ nhân tạo (AI), đặc biệt là các LLMs, các framework hỗ trợ xây dựng bot đơn giản và dễ hiểu.

GPT-4o-mini được sử dụng để làm “bộ não” chính cho các AI Agents, đồng thời sử dụng OpenAI Embeddings làm công cụ quan trọng để cho các vector stores và chuyển đổi các câu truy vấn thành dạng vector để truy vấn ngữ nghĩa (semantic search).

Sau đó, hệ thống nghiên cứu, xây dựng vectorstore phù hợp, đặc biệt với Weaviate. Sau khi tiền xử lý, tệp dữ liệu được vector hóa và lưu trữ trên Weaviate Cloud. Đồng thời, GraphQL và RESTful API để cập nhật nhanh chóng cơ sở dữ liệu (CSDL).

Cùng với thư viện LangChain, xây dựng RAG để tạo nên một chatbot đơn giản, có khả năng quản lý hội thoại, lịch sử chat và truy vấn vectorstore.

Sau đó, kiến trúc AI Agents được áp dụng để chia hệ thống thành các Agents riêng lẻ, thực hiện chuyên cho một nghiệp vụ, và kết hợp với nhau thành một hệ thống tổng thể.

Và cuối cùng, thu thập lịch sử chat thật giữa các khách hàng với người bán hàng, nhằm đánh giá hiệu suất của mô hình, bao gồm độ chính xác, thời gian phản hồi và độ hài lòng của người dùng sau khi trò chuyện với chatbot.

1.4. Bô cục báo cáo

Báo cáo này bao gồm 4 chương chính, lần lượt trình bày về tổng quan đề tài, các công nghệ được sử dụng, chi tiết cách kiến trúc xây dựng ChatBot, cũng như kết quả độ tin cậy và phương hướng phát triển trong tương lai. Các chương bao gồm:

Chương 1: TỔNG QUAN ĐỀ TÀI

Chương 2: CƠ SỞ LÝ THUYẾT

Chương 3: CHATBOT TƯ VẤN BÁN HÀNG

Chương 4: KẾT LUẬN & HƯỚNG PHÁT TRIỂN

Chương 2. CƠ SỞ LÝ THUYẾT

2.1. LangChain

2.1.1. Giới thiệu

LangChain là framework mạnh mẽ và thông dụng trong việc thiết kế các ứng dụng AI [4]. Nó cung cấp các công cụ cùng chung hệ sinh thái để xây dựng các phần mềm, triển khai và quản lý hiệu suất, mở rộng hệ thống. Nhờ đó, LangChain trở nên phù hợp với các ứng dụng như Chatbot, với khả năng nhận diện ngữ cảnh, tích hợp APIs linh hoạt, giúp các nghiệp vụ quản lý luồng hội thoại, xử lý ngôn ngữ, và tích hợp dữ liệu trở nên thuận tiện hơn [4]. Ngoài ra, LangChain còn các thư viện cho việc xử lý dữ liệu (data processing) phục vụ xây dựng cơ sở kiến thức (knowledge base) cho hệ thống, bằng cách chia dữ liệu thành các mảnh (chunks), tổng hợp các mảnh vào đối tượng Documents cùng với các thông tin metadata, rất hữu dụng cho việc lập chỉ mục để ChatBot truy vết và chỉnh sửa (CRUD) sau này [5].

LangChain sở hữu một số ưu điểm khá nổi bật, khiến nó luôn là framework được ưu tiên để phát triển Chatbot. Đầu tiên, công nghệ được phát triển với kiến trúc hiện đại, thân thiện đối với lập trình viên mới bắt đầu tìm hiểu tới. Tiếp đến, nó cung cấp các công cụ, thư viện hỗ trợ xử lý ngôn ngữ tự nhiên và hội thoại một cách tiện lợi. Ba là, dễ dàng tích hợp với các mô hình AI, CSDL khác nhau như các mô hình LLMs của ChatGPT, hoặc các vectorstore,...

Tuy nhiên, LangChain cũng tồn tại một số nhược điểm nhất định, khi yêu cầu kiến thức kỹ thuật về kiến trúc, tham số để cấu hình và triển khai hệ thống hiệu quả. Hiệu quả của ChatBot vẫn phụ thuộc vào các yếu tố khác như mô hình GPT phù hợp, dữ liệu huấn luyện,... do LangChain chỉ là framework xây dựng lên khung phần mềm, chưa phải là “bộ não” cho ChatBot.

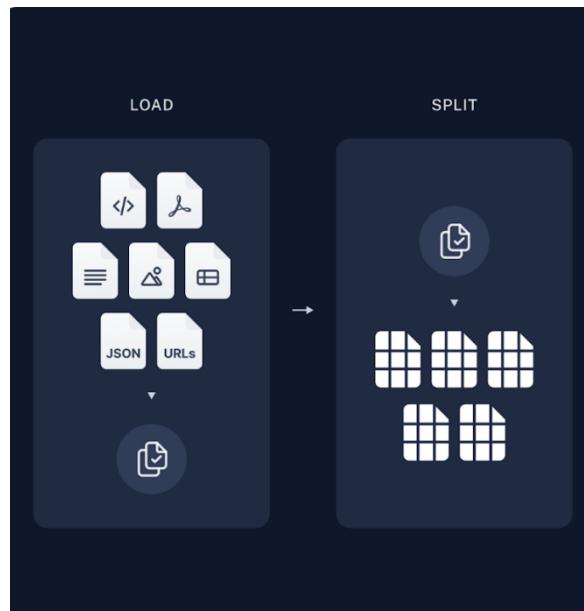
2.1.2. Lý do lựa chọn kiến trúc

LangChain là framework dễ tiếp cận với người dùng mới, cũng như tích hợp các công cụ và API có sẵn, giúp dễ dàng triển khai và mở rộng hệ thống một cách linh hoạt.

2.1.3. Cách thức sử dụng

2.1.3.1. Phân mảnh dữ liệu ban đầu thành các chunks

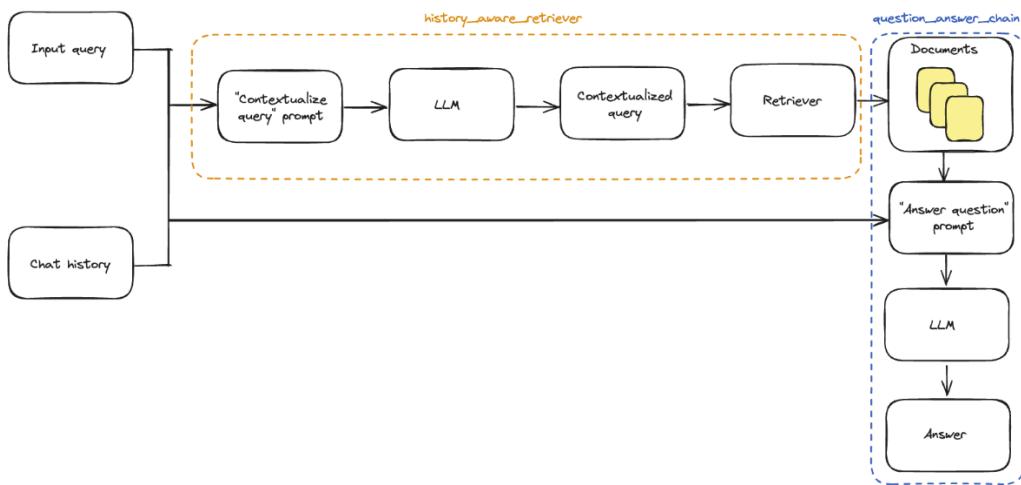
Sau khi nạp dữ liệu bằng phương thức Document Loaders của LangChain, toàn bộ dữ liệu sẽ được duyệt để phân mảnh (split) và lưu vào đối tượng Documents với metadata của nó.



Hình 2.1: Quá trình tải dữ liệu và phân mảnh dữ liệu (Nguồn:

<https://python.langchain.com/docs/tutorials/rag>)

Langchain kết hợp yêu cầu của người dùng (query) và lịch sử hội thoại (chat history) để xây dựng yêu cầu có ngữ cảnh (contextualized query). Sau đó, truy vấn vào vectorstore để trả về các đối tượng Document có liên quan đến yêu cầu, và sử dụng LLM của OpenAI để trả về phản hồi phù hợp nhất.



Hình 2.2: Kiến trúc hoạt động của Langchain RAG với lịch sử hội thoại (Nguồn: https://python.langchain.com/docs/tutorials/qa_chat_history)

2.1.4. Ứng dụng trong đề tài

Là “xương sống” cho các hoạt động phản hồi của chatbot. Khi người dùng nhập câu hỏi, hệ thống sẽ truy xuất các tài liệu liên quan từ các bộ nhớ, sau đó truyền cho model GPT-4o-mini để trả về câu trả lời cho khách hàng đúng với ngữ cảnh và lịch sử tin nhắn nhất.

2.2. OpenAI Models

2.2.1. Giới thiệu

OpenAI Models gồm các mô hình LLM của nhà phát triển OpenAI, bao gồm các loại mô hình khác nhau được từng khả năng khác nhau. Để phân tích và trả lời hội thoại, khóa luận dùng model GPT-4o-mini, là phiên bản khá mạnh mẽ, chuyên về các tác vụ yêu cầu tốc độ phản hồi nhanh mà không yêu cầu sự suy luận quá cao cấp.

Ngoài ra, OpenAI Embeddings là mô hình được sử dụng để vector hóa dữ liệu huấn luyện dạng text, để đo độ tương đồng giữa các đoạn văn bản với nhau. Việc embedding là đặc biệt quan trọng để hỗ trợ tìm kiếm ngữ nghĩa từ các câu hỏi.

Mô hình GPT-4o-mini là model nhanh nhất và tiết kiệm nhất của OpenAI, tập trung tối ưu tốc độ phản hồi mà xử lý được nhiều tác vụ, phù hợp trong việc tương tác với người dùng đại chúng [6]. Ngoài ra, các mô hình được cung cấp dưới dạng API, dễ

dàng tích hợp vào hệ thống ChatBot và các mô hình khác. Không những vậy, OpenAI thường xuyên tinh chỉnh và cho ra đời các mô hình mạnh mẽ hơn, tối ưu chi phí hơn. Khi đó, hệ thống có thể sử dụng các mô hình mới để cải thiện độ chính xác cho phản hồi sau này.

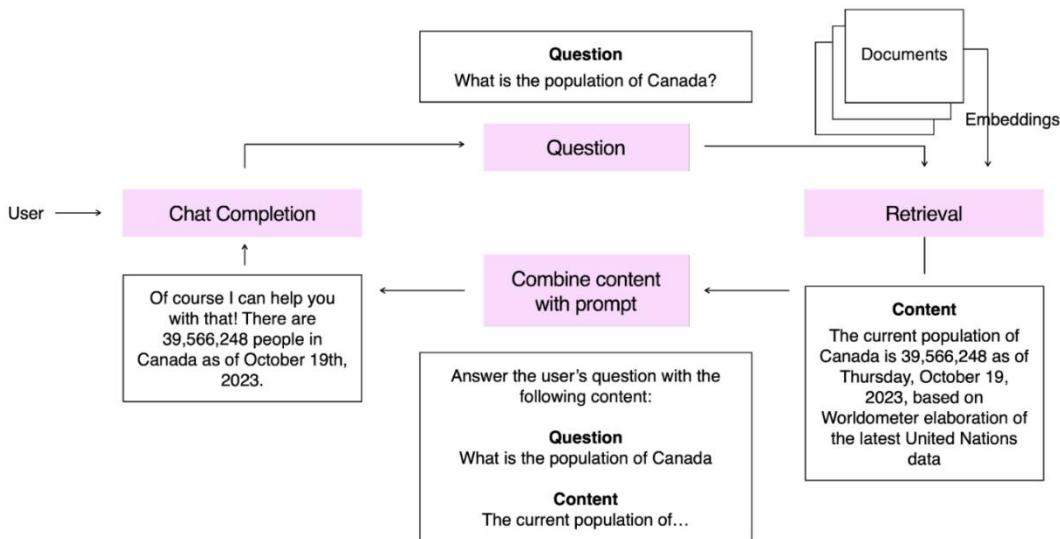
Tuy vậy, các APIs được tính phí theo từng lượt truy vấn, từng loại mô hình, và độ lớn của tập dữ liệu cần được vector hóa, yêu cầu lượng chi phí lớn cho từng lượt phản hồi mà hệ thống Chatbot bắt buộc phải có, cũng như không thể sử dụng tính năng batch của OpenAI để tiết kiệm chi phí.Thêm vào đó, nếu như mô hình ChatBot không được huấn luyện, tinh chỉnh, cũng như prompt engineering không tốt, thì việc phân tích và trả lời câu hỏi của mô hình sẽ trở nên kém chính xác và phù hợp.

2.2.2. Lý do lựa chọn mô hình

Việc sử dụng OpenAI Models trong dự án này là lựa chọn phù hợp cho đê tài, không chỉ tận dụng điểm mạnh về tốc độ và sự tối ưu chi phí của mô hình GPT-4o-mini, mà đây cũng là mô hình nổi tiếng, tương đương với nhiều cộng đồng hỗ trợ về các công nghệ này và các công nghệ liên quan. Điều này giúp ChatBot có xử lý tối ưu câu hỏi và phản hồi, từ đó nâng cao trải nghiệm người dùng.

2.2.3. Cách thức hoạt động

OpenAI models, cụ thể là mô hình GPT-4o-mini (hoặc mô hình này được học tăng cường - Reinforcement Learning bởi lập trình viên), sẽ dựa vào prompt để phiên dịch lại câu truy vấn của khách hàng và truy vết dữ liệu trong cơ sở kiến thức, rồi dùng nó để tạo sinh ra câu trả lời hoàn chỉnh.



Hình 2.3: Mô hình RAG của OpenAI Models (Nguồn:

<https://platform.openai.com/docs/guides/optimizing-lm-accuracy/understanding-the-tools>)

2.2.4. Ứng dụng trong đề tài

OpenAI model GPT-4o-mini: Là bộ não cho các Agent ra quyết định, diễn đạt câu truy vấn và phản hồi cho người dùng, cũng như cập nhật thông tin khách hàng.

OpenAI Embeddings: Dùng để vector hóa câu truy vấn của người dùng thành vector. Nhờ đó, hệ thống có thể truy xuất chính xác các bộ nhớ bên ngoài như Weaviate và Qdrant vectorstore, giúp tìm các sản phẩm và gợi ý phù hợp với truy vấn.

2.3. AI Agents

2.3.1. Giới thiệu

Nếu như LLMs đóng vai trò nhận diện văn bản và tạo sinh nội dung dựa trên prompt, thì AI Agents là các hệ thống tự động hóa, được thiết kế để nhận diện môi trường, đưa ra quyết định và thực thi hành động để đạt kết quả nào đó [7]. Để dễ hiểu hơn, AI Agent là LLM, nhưng không chỉ dừng lại ở việc trả lời câu hỏi, mà còn có thể thay thế hoạt động của con người để lập kế hoạch, ra quyết định và đưa ra các hành động để trả

về các kết quả thực tế, cũng như học hỏi từ các hoạt động trong quá khứ để cá nhân hóa kết quả cho người dùng.

Mô hình AI Agents mang lại nhiều ưu điểm nổi bật, góp phần nâng cao hiệu quả của hệ thống ChatBot đa tác vụ. Trước hết, AI Agents có khả năng tự động lập kế hoạch và phân chia công việc một cách rõ ràng, đồng thời có thể chủ động gọi các API cần thiết để hoàn thành các tác vụ mà người dùng không cần can thiệp trực tiếp. Tính năng này đặc biệt hữu ích trong các hoạt động dịch vụ, chăm sóc khách hàng, nơi việc phản hồi tự động và đúng quy trình đóng vai trò then chốt trong nâng cao trải nghiệm người dùng [8]. Ngoài ra, AI Agents khắc phục được hạn chế bộ nhớ của các LLMs truyền thống bằng cách tận dụng các hệ thống bộ nhớ ngoài như vectorstore, giúp lưu trữ và truy xuất thông tin một cách hiệu quả hơn.

Chính nhờ vào khả năng ghi nhớ được cải thiện này, hệ thống có thể gia tăng mức độ cá nhân hóa khi phản hồi theo từng khách hàng, dựa trên việc học hỏi từ các cuộc hội thoại trước đó. Một lợi ích khác là tính đóng gói và khả năng mở rộng linh hoạt của từng Agent: mỗi tác tử có thể đảm nhiệm một nghiệp vụ riêng biệt và được tích hợp vào hệ thống tổng thể mà không cần chỉnh sửa kiến trúc lõi, giúp tối ưu hóa quy trình phát triển và bảo trì hệ thống.

Tuy nhiên, việc áp dụng AI Agents cũng kéo theo một số thách thức nhất định. Các tác tử này không thể đảm bảo đưa ra kết quả chính xác tuyệt đối, và khi số lượng bước xử lý tăng lên, nguy cơ lệch hướng hoặc sai sót trong quá trình suy luận cũng tăng theo [9]. Mỗi lần AI Agent gọi mô hình ngôn ngữ hoặc thực thi API sẽ phát sinh chi phí về token đầu vào và đầu ra, đồng thời làm gia tăng độ trễ phản hồi. Điều này khiến cho các hệ thống đa tác tử (multi-agent systems) thường gặp phải hiện tượng chồng chéo chu kỳ request-response, gây tốn kém chi phí vận hành và làm giảm hiệu năng thời gian thực [10].

Bên cạnh đó, việc kiểm soát nhiều Agent hoạt động song song cũng đặt ra yêu cầu cao về kỹ thuật, đặc biệt là nguy cơ "ảo hóa" (AI hallucination) – khi mô hình tạo ra thông tin không chính xác dựa trên suy luận sai. Việc kiểm thử và gỡ lỗi (test/debug) cho từng Agent cũng trở nên phức tạp hơn, bởi quy trình không chỉ đơn thuần là kiểm tra luồng code tuyến tính, mà đòi hỏi phải theo dõi logic suy luận và trả lời của mô hình ngôn ngữ một cách kỹ càng [11].

2.3.2. Lý do lựa chọn kiến trúc

Kiến trúc của AI Agents cho phép hệ thống Chatbot được chia thành nhiều thành phần tiếp sức cho nhau, từng phần sẽ khai thác tốt nhất khả năng của LLM cho từng nghiệp vụ. Từ đó, việc tích hợp các yêu cầu như hiểu rõ quy trình bán hàng, truy xuất các tầng bộ nhớ, và việc ra quyết định phân bổ công việc cho các Agents sẽ mang lại kết quả tốt nhất, cũng như việc tích hợp thêm các tác vụ rời khác vào trong hệ thống cũng sẽ trở nên đơn giản và được tự động chọn lọc sao cho phù hợp với ngữ cảnh nhất, giúp tăng trải nghiệm cá nhân hóa cho người dùng.

Ngoài ra, AI Agents đã trở thành một trong những công nghệ nổi bật nhất, được đề cập khá sôi nổi trong thời gian gần đây. Vì thế, việc tích hợp kiến trúc AI Agent vào trong đề tài này, không chỉ để tận dụng các điểm mạnh đã được nêu trên, mà còn là cơ hội để bản thân thực hành ngay xu hướng cực kỳ tiềm năng đang được thị trường và doanh nghiệp đầu tư. Điều này cho phép khóa luận tăng tính bám sát nhu cầu thực tiễn, vừa cập nhật kỹ năng bản thân cho các yêu cầu của doanh nghiệp trong tương lai.

2.3.3. Cách thức sử dụng

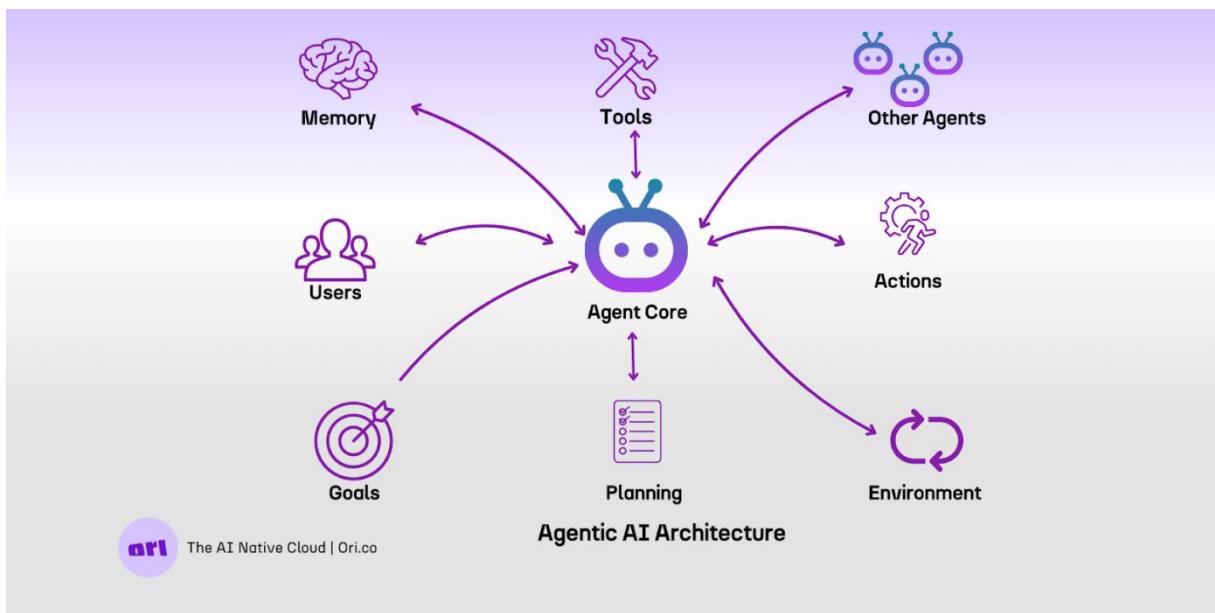
Một AI Agent hiện đại được cấu thành từ nhiều thành phần chính, mỗi thành phần đảm nhiệm một vai trò cụ thể trong quá trình xử lý thông tin và ra quyết định. Đầu tiên, **mô hình ngôn ngữ lớn (LLM)** đóng vai trò là “bộ não” của Agent, có chức năng hiểu ngữ nghĩa của thông tin đầu vào và tạo ra phản hồi dưới dạng ngôn ngữ tự nhiên. Các mô hình điển hình thường được sử dụng là GPT-4, Claude, hay các biến thể tối ưu hóa tốc độ như GPT-4o-mini.

Tiếp theo là **hệ thống bộ nhớ**, đóng vai trò như “trí nhớ” của Agent, cho phép lưu giữ và truy xuất thông tin trong suốt quá trình hoạt động. Hệ thống này bao gồm ba loại chính. Thứ nhất, **bộ nhớ ngắn hạn (short-term memory)**, chính là cửa sổ ngữ cảnh (window context) của LLM – chẳng hạn như GPT-4 hỗ trợ tới 128K tokens – giúp duy trì mạch hội thoại và thông tin liên kết ngắn hạn. Thứ hai, **bộ nhớ dài hạn (long-term memory)**, thường được lưu trữ trong các vectorstore như Weaviate hoặc FAISS, giúp hệ thống truy xuất kiến thức một cách bền vững giữa các phiên tương tác. Thứ ba là **bộ**

nhớ sự kiện (episodic memory), dùng để ghi lại các thông tin cụ thể phát sinh trong một phiên nhất định, bao gồm lịch sử trò chuyện, hồ sơ người dùng hoặc trạng thái tiến trình của một quy trình nào đó.

Ngoài ra, một thành phần quan trọng không thể thiếu là **khả năng lập kế hoạch và hành động**. Đây là chức năng cho phép Agent đưa ra quyết định về các bước xử lý kế tiếp và thực thi các hành động cụ thể – chẳng hạn như gọi API bên ngoài hoặc điều phối các Agent khác để hoàn thành nhiệm vụ một cách hiệu quả. Cuối cùng, **khả năng học hỏi từ các tương tác trước đó** giúp AI Agent tích lũy kinh nghiệm, ghi nhớ các tình huống đã xử lý và tái sử dụng kiến thức đó để đưa ra phản hồi ngày càng chính xác và cá nhân hóa hơn trong các lần tương tác sau.

AI Agents còn nhiều hơn các thành phần quan trọng để quy trình hoạt động trở nên tự động nhất. Tuy nhiên, do phạm vi của đề tài chưa khai thác hết tất cả các khả năng của AI Agents, nên chúng sẽ không được đề cập hết trong khóa luận này.



Hình 2.4: Kiến trúc cơ bản của một AI Agent (Nguồn: <https://blog.ori.co/ai-agent-introduction>)

Tương tự như kiến trúc tổng quát của AI Agent đã được trình bày, hệ thống ChatBot trong đề tài này cũng áp dụng mô hình phân tách tác vụ dựa trên các thành phần Agent chuyên biệt, với từng Agent đảm nhiệm một vai trò rõ ràng và có khả năng phối hợp chặt chẽ với nhau. Trước hết, **mô hình ngôn ngữ lớn** được sử dụng là GPT-4o-mini, được tích hợp thông qua framework LangChain nhằm thực hiện các thao tác gọi API

và tương tác với dữ liệu ngữ cảnh. LLM này đóng vai trò trọng yếu trong việc tiếp nhận thông tin đầu vào, xử lý ngữ nghĩa và sinh phản hồi phù hợp với từng tình huống, tùy thuộc vào vai trò mà mỗi Agent đảm nhiệm.

Tiếp theo là **hệ thống bộ nhớ**, đóng vai trò ghi nhớ và cung cấp thông tin ngữ cảnh cần thiết trong toàn bộ phiên trò chuyện. Cụ thể, bộ nhớ ngắn hạn được hiểu là cửa sổ ngữ cảnh mặc định của mô hình GPT-4o-mini, giữ vai trò liên kết thông tin giữa các lượt tương tác trong cùng một phiên. **Bộ nhớ dài hạn** được chia theo chức năng của từng Agent: trong khi Weaviate vectorstore được sử dụng để truy xuất thông tin sản phẩm, thì Qdrant vectorstore đóng vai trò lưu trữ các logic nghiệp vụ và phản hồi của người dùng trước đó, phục vụ cho các nhiệm vụ phức tạp hơn như định hướng hội thoại hoặc xử lý từ chối. Ngoài ra, hệ thống còn sử dụng **bộ nhớ sự kiện**, bao gồm lịch sử trò chuyện, hồ sơ khách hàng (như nhu cầu, sở thích, ngân sách), và thông tin về vị trí hiện tại của người dùng trong quy trình bán hàng (ví dụ: phân loại khách hàng, xử lý từ chối,...).

Về mặt lập **kế hoạch và hành động**, hệ thống được thiết kế để Agent đầu tiên (Agent Orchestrator) phân tích câu truy vấn đầu vào, truy xuất các thông tin cần thiết từ bộ nhớ sự kiện và đưa ra quyết định về việc kích hoạt các Agent tiếp theo. Các Agents trong hệ thống không trực tiếp gọi API bên ngoài để thực hiện hành động, mà sẽ phản hồi dưới dạng đối tượng JSON – đóng vai trò như một gói dữ liệu truyền tiếp cho các Agent tiếp theo, theo đúng mô hình hoạt động “tiếp sức” (relay-based execution). Cơ chế này đảm bảo rằng từng Agent sẽ chỉ xử lý phần việc chuyên biệt của mình, từ đó nâng cao tính modular và dễ bảo trì cho toàn hệ thống.

Cuối cùng, hệ thống còn được trang bị khả năng **học hỏi từ các tương tác trong quá khứ**, thông qua việc tổng hợp thông tin hồ sơ khách hàng và lịch sử hội thoại trước đó. Điều này cho phép các Agent có thể nhận diện lại nhu cầu, điểm đau hoặc đặc điểm cá nhân của khách hàng trong các phiên chat kế tiếp, từ đó đưa ra phản hồi ngày càng chính xác, tự nhiên và cá nhân hóa, góp phần nâng cao trải nghiệm tổng thể của người dùng.

2.3.4. Ứng dụng trong đề tài

Hệ thống Chatbot được cấu trúc thành bốn Agent chính:

Agent 1 – Agent Orchestrator: thu thập các thông tin ngữ cảnh cần thiết từ các nguồn như luồng chat, vectostore

Agent 2 – Product Retriever & Responser: Truy vấn thông tin sản phẩm từ Weaviate vectorstore và phản hồi câu trả lời phù hợp cho khách hàng.

Agent 3 – Followup Generator: Đánh giá cuộc hội thoại, rồi dự đoán giai đoạn tiếp theo trong quy trình, để đưa ra câu hỏi dẫn dắt.

Agent 4 – User Profile Updater: Cập nhật thông tin khách hàng (nhu cầu, ngân sách, điểm đau,...) dựa trên lịch sử hội thoại, giúp cá nhân hóa thông tin ngữ cảnh các Agents sau.

2.4. Weaviate

2.4.1. Giới thiệu

Weaviate Cloud (WCD) là một nền tảng cơ sở dữ liệu vector (vectorstore) mã nguồn mở, dễ mở rộng và linh hoạt [12], được thiết kế với mục tiêu tối ưu khả năng lưu trữ và truy vấn vector hiệu quả trong các hệ thống trí tuệ nhân tạo. Với tính chất linh hoạt và khả năng mở rộng tốt, WCD cho phép tích hợp dễ dàng vào các kiến trúc ChatBot hiện đại, đặc biệt là những hệ thống cần thực hiện tìm kiếm ngữ nghĩa và phản hồi theo ngữ cảnh. Trong khuôn khổ đề tài này, WCD không chỉ đóng vai trò là công cụ truy vết dữ liệu phục vụ phản hồi ChatBot, mà còn đảm nhiệm chức năng quan trọng trong việc quản lý và cập nhật dữ liệu thông qua cơ chế CRUD.

Cụ thể, nền tảng hỗ trợ hai giao thức chính là **RESTful API** và **GraphQL API**, cho phép tương tác với vectorstore một cách linh hoạt. RESTful API được sử dụng cho các thao tác quản trị cơ sở dữ liệu như thêm mới, truy vấn, cập nhật và xóa dữ liệu sản phẩm – tương đương với các thao tác CRUD truyền thống. Trong khi đó, GraphQL API cho phép truy vấn dữ liệu một cách chi tiết và có chọn lọc, giúp hệ thống chỉ lấy về các thuộc tính thực sự cần thiết để phục vụ cho các thao tác kế tiếp của RESTful API. Sự kết hợp giữa hai API này mang lại hiệu quả cao trong việc tối ưu luồng dữ liệu và giảm thiểu chi phí tính toán, đặc biệt trong các hệ thống ChatBot có khối lượng truy vấn lớn và yêu cầu phản hồi theo thời gian thực.

Weaviate Cloud (WCD) sở hữu nhiều ưu điểm nổi bật, khiến nó trở thành lựa chọn lý tưởng cho các hệ thống cần lưu trữ và truy vấn dữ liệu vector hiệu quả. Trước hết, nền tảng này được tối ưu hóa để hoạt động với nhiều loại mô hình embedding khác nhau, điển hình là text-embedding-3-large, giúp tăng cường khả năng tìm kiếm ngữ nghĩa (semantic search) và truy vấn gần đúng (approximate search) trong không gian vector. Ngoài ra, thư viện khách hàng (client library) của Weaviate được xây dựng thân thiện và hỗ trợ tốt cho ngôn ngữ Python, tạo điều kiện thuận lợi trong việc tích hợp vào các ứng dụng ChatBot, đặc biệt là trong môi trường phát triển nhanh như hiện nay.

Một điểm mạnh khác là khả năng hỗ trợ **GraphQL API**, cho phép hệ thống thực hiện truy vấn dữ liệu linh hoạt với cú pháp rõ ràng, đồng thời tối ưu hóa hiệu suất truyền tải dữ liệu giữa client và server. Nhờ hoạt động trên nền tảng điện toán đám mây, Weaviate Cloud còn mang lại khả năng mở rộng quy mô linh hoạt mà không cần triển khai hạ tầng vật lý riêng, giúp giảm áp lực vận hành cho các doanh nghiệp vừa và nhỏ [12].

Tuy nhiên, việc sử dụng Weaviate cũng tiềm ẩn một số hạn chế cần cân nhắc. Trước hết, đối với những vectorstore có quy mô lớn hoặc các doanh nghiệp cần sử dụng các tính năng nâng cao, chi phí dịch vụ đám mây có thể tăng cao đáng kể. Bên cạnh đó, quá trình thiết lập và quản lý vectorstore đòi hỏi người triển khai phải có kiến thức chuyên sâu về cơ sở dữ liệu và ngôn ngữ truy vấn GraphQL – điều này có thể trở thành rào cản đối với các nhóm phát triển không chuyên sâu về kỹ thuật hoặc thiếu tài nguyên nhân lực.

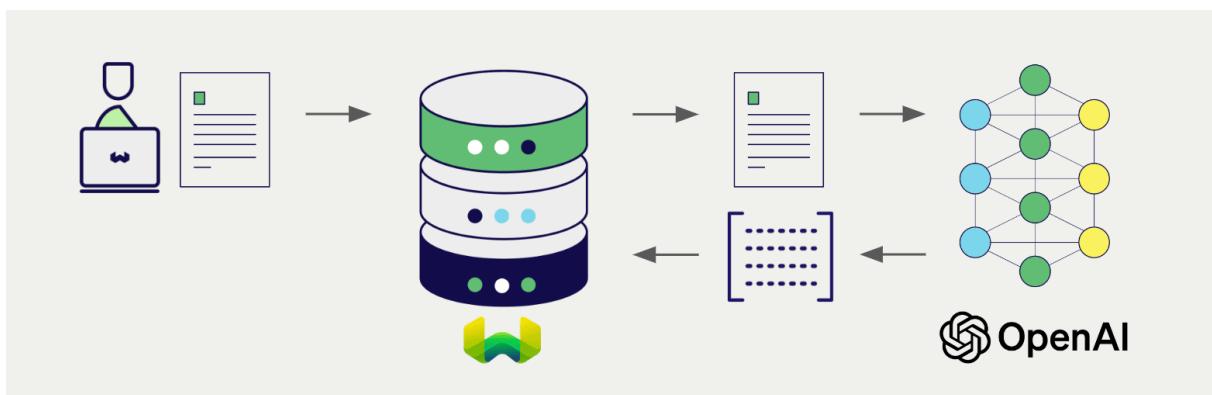
2.4.2. Lý do lựa chọn kiến trúc

Việc sử dụng WCD làm vector trên nền tảng đám mây giúp ChatBot dễ dàng triển khai trên nền tảng thực tế khi có thể cấp cho từng người dùng 1 phiên tương tác với Weaviate Client, cũng như dễ dàng chỉnh sửa các thông tin sản phẩm theo yêu cầu của doanh nghiệp mà không cần phải chạy lại chương trình.

2.4.3. Cách thức sử dụng

2.4.3.1. Xây dựng Weaviate vectorstore

Sau khi tiền xử lý dữ liệu, dữ liệu sẽ được vector hóa bằng mô hình OpenAI Embeddings và lưu vào Weaviate vectorstore trên Cloud. Ở đây, chỉ có nội dung của chunks là được vector hóa, còn những thông tin phụ trong metadata thì không cần, nhằm giảm chi phí embed, cũng như sử dụng các thuộc tính trong metadata để truy vấn CRUD.

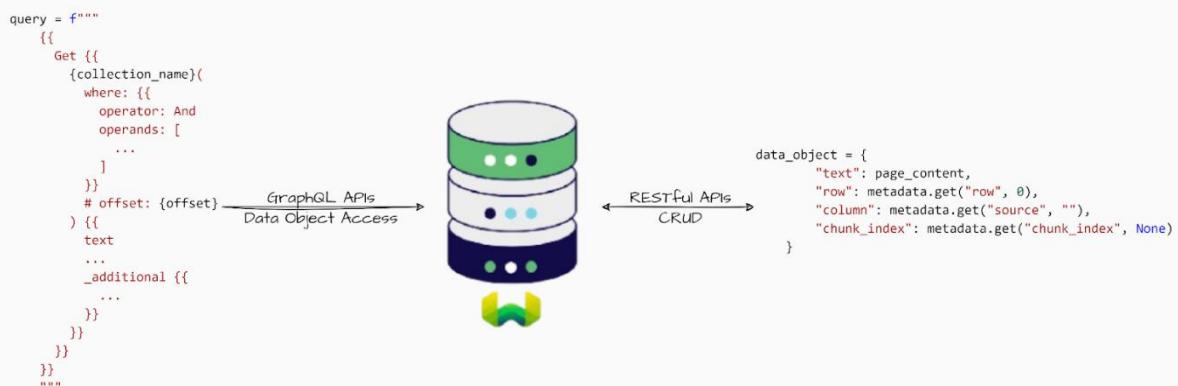


Hình 2.5: Kiến trúc Weaviate Cloud sử dụng dữ liệu sản phẩm để xây dựng vectorstore bằng OpenAI Embeddings (Nguồn:

<https://weaviate.io/developers/weaviate/model-providers/openai/embeddings>)

2.4.3.2. CRUD vectorstore bằng APIs

Ta sử dụng GraphQL API để truy xuất thông tin sản phẩm, lấy về các thông tin cần thiết để xem, cũng như thêm, chỉnh sửa và xóa các thông tin đó bằng RESTful API.



Hình 2.6: Mô hình truy xuất vectorstore bằng Weaviate RESTful API và GraphQL API

2.4.4. Úng dụng trong đề tài

Sử dụng toàn bộ dữ liệu sản phẩm của doanh nghiệp để lưu trữ dưới dạng vectorstore. Chatbot sẽ truy vấn các mẫu thông tin có liên quan nhất với câu hỏi của khách hàng. Đây là cơ chế cốt lõi nhất trong hệ thống tìm kiếm theo ngữ nghĩa mà không phải truy xuất từ khoá.

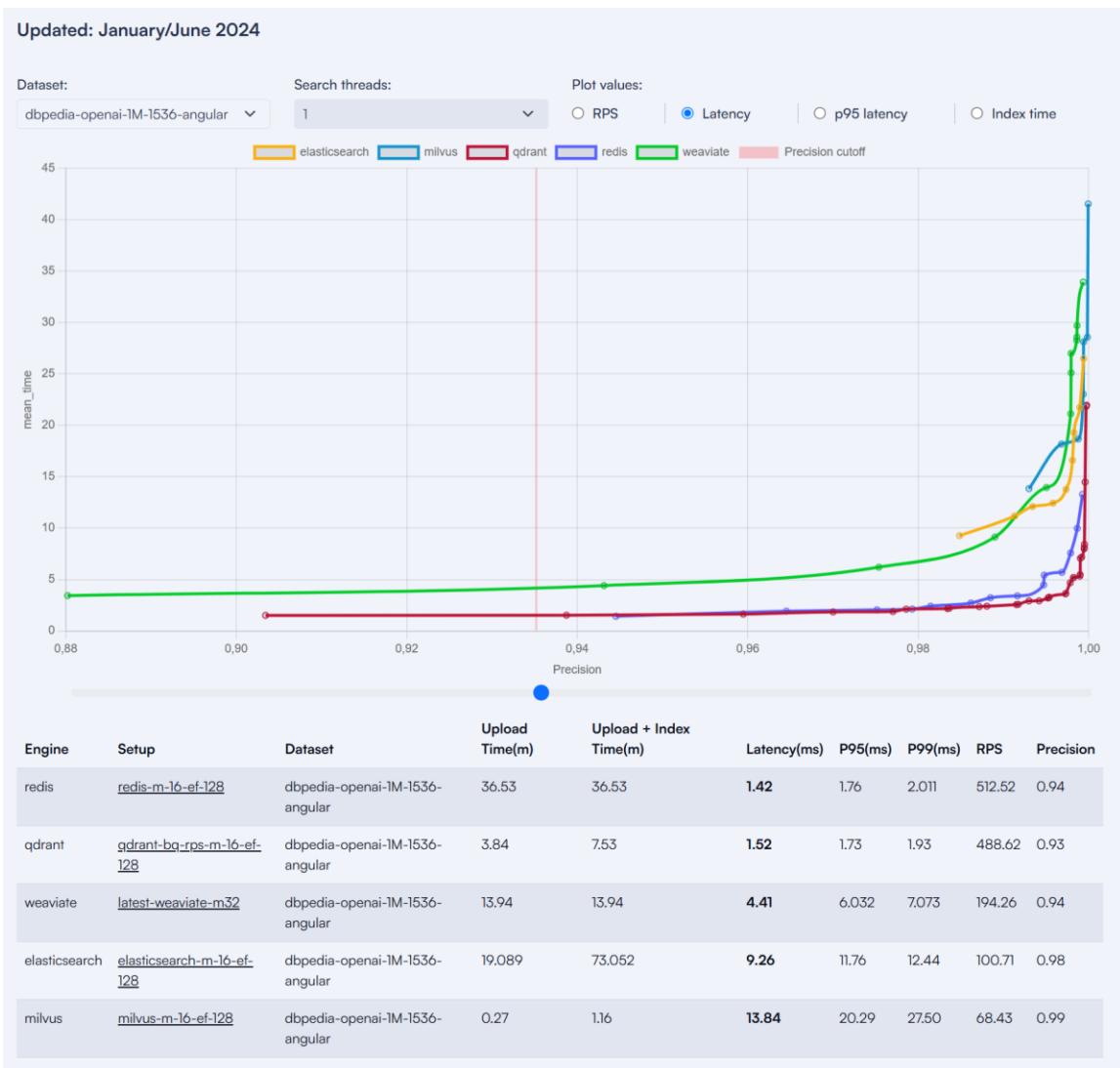
Ngoài ra, hệ thống còn sử dụng GraphQL và RESTful API để CRUD vectorstore, giúp cập nhật thông tin sản phẩm kịp thời.

2.5. Qdrant

2.5.1. Giới thiệu

Qdrant cũng là một vectorstore, tương tự như Weaviate, với khả năng hỗ trợ semantic search, metadata linh hoạt, được tối ưu bằng thuật toán kNN nhờ vào cấu trúc đồ thị Hierarchical Navigable Small World (HNSW). Ngoài ra, Qdrant còn hỗ trợ lưu trữ và truy xuất song song, RESTful APIs, và tính mở rộng linh hoạt với Qdrant Cloud. Các đặc điểm này giúp Qdrant phù hợp với khóa luận, cũng như các định hướng phát triển sau này.

Qdrant là một trong những nền tảng vectorstore mã nguồn mở hiệu suất cao, được đánh giá cao về khả năng truy xuất dữ liệu nhanh chóng và chính xác.



Hình 2.7: Biểu đồ thể hiện độ trễ trung bình (ms) so với độ chính xác của Redis, Qdrant, Weaviate, Elasticsearch và Milvus trên tập dbpedia-openai-1M-1536-angular (cập nhật 06/2024) [13]

Hơn nữa, Qdrant hỗ trợ triển khai linh hoạt trên nền tảng đám mây thông qua dịch vụ Qdrant Cloud, cho phép người dùng dễ dàng mở rộng quy mô hệ thống mà không phải đầu tư hạ tầng vật lý. Điểm đáng chú ý là hệ thống còn cung cấp API đa dạng để thực hiện thao tác CRUD (Create, Read, Update, Delete) một cách liền mạch, giúp đảm bảo dữ liệu được cập nhật nhanh chóng mà không gây gián đoạn đến toàn bộ hệ thống đang vận hành.

Tuy nhiên, Qdrant cũng có một số hạn chế khi áp dụng trong các ứng dụng cụ thể như ChatBot tư vấn sản phẩm. Dù có hiệu năng cao trong truy vấn ngữ nghĩa, Qdrant lại thiếu sự hỗ trợ đối với các chức năng tìm kiếm lai (hybrid search), vốn đóng vai trò quan trọng trong việc kết hợp dữ liệu có cấu trúc và phi cấu trúc. Đồng thời, nền tảng

này hiện chưa hỗ trợ GraphQL, gây ra một số khó khăn trong việc truy vấn các thuộc tính chi tiết hoặc kết hợp dữ liệu phức tạp – điều mà các hệ thống như Weaviate làm tốt hơn trong cùng ngữ cảnh.Thêm vào đó, chi phí mở rộng khi sử dụng phiên bản Hybrid Cloud cũng cần được cân nhắc, với mức giá vào khoảng 0.014 USD/giờ, có thể trở thành gánh nặng đối với các doanh nghiệp nhỏ và vừa nếu không có kế hoạch tối ưu tài nguyên rõ ràng [14].

2.5.2. Lý do lựa chọn kiến trúc

Qdrant gần như có đầy đủ các chức năng của Weaviate, với có tốc độ truy vấn nhanh hơn nhiều, nhưng không tối ưu với tệp thông tin sản phẩm lớn. Vì thế, Qdrant tối ưu cho việc truy xuất các thông tin ngữ cảnh, nhờ vào khả năng tìm kiếm ngữ nghĩa tốt và bộ lọc metadata mạnh mẽ, hỗ trợ chatbot phản hồi cá nhân hóa hơn, thay vì trực tiếp cung cấp thông tin sản phẩm cho chatbot.

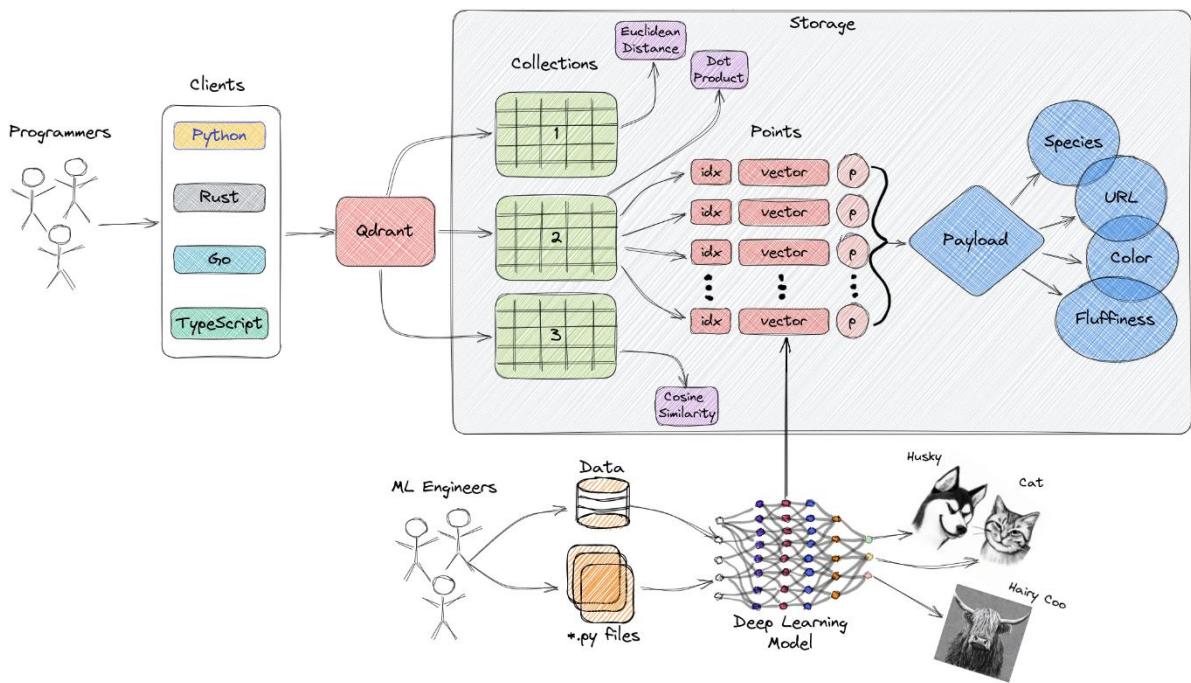
2.5.3. Cách thức hoạt động

Các thông tin ngữ cảnh sẽ được lưu vào các collections – tập hợp các điểm (points) có cùng số chiều dữ liệu (dimensions) và độ đo (Dot Product, Cosine,...) để chúng so sánh độ tương đồng với câu truy vấn. Điều này giúp lập trình viên cấu hình từng collection để tối ưu cho từng loại tìm kiếm hơn.

Với mỗi point, chúng bao gồm các vector, được vector hóa bằng OpenAI Embedding, để đại diện cho từng dữ liệu, và payload – một dạng metadata để phục vụ cho việc gán nhãn cho dữ liệu.

Các collections được lưu trữ bằng một trong hai cách, gồm RAM – có tốc độ truy xuất nhanh nhất, hoặc bộ nhớ Memmap – lập chỉ mục trên không gian RAM ảo, cho phép truy cập với tốc độ tương đương nhưng tiết kiệm bộ nhớ hơn. Nhờ vào cơ chế lưu trữ này, độ trễ trong việc truy xuất giảm nhiều hơn so với vectorstore của Weaviate.

Khi hệ thống cần truy xuất dữ liệu từ câu truy vấn, nếu như Weaviate trả về chunks – từng mảnh dữ liệu phù hợp nhất, thì Qdrant trả về points, và có thể áp dụng bộ lọc cho payload.



Hình 2.8: Tổng quan kiến trúc của Qdrant vectorstore (Nguồn: <https://qdrant.tech/documentation/overview/>)

2.5.4. Ứng dụng trong đề tài

Danh sách các business logic và user feedback sẽ được lưu vào hai collections của Qdrant. Câu truy vấn của người dùng sẽ được dùng để thu thập về các logic nghiệp vụ và phản hồi người dùng cần thiết, để chatbot phản hồi tốt hơn.

Ngoài ra, các thông tin được truy vết sẽ được lọc dựa trên bước hiện tại trong quy trình bán hàng được chatbot xác định, giúp giảm thời gian truy vấn, cũng như xác định đúng hơn các logic phù hợp với ngữ cảnh hiện tại.

2.6. Các công nghệ phụ

2.6.1. Redis - Giải pháp cache lịch sử chat

Redis (Remote Dictionary Server) là hệ thống lưu trữ dữ liệu theo dạng key-value tốc độ cao, theo thời gian thực, là lựa chọn phù hợp cho các hệ thống ChatBot bán hàng. Đây là công nghệ giúp lịch sử tin nhắn tạm thời (cache) của từng người dùng theo từng phiên, giúp ChatBot truy xuất nhanh chóng lịch sử tương tác riêng mỗi khách hàng.

Bên cạnh đó, Redis cũng là nơi lưu trữ quy trình bán hàng của ChatBot, giúp hệ thống truy xuất hướng dẫn tư vấn nhanh chóng của từng giai đoạn dành cho từng khách hàng.



Hình 2.9: Hệ thống lưu trữ cache Redis (Nguồn:
<https://iconduck.com/icons/13186/redis-original-wordmark>)

2.6.2. Chatwoot – Nền tảng quản lý hội thoại của khách hàng cho ChatBot

Chatwoot là nền tảng quản lý tin nhắn mã nguồn mở giữa khách hàng và doanh nghiệp, quản lý tập trung các kênh tương tác khác nhau như email, website live chat, mạng xã hội (Facebook, Zalo,...). Trong dự án này, Chatwoot được dùng làm frontend cho hệ thống ChatBot, giúp khách hàng tương tác với ChatBot dễ dàng hơn, cũng như nhanh chóng chuyển luồng trò chuyện với tư vấn viên khi cần thiết và Chatwoot API để liên kết các luồng tin nhắn với hệ thống Chatbot.



Hình 2.10: Nền tảng quản lý tin nhắn Chatwoot (Nguồn:
<https://dribbble.com/shots/10077118-Chatwoot-Logo-Animation>)

Chương 3. CHATBOT TƯ VẤN BÁN HÀNG

3.1. Giới thiệu bài toán

Trong bối cảnh AI ngày càng phát triển và dần trở thành “từ khóa SEO” quan trọng mà bất kỳ doanh nghiệp nào cũng muốn đưa vào trong nội dung Marketing của mình, việc ứng dụng ChatBot ngày càng trở thành công cụ không thể thiếu, không chỉ dành cho

việc tư vấn bán hàng, mà còn giúp hình ảnh doanh nghiệp bắt kịp với xu hướng hơn. Tuy nhiên, phần với các nền tảng ChatBot chỉ có khả năng phản hồi thụ động mà không thể dẫn dắt khách hàng theo bất kỳ quy trình bán hàng cụ thể nào, dẫn đến trải nghiệm người dùng rời rạc, thiếu hiệu quả, và không thể giúp tăng tỷ lệ chuyển đổi trong bán hàng, từ đó gây lãng phí ngân sách, suy giảm sự tin tưởng vào AI của doanh nghiệp.

Bài toán đặt ra là phải thiết kế một ChatBot có khả năng hướng dẫn người dùng đi qua từng bước trong quy trình bán hàng (có thể nhảy bước nếu thông tin khách hàng đã đáp ứng các bước trước đó), từ đó giúp kéo dài sự tương tác giữa người và ChatBot, thu thập thông tin có hệ thống hơn, giúp tối ưu hóa trải nghiệm người dùng và nâng cao tỷ lệ chuyển đổi.

Bên cạnh đó, kiến trúc AI Agent được đề tài áp dụng. Hệ thống sẽ được chia thành nhiều Agents với những tác vụ khác nhau với chung bộ nhớ bên ngoài, giúp ChatBot không chỉ làm tốt các nghiệp vụ được đề ra, mà còn linh hoạt hơn trong việc tích hợp các chức năng khác (đặt hàng online, kiểm tra giao nhận hàng,...), giúp tăng cường trải nghiệm người dùng.

3.2. Kiến trúc hệ thống

Hệ thống ChatBot này được xây dựng dựa trên framework LangChain, sử dụng mô hình ngôn ngữ GPT-4o-mini để thực hiện nhiều tác vụ cùng lúc theo mô hình AI Agent dạng "tiếp sức". Mỗi thành phần trong hệ thống đảm nhiệm một vai trò riêng biệt, phối hợp chặt chẽ với nhau nhằm mục tiêu cuối cùng: hỗ trợ tư vấn bán hàng chính xác, tự nhiên và đúng theo quy trình chuẩn.

Các AI Agents chia sẻ chung bộ nhớ sự kiện (Episodic Memory) và cập nhật vào đó thông tin trả về của mỗi Agent, giúp cá nhân hóa thông tin phản hồi theo từng khách hàng dựa vào lịch sử chat, hồ sơ khách hàng và trạng thái hiện tại trong quy trình. Lịch sử chat và quy trình bán hàng được lưu bằng Redis và hồ sơ khách hàng trong từng phiên chat trả về từ Chatwoot API, ChatBot có thể vừa nắm bắt nhanh các tương tác gần nhất, vừa khai thác được kinh nghiệm tích lũy trong suốt quá trình giao tiếp với khách hàng.

Mỗi AI Agent trong hệ thống đảm nhận một nhiệm vụ riêng, theo mô hình "tiếp sức" liên tục giữa các tác tử:

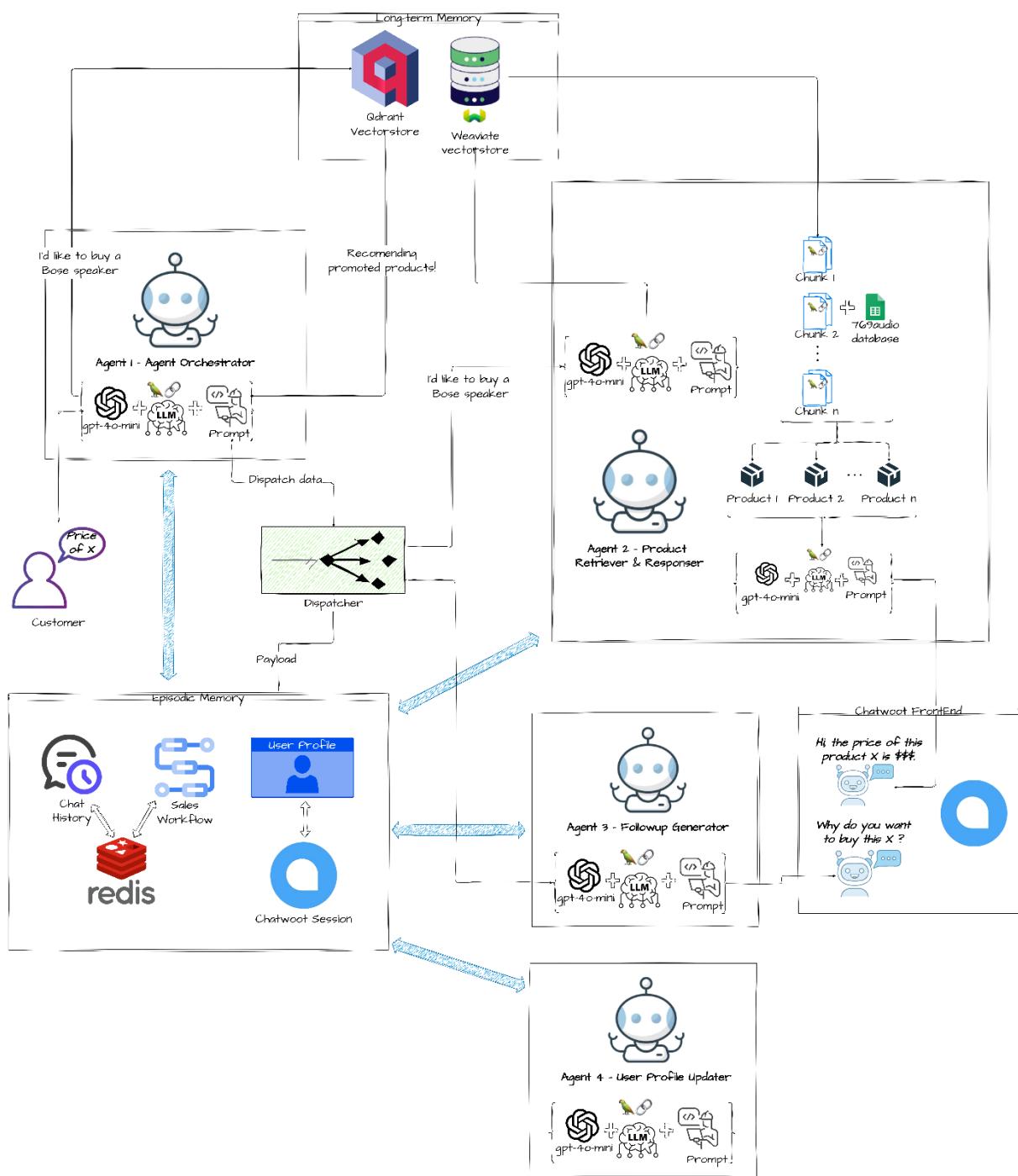
Agent 1 – Agent Orchestrator: Phân tích câu hỏi đầu vào, xác định giai đoạn tương ứng trong quy trình bán hàng (ví dụ: Greeting, Needs Assessment, Qualification...), truy xuất các nghiệp vụ bán hàng (Business Logic) và phản hồi của người dùng (User Feedback) thông qua Qdrant Vectorstore, sau đó quyết định gọi Agent 2 hoặc Agent 3 tiếp theo.

Agent 2 - Product Retriever & Responser: Chịu trách nhiệm tìm kiếm và tổng hợp thông tin sản phẩm từ Weaviate Vectorstore, trả lời kèm theo link và hình ảnh sản phẩm phù hợp.

Agent 3 - Followup Generator: Tạo ra các câu hỏi hoặc nội dung gợi mở tiếp theo dựa trên quy trình bán hàng, nhằm dẫn dắt khách hàng đi qua các giai đoạn kế tiếp.

Agent 4 - User Profile Updater: Tự động cập nhật hồ sơ người dùng dựa trên những gì khách hàng đã chia sẻ trong suốt phiên trò chuyện.

Kiến trúc này mang lại nhiều điểm mạnh đáng kể. Trước hết, nó bám sát thực tế quy trình tư vấn bán hàng tiêu chuẩn, do đó dễ dàng áp dụng vào các môi trường kinh doanh cụ thể mà không cần thay đổi nhiều về mặt nghiệp vụ. Thứ hai, nhờ thiết kế mở và phân tách tác vụ rõ ràng, hệ thống có thể linh hoạt thay thế mô hình GPT-4o-mini bằng các mô hình mới hơn, tối ưu chi phí hơn như DeepSeek API,... Cuối cùng, cấu trúc này còn hỗ trợ khả năng mở rộng theo chiều sâu – các doanh nghiệp có thể bổ sung thêm các Agent xử lý các tác vụ mới như Agent 5 cho thiết lập đơn hàng, mà không cần can thiệp vào các thành phần đã hoạt động ổn định.

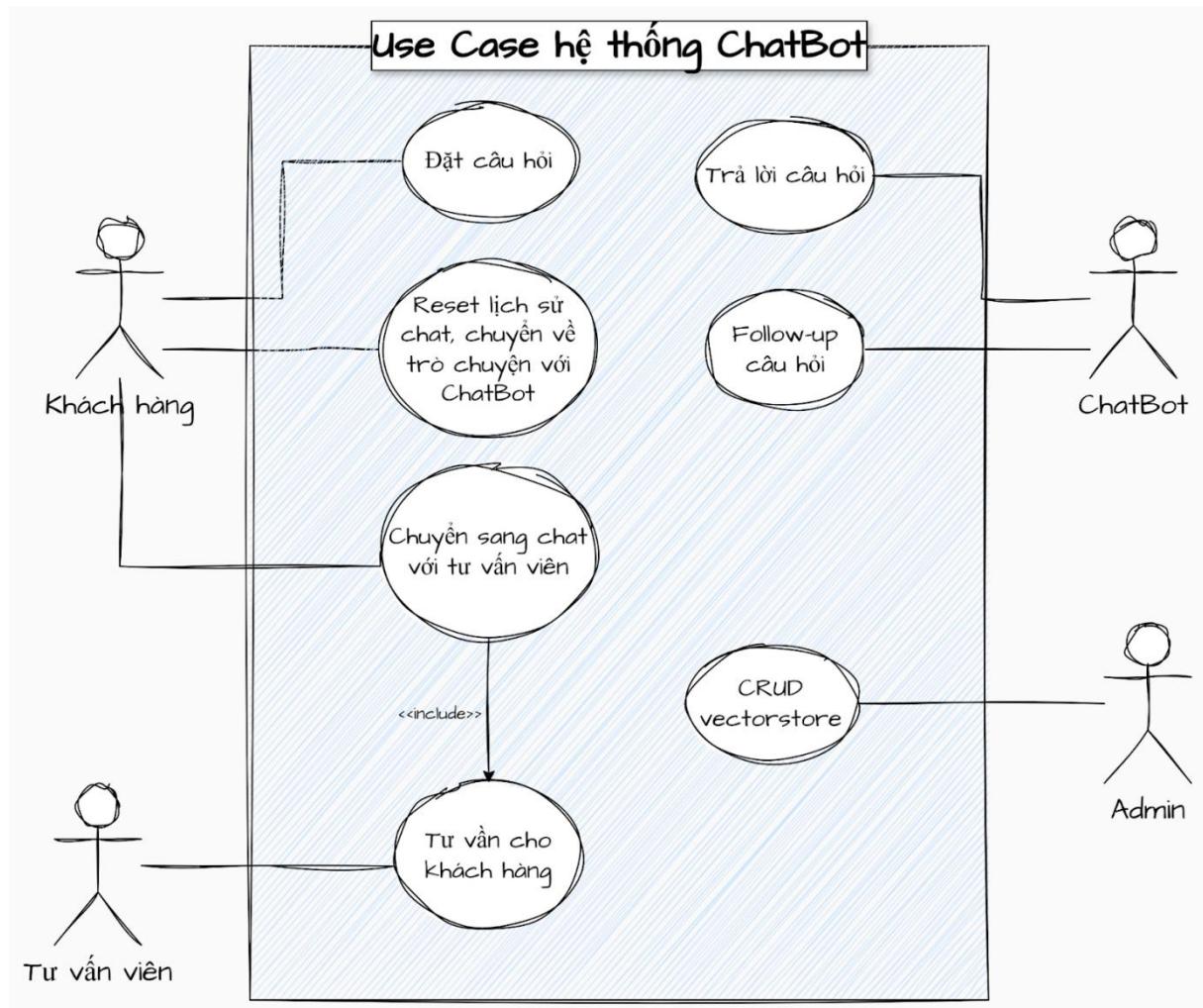


Hình 3.1: Kiến trúc tổng thể của ChatBot tư vấn bán hàng

3.3. Phân tích hệ thống

3.3.1. Lược đồ Use Case hệ thống ChatBot

Lược đồ sau đây trình bày các actors và các nghiệp vụ có liên quan trong hệ thống ChatBot:



Hình 3.2: Sơ đồ Use case hệ thống ChatBot

Hệ thống ChatBot được xây dựng trong khuôn khổ đề tài bao gồm ba nhóm tính năng chính, nhằm đáp ứng các yêu cầu nghiệp vụ thực tiễn trong quá trình tư vấn và hỗ trợ bán hàng tự động.

Thứ nhất, **tính năng tư vấn bán hàng qua hội thoại** cho phép người dùng đặt câu hỏi trực tiếp với ChatBot về các sản phẩm hoặc dịch vụ của doanh nghiệp. ChatBot không chỉ trả lời các truy vấn cụ thể, mà còn có khả năng đặt các câu hỏi tiếp theo (follow-up) nhằm khai thác thêm nhu cầu của khách hàng, dựa trên ngữ cảnh và vị trí hiện tại của họ trong quy trình bán hàng. Ngoài ra, người dùng có thể đặt các câu hỏi liên quan khác, và ChatBot được yêu cầu phải phản hồi một cách nhất quán dựa trên lịch sử trò chuyện đang diễn ra. Trong trường hợp cần thiết, khách hàng có thể chủ động khởi tạo lại toàn bộ phiên tương tác bằng cách sử dụng nút "reset", giúp làm mới lịch sử chat và bắt đầu lại quy trình từ đầu.

Thứ hai, **tính năng chuyển tiếp cuộc hội thoại sang tư vấn viên thật** cho phép hệ thống linh hoạt can thiệp hỗ trợ khi cần thiết. Cụ thể, nếu người dùng nhập câu lệnh “/tuvanvien”, ChatBot sẽ tự động chuyển phiên trò chuyện hiện tại đến một tư vấn viên có sẵn. Toàn bộ lịch sử tương tác trước đó và thông tin hồ sơ khách hàng (được tổng hợp bởi hệ thống) sẽ được truyền đến tư vấn viên, giúp đảm bảo sự liên tục trong hỗ trợ và tối ưu hóa trải nghiệm người dùng trong các tình huống cần xử lý chuyên sâu.

Cuối cùng, hệ thống còn tích hợp **tính năng CRUD vectorstore**, dành riêng cho quản trị viên (Admin). Thông qua các giao diện quản trị hoặc truy vấn API, Admin có thể thực hiện các thao tác tạo mới, chỉnh sửa, xóa hoặc cập nhật dữ liệu sản phẩm. Dữ liệu này sẽ được lưu trữ dưới dạng vector trong các hệ thống như Weaviate, nhằm phục vụ hiệu quả cho các truy vấn ngữ nghĩa (semantic search) của ChatBot trong quá trình tư vấn. Đây là thành phần thiết yếu giúp đảm bảo độ chính xác, tính cập nhật và độ phản hồi kịp thời cho các câu hỏi liên quan đến sản phẩm trong hệ thống.

3.3.2. Đặc tả Use Case:

3.3.2.1. Đặc tả Use Case Chat tư vấn bán hàng:

Bảng 3.1: Bảng đặc tả Use Case Chat tư vấn bán hàng

Tên use case	Chat tư vấn bán hàng
Mô tả	Trò chuyện và tư vấn về sản phẩm giữa khách hàng và ChatBot
Actor chính	Khách hàng
Actor phụ	ChatBot
Tiền điều kiện	Không có
Hậu điều kiện	Không có
Luồng hoạt động	<ol style="list-style-type: none"> 1. Khách hàng hỏi bot câu hỏi 2. Bot trả lời câu hỏi của khách hàng 3. Bot đưa ra câu dẫn dắt phù hợp cho khách hàng 4. Lặp lại bước 1 và lưu lại lịch sử chat để bot trả lời đúng

	theo luồng trò chuyện
Luồng thay thế	Nếu khách hàng bấm nút reset, lịch sử trò chuyện sẽ được xóa và bắt đầu lại bước 1
Luồng ngoại lệ	Không có

3.3.2.2. Đặc tả use case Chuyển phiên chat cho tư vấn viên:

Bảng 3.2: Bảng đặc tả Use Case Chuyển phiên chat cho tư vấn viên

Tên use case	Chuyển phiên chat cho tư vấn viên
Mô tả	Khách hàng có thể sử dụng lệnh “/tuvanvien” để yêu cầu chuyển cuộc hội thoại từ chatbot sang tư vấn viên. Hệ thống sẽ tự động kết nối khách hàng với tư vấn viên phù hợp, cung cấp lịch sử chat và thông tin khách hàng đã được chatbot tổng hợp. Tư vấn viên có thể tiếp tục cuộc trò chuyện dựa trên ngữ cảnh trước đó.
Actor chính	Khách hàng
Actor phụ	Tư vấn viên
Tiền điều kiện	<ol style="list-style-type: none"> Khách hàng đang trong cuộc trò chuyện với chatbot. Hệ thống có ít nhất một tư vấn viên sẵn sàng tiếp nhận cuộc hội thoại.
Hậu điều kiện	<ol style="list-style-type: none"> Cuộc hội thoại được chuyển thành công từ chatbot sang tư vấn viên. Tư vấn viên có quyền truy cập vào lịch sử chat và thông tin khách hàng để tiếp tục hỗ trợ.
Luồng hoạt động	<ol style="list-style-type: none"> Khách hàng nhập “/tuvanvien” vào khung chat. Chatbot tiếp nhận yêu cầu và xác nhận rằng khách hàng

	<p>muốn chuyển đổi.</p> <ol style="list-style-type: none"> 3. Hệ thống tìm kiếm tư vấn viên đang trực tuyến và sẵn sàng tiếp nhận cuộc hội thoại. 4. Hệ thống kết nối khách hàng với tư vấn viên. 5. Hệ thống gửi lịch sử chat và thông tin khách hàng đến tư vấn viên. 6. Tư vấn viên tiếp nhận và bắt đầu trò chuyện với khách hàng. 7. Cuộc hội thoại tiếp tục cho đến khi khách hàng kết thúc hoặc yêu cầu hỗ trợ khác.
Luồng thay thế	Không có
Luồng ngoại lệ	Không có

3.3.2.3. Đặc tả Use Case CRUD vectorstore:

Bảng 3.3: Bảng đặc tả Use Case CRUD vectorstore

Tên use case	CRUD vectorstore
Mô tả	CRUD các thông tin cần thiết của sản phẩm ứng với thông tin thực tế của sản phẩm
Actor chính	Admin
Actor phụ	Không có
Tiền điều kiện	Không có
Hậu điều kiện	Không có
Luồng hoạt động	Admin lựa chọn CRUD sản phẩm mới hoặc các sản phẩm có sẵn

Luồng thay thế	Không có
Luồng ngoại lệ	Không có

3.4. Thiết kế hệ thống

3.4.1. Tổng quan kiến trúc AI Agents

3.4.1.1. Kiến trúc chung của các AI Agents trong ChatBot

Trong bối cảnh của khóa luận này, kiến trúc của AI Agents sẽ được đơn giản hóa để phù hợp với tác vụ của từng Agent, nhằm phục vụ tốt nhất cho ChatBot - vốn không phải là AI Agents (non-agentic agent) mà chỉ là hệ thống tập hợp các Agents (multi-agent system).

Các Agents trong ChatBot được thiết kế để phục vụ riêng cho các nghiệp vụ dành riêng cho ChatBot, tức hoạt động trong môi trường có phạm vi hẹp hơn, không phải là hệ thống độc lập hoàn chỉnh (standalone systems). Chính vì vậy, khi thiết kế các Agents, ta cần tập trung vào sự tối giản của kiến trúc, tốc độ xử lý (thời gian để phản hồi cho người dùng vẫn là yếu tố tối quan trọng, và khả năng phối hợp, “tiếp sức” một cách tối ưu nhất giữa các Agents.

3.4.1.2. Phân tích chi tiết các AI Agents trong ChatBot

(a) Các input chung cho các Agents

Bảng 3.4: Các thông tin đầu vào (input) dùng chung cho các AI Agents

Input	Mô tả
Lịch sử chat (<i>chat_history</i>)	Là lịch sử tin nhắn giữa người dùng và ChatBot, bao gồm: <ul style="list-style-type: none"> • Câu hỏi được diễn giải • Phản hồi của ChatBot • Câu followup của ChatBot
Hồ sơ khách hàng (<i>user_profile</i>)	Đoạn mô tả khách hàng dựa vào lịch sử chat, mặc định là rỗng khi khách hàng chat lần đầu

Input	Mô tả
Quy trình bán hàng (<i>workflow_stages</i>)	Là các bước của quy trình bán hàng mà Chatbot phải tuân theo để dẫn dắt khách hàng
Giai đoạn quy trình hiện tại (<i>workflow_stage</i>)	Là giai đoạn hiện tại của khách hàng trong quy trình bán hàng xác định
Giai đoạn quy trình dự kiến (<i>predicted_stage</i>)	Là bước tiếp theo mà Chatbot dự đoán để đưa ra phản hồi dẫn dắt khách hàng

Trong thiết kế hệ thống ChatBot định hướng theo quy trình bán hàng, việc phân biệt rõ ràng giữa **giai đoạn hiện tại** và **giai đoạn dự kiến** đóng vai trò then chốt trong đảm bảo tính linh hoạt và tính chiến lược của hội thoại. Mỗi phiên trò chuyện có thể phát sinh những tình huống không nằm trong lộ trình kịch bản dự đoán trước, do đó hệ thống cần vừa phản ứng tức thời, vừa giữ được định hướng tổng thể đã được thiết kế.

Cụ thể, **giai đoạn hiện tại** được hiểu là trạng thái thực tế của cuộc hội thoại tại thời điểm người dùng đưa ra một phản hồi cụ thể, phản ánh đúng tâm lý hoặc vấn đề họ đang gặp phải. Trong khi đó, **giai đoạn dự kiến** là bước tiếp theo trong kịch bản bán hàng mà hệ thống đang hướng đến, thể hiện mục tiêu chiến lược cần đạt được sau khi giải quyết được các trở ngại phát sinh.

Chẳng hạn, sau khi ChatBot đã hoàn tất việc trình bày thông tin sản phẩm, theo kịch bản định sẵn, bước kế tiếp sẽ là **Chốt đơn hàng** bằng cách đặt câu hỏi như “Anh/Chị có muốn đặt mua sản phẩm này không?”. Tuy nhiên, nếu khách hàng phản hồi bằng một lời từ chối gián tiếp như “Giá cao quá!”, thì lúc này, ChatBot buộc phải chuyển sang **giai đoạn hiện tại** là “**Xử lý từ chối**”, tức cần tập trung giải đáp băn khoăn, có thể là đề xuất các chương trình ưu đãi hoặc giới thiệu lựa chọn thay thế phù hợp với ngân sách hơn. Trong khi đó, **giai đoạn dự kiến vẫn được giữ nguyên là “Chốt đơn”**, với mục tiêu quay lại kịch bản ban đầu sau khi đã giải quyết rào cản về giá cả.

Việc xác lập đồng thời hai giai đoạn – hiện tại và dự kiến – giúp ChatBot xử lý chính xác tình huống phát sinh mà vẫn đảm bảo mạch dẫn chiến lược không bị đứt gãy. Đây là một cơ chế đặc biệt quan trọng trong các hệ thống AI định hướng mục tiêu, giúp cân bằng giữa tính phản ứng linh hoạt và tính định hướng dài hạn trong quy trình bán hàng tự động.

(b) Các input đặc trưng của các AI Agents

Với mỗi Agent, thì sẽ có những tác vụ, input và output khác nhau cho từng nghiệp vụ của riêng nó:

Bảng 3.5: Các thông tin đầu vào (input) dùng chung cho các AI Agents

STT	Agent	Mục đích / Vai trò	Đầu vào đặc trưng	Đầu ra đặc trưng	Vectorstore
1	Agent Orchestrator	<ul style="list-style-type: none"> Chuẩn sửa chính tả cho câu truy vấn Xác định <i>giai đoạn hiện tại</i> của quy trình bán hàng Sinh <i>action list</i> (gọi Agent 2 hoặc 3) 	<ul style="list-style-type: none"> Câu hỏi thô của khách hàng Giai đoạn quy trình dự kiến 	<ul style="list-style-type: none"> Câu truy vấn đúng chính tả Giai đoạn quy trình hiện tại – Danh sách actions ở dạng JSON 	Qdrant - để lấy nghiệp vụ bán hàng (<i>business logic</i>) & phản hồi người dùng (<i>user feedback</i>)
2	Product Retriever & Responder	<ul style="list-style-type: none"> Chuẩn hóa câu truy vấn đúng chính tả của Agent 1 Truy vấn, tùy chỉnh chunks thành các sản phẩm liên quan Tạo ra câu phản hồi 	<ul style="list-style-type: none"> Câu truy vấn đúng chính tả của Agent 1 Business Logic User Feedback 	Câu trả lời - AI answer cho người dùng	Weaviate - chứa vector sản phẩm
3	Follow-up Generator	<ul style="list-style-type: none"> Đánh giá AI answer vừa gửi có cần hỏi tiếp không Nếu cần, dự đoán bước tiếp theo và đặt câu hỏi 	<ul style="list-style-type: none"> Câu truy vấn đúng chính tả của Agent 1 AI answer của Agent 2 Business Logic User Feedback 	<ul style="list-style-type: none"> Bước tiếp theo trong quy trình Câu followup - AI follow up cho người dùng 	Không có

		follow up để gọi mở khách hàng theo quy trình bán hàng			
4	User Profile Updater	Cập nhật lại thông tin khách hàng cho các lần/phiên chat khác	<ul style="list-style-type: none"> • Câu truy vấn đúng chính tả của Agent 1 • AI answer của Agent 2 • AI follow up của Agent 3 • Bước tiếp theo trong quy trình 	Hồ sơ khách hàng mới	Không có

3.4.1.3. Quy trình bán hàng cho ChatBot

Trong hệ thống ChatBot, quy trình bán hàng đóng vai trò then chốt để tách biệt mình với những ChatBot truyền thống. Quy trình bán hàng giúp định hướng luồng trao đổi giữa khách hàng và ChatBot, dẫn dắt người dùng bằng các câu followup theo quy chuẩn nhất định, thay vì chỉ bị động trả lời câu hỏi mà người dùng đặt ra. Người dùng có thể thay đổi quy trình bán hàng, nội dung prompt, và các điều kiện tùy theo quy trình bán hàng thực tế của doanh nghiệp.

Quy trình mẫu được thiết kế trong hệ thống được biểu diễn dưới dạng cấu trúc dữ liệu JSON, với một ví dụ như sau:

```

"Greeting": {
    "internal_instruction": (
        "Chào hỏi khách hàng thân thiện, giới thiệu dịch vụ/sản
phẩm",
        "và mời họ chia sẻ nhu cầu hoặc thắc mắc."
    ),
    "sample_user_message": (
        "Chào anh/chị! Em có thể hỗ trợ anh/chị về thiết bị âm
thanh gì hôm nay ạ?"
    ),
    "identify": (

```

```

        "Tin nhắn chỉ chào hỏi (hello, hi, chào...), không chứa
từ khoá sản phẩm, "
    "thường ≤ 10 từ."
),
},
...

```

Trong đó:

Greeting (Tên giai đoạn): Đại diện cho tên một giai đoạn nhất định, được đặt trong trường key của dữ liệu từ điển.

internal_instruction (Hướng dẫn): Hướng dẫn điều nên làm cho quy trình.

sample_user_message (Câu mẫu): Câu followup mẫu cho ChatBot tham khảo.

identify (Nhận biết): Cách nhận biết quy trình hiện tại dựa vào query của người dùng.

3.4.2. Xây dựng các AI Agents

3.4.2.1. Agent 1 - Làm Rõ Câu Hỏi & Nhận Diện Giai Đoạn

Hãy bắt đầu với kết quả trả về của Agent:

```
{
  "query_and_stage": {
    "semantic_query": "<câu hỏi chuẩn hóa để embed retrieval>",
    "workflow_stage": "<tên giai đoạn>",
    "reason": "<lý do chọn giai đoạn>",
    "answer": "<nội dung trả lời ngay nếu greeting/simple; ngược
lại null>"
  },
  "actions": [
    {
      "agent": "<Agent2 hoặc Agent3 hoặc cả hai>",
      "task": "<nhiệm vụ cụ thể>",
      "payload": {
        "hint": "<nội dung để Agent2 hoặc Agent3 xử lý>"
      }
    }
  ]
  // Có thể trông nếu không gọi Agent nào
}
```

Bảng 3.6: Giải thích ý nghĩa của đối tượng JSON được trả về của Agent 1

Key	Value
query_and_stage	<ul style="list-style-type: none"> • semantic_query: Câu hỏi gốc của người dùng được sửa lỗi chính tả • workflow_stage: Bước bán hàng bot xác định dựa vào câu hỏi của khách hàng • reason: lý do chọn bước bán hàng ở trên – dành cho debug • answer: trả lời ngay nếu như câu hỏi đơn giản – trả null khi cần các Agents khác.
actions	<p>Đây là phần tử chính để ra lệnh gọi các Agent 2 hoặc Agent 3</p> <ul style="list-style-type: none"> • agent: tên của Agent được ra lệnh • task và payload: các khóa bổ sung để hướng dẫn Agents
retrieval_context	<p>Đây là kết quả truy vấn từ Qdrant Vectorstore để lấy về:</p> <ul style="list-style-type: none"> • business_logic: các nghiệp vụ doanh nghiệp cần thiết cho truy vấn • user_feedback: những phản hồi của người dùng trước có liên quan đến câu truy vấn

Agent 1 là thành phần thể hiện rõ nét nhất đặc trưng của một AI Agent hiện đại trong hệ thống ChatBot. Dù chỉ sử dụng mô hình GPT-4o-mini - tức đánh đổi khả năng suy luận để đạt tốc độ phản hồi nhanh nhất, Agent 1 vẫn được xây dựng để đảm nhận nhiều vai trò quan trọng.

Trước hết, Agent 1 có chức năng **phân tích ngữ cảnh hội thoại**, thông qua việc diễn giải nội dung câu hỏi của người dùng, xác định ý định ngôn ngữ và liên kết thông tin với quy trình bán hàng đã định sẵn. Trên cơ sở đó, hệ thống có thể xác định chính xác **giai đoạn hiện tại** của khách hàng trong phiếu bán hàng – chẳng hạn như đang ở bước chào hỏi, khai thác nhu cầu hay xử lý từ chối.

Bên cạnh đó, Agent 1 có khả năng **khai thác bộ nhớ ngoài**, cụ thể là thực hiện truy vấn đến cơ sở dữ liệu ngữ nghĩa Qdrant Vectorstore để tìm kiếm các thông tin nghiệp vụ (business logic) và phản hồi người dùng (user feedback) có liên quan. Việc kết hợp giữa phân tích ngữ cảnh hiện tại và dữ liệu truy xuất giúp gia tăng độ chính xác cũng như mức độ phù hợp của các hành động được đề xuất tiếp theo.

Đặc biệt, Agent 1 còn thực hiện vai trò **điều phối và phân phối nhiệm vụ** cho các Agent kế tiếp. Dựa vào nội dung đã phân tích, Agent này sẽ quyết định nên kích hoạt Agent 2 (phản hồi sản phẩm) hoặc Agent 3 (tạo câu hỏi follow-up), hoặc cả hai, nhằm duy trì dòng chảy hội thoại liền mạch và định hướng khách hàng theo đúng quy trình bán hàng.

Với thiết kế theo mô hình “**orchestrator**”, Agent 1 không chỉ là một thành phần xử lý logic đơn thuần mà còn đóng vai trò trung tâm điều phối tổng thể cho toàn bộ hệ thống. Nó đảm bảo rằng ChatBot có thể phản ứng một cách **linh hoạt, ngẫu hứng hóa và có chiến lược**, từ đó nâng cao hiệu quả tương tác và khả năng chuyển đổi trong quá trình tư vấn bán hàng quan trọng.

Prompt Engineering:

Nguyên tắc quyết định gọi Agent

1. **Nếu câu hỏi đơn giản / chào hỏi** (ví dụ: "hello", "chào bạn") hoặc giai đoạn là "Greeting":

- Không cần gọi Agent2 hay Agent3.
- Trả lời chào hỏi lịch sự.
- Gán `workflow_stage = "Greeting"`.

2. **Gọi Agent2** nếu:

- Khách hàng hỏi về sản phẩm, thương hiệu, tính năng, so sánh, giá cả.

- Ví dụ: "loa karaoke nào hay?", "micro nào cho sân khấu?", "loa nào rẻ?"

3. **Gọi Agent3** trong hầu hết trường hợp:

- Luôn kèm Agent3 để sinh follow-up, trừ khi câu trả lời trước đó đã kết thúc bằng một câu hỏi mở.

- Nếu số lượt trò chuyện dưới 2-3 vòng, Agent3 nên hỏi thêm thông tin nhu cầu.

- Nếu trên 2-3 vòng, Agent3 nên chuyển dần sang tư vấn sản phẩm cụ thể.

4. **Chỉ gọi Agent3 (không gọi Agent2)** khi:

- Khách bày tỏ từ chối, do dự (ví dụ: "đắt quá", "để suy nghĩ", "hỏi vợ đã") mà không hỏi về sản phẩm mới.

Agent 1 xác định cách phân bổ tác vụ dựa trên nội dung câu hỏi và ngữ cảnh hội thoại:

Trường hợp đơn giản (chào hỏi, hỏi thăm): Agent 1 sẽ tự động phản hồi lại mà không gọi sự giúp đỡ của các Agents khác, tương tự với hoạt động thực tế, khi việc chào hỏi không yêu cầu hiểu biết về sản phẩm/dịch vụ cần bán.

Trường hợp liên quan đến sản phẩm, tính năng, giá cả: Agent 1 sẽ gọi Agent 2 để truy xuất thông tin sản phẩm phù hợp và trả lời câu hỏi cho người dùng.

Trường hợp cần tiếp tục dẫn dắt khách hàng: Agent 1 sẽ gọi thêm Agent 3 để đặt câu hỏi follow-up hoặc thuyết phục mua hàng.

Trường hợp khách hàng bày tỏ từ chối hoặc do dự: Agent 1 chỉ gọi Agent 3 để xử lý phản hồi, không cần tìm kiếm sản phẩm mới.

Việc phân bổ này được hướng dẫn chi tiết trong phần Prompt thiết kế cho Agent 1, đảm bảo rằng chatbot có thể ứng xử linh hoạt, tự nhiên theo từng bối cảnh giao tiếp.

3.4.2.2. Agent 2 - Tìm Kiếm Thông Tin và Phản Hồi

(a) Tối ưu ký tự cho phân mảnh dữ liệu (data splitting)

Do trong dữ liệu huấn luyện (training data), thuộc tính chiếm nhiều thông tin nhất của một sản phẩm là Nội dung và Mô tả chi tiết. Vì thế, khi truy vết, đa phần các thông tin được thu thập sẽ thuộc về hai thuộc tính trên. Như vậy, hệ thống cần phải tối ưu số ký tự của mỗi chunk sao cho mỗi chunk sẽ chứa số ký tự trung bình của một câu, nhằm đảm bảo độ liền mạch của một câu và giữ lại thông tin đặc trưng của sản phẩm có độ dài lớn hơn các cụm từ bình thường.

Tính toán số ký tự trung bình mỗi câu của hai cột Nội dung và Mô tả chi tiết:

Trung bình số ký tự mỗi câu của cột Nội dung **67.33**

Trung bình số ký tự mỗi câu của cột Mô tả chi tiết: **63.25**

Dựa vào 2 số liệu trên, ta chọn chunk size là giá trị trung bình của hai số là 65:

```
text_splitter = RecursiveCharacterTextSplitter(chunk_size=65,
chunk_overlap=16)
```

(b) Xây dựng Weaviate Vectorstore

(i) Tinh chỉnh dữ liệu phù hợp trước khi xây dựng vectorstore

LangChain có cung cấp các phương thức tải dữ liệu để khởi tạo vectorstore rất thuận tiện và đơn giản. Tuy nhiên, cấu trúc metadata của Documents khi được tạo bởi phương thức thuần của LangChain lại không đủ thông tin cho việc chỉnh sửa thông tin sản phẩm sau này:

```
from langchain.document_loaders.csv_loader import CSVLoader
...
print(data[0].metadata)
metadata: {'source': 'your-data-path-here', 'row': 0}
```

Vì thế, các trường trong metadata đã được thay thế để tiện cho việc truy xuất vectorstore sau này.

Đối với các cột không thuộc cột Nội dung và Mô tả chi tiết, khi không có quá nhiều ký tự, hoặc link sản phẩm (không nên cắt thành chunks), thì nội dung của ô thuộc cột đó sẽ được lấy trực tiếp làm nội dung chính của chunks (*page_content*) và các thuộc tính như tên cột (*column*) và thứ tự hàng (*row*) làm metadata:

```
Document(metadata={'column': 'Tên', 'row': 0}, page_content='Tên: Đầu CD Denon QS 10 II')
```

Đối với hai cột Nội dung và Mô tả chi tiết, chứa nội dung lớn và cần được phân mảnh, thì sẽ được tách bằng *text_splitter* đã được khởi tạo ở trên, metadata tương tự và bổ sung thêm *chunk_index* - chỉ số của chunk để biết thứ tự của chunk trong toàn bộ đoạn văn, giúp dễ dàng tổng hợp thành đoạn văn ban đầu cho Admin chỉnh sửa:

```
Document(metadata={'column': 'Chi tiết', 'row': 0, 'chunk_index': 0}, page_content='Chi tiết sản phẩm\nMô tả\nĐầu cd denon qs10ii\ndẹp\nxuất sắc')
```

(ii) Xây dựng Weaviate vectorstore

Trong Python Client v4, mỗi vectorstore được gọi là *collection*. CSDL *collection* định nghĩa dữ liệu được lưu trữ, tổ chức và truy vết trong Weaviate.

Collection của vectorstore dành cho ChatBot được tổ chức như sau:

```
collection = client.collections.create(
    "ChatBot_769Audio",
```

```

properties=[

    Property(
        name="text",
        data_type=DataType.TEXT,
        vectorize_property_name=True, # Use "text" as part of the
value to vectorize
    ),
    Property(
        name="row",
        data_type=DataType.INT,
        skip_vectorization=True, # Don't vectorize this property
    ),
    Property(
        name="column",
        data_type=DataType.TEXT,
        skip_vectorization=True, # Don't vectorize this property
    ),
    Property(
        name="chunk_index",
        data_type=DataType.INT,
        skip_vectorization=True, # Don't vectorize this property
    ),
]
)

```

Trong *collection* trên, có một vài thuộc tính quan trọng trong việc cấu hình vectorstore:

properties: Các thuộc tính của vectorstore, chứa các thuộc tính của dữ liệu trong Documents:

- “text” tương ứng với *page_content*.
- “row” tương ứng với *row* trong metadata.
- “column” tương ứng với *column* trong metadata.
- “chunk_index” tương ứng với *chunk_index* trong metadata.

vectorize_property_name=True: Khi vector hóa / lập chỉ mục bằng OpenAI Embeddings, thì bao gồm cả tên thuộc tính luôn, bên cạnh cả nội dung của chunk, hỗ trợ tìm kiếm tốt hơn.

`skip_vectorization=True`: Các thuộc tính được đặt cờ này sẽ được bỏ qua việc vector hóa, khi chúng chỉ đơn giản là các thuộc tính metedata, không có giá trị tìm kiếm. Điều này giúp giảm chi phí vector hóa những cột không cần thiết.

(c) Các tham số và cải thiện công cụ truy vết (retriever)

Sau khi khởi tạo vectorstore, ta cần phải truy vết dữ liệu cần thiết dựa trên câu truy vấn của người dùng. LangChain cung cấp ba kiểu tìm kiếm (search type) cho việc truy vấn, tuy nhiên, ta sẽ chọn kiểu truy vấn `similarity_score_threshold`. Đây là kiểu truy vấn cho phép trả về các thông tin phù hợp với một ngưỡng nhất định, giúp đảm bảo kết quả được truy vấn đạt độ tin cậy tốt nhất:

```
retriever = docsearch.as_retriever(
    search_type="similarity_score_threshold",
    search_kwargs={
        "score_threshold": 0.3,
        "k": 20
    }
)
```

Tuy nhiên, phản hồi của bot đôi khi sẽ không thể truy vết tất cả các thuộc tính của sản phẩm, do các từ khóa của một sản phẩm đa phần nằm trong cột Nội dung và Mô tả chi tiết, và retriever chỉ thu thập các chunk tương đồng với câu truy vấn, dẫn đến một số tình trạng như sai đường link sản phẩm, báo giá không đúng,... Vì thế, để tài đã bổ sung thêm cho công cụ truy vết, với chức năng tổng hợp các thứ tự dòng của các chunks được thu thập, và thu thập các thuộc tính bổ sung dựa trên các dòng đã được tổng hợp:

```
# Query
query = """
Dạ, Chào anh chị, em đang quan tâm Loa JBL Control 25, Giá bán bên mình
loa này sao anh chị
"""

# Retrieve initial chunks
# Extract row numbers from initial retrieval
# Using a set comprehension to ensure uniqueness
```

```
# Result as a list

print("Row numbers:", row_numbers)
Row numbers: [289, 1161, 1038, 1005, 653, 658]
```

Sản phẩm chính ở trong câu truy vấn này là **Loa JBL Control 25**, ta kiểm tra thử các chunks bổ sung của 1 dòng:

```
Row: 658 - Loa JBL Control 25
Row: 658 - https://769audio.vn/san-pham/312/loa-jbl-control-25-chinh-hang.html
Row: 658 - 2650000.0
Row: 658 - thumb/400_400/2/upload/product/664225935743.jpg
Row: 658 - 1.0
Row: 658 - https://769audio.vn/upload/images/jbl%20control%2025%...
```

Cái phải đánh đổi ở đây là sự phình to chunks, do ta phải thu thập thêm chunks bên ngoài vectorstore. May mắn thay, số lượng chunks và kích thước của chunks là không nhiều khi chúng chỉ bao gồm các thông tin đơn giản như tên, giá, tình trạng,...

Tổng số ký tự trong kết quả truy vết ban đầu: **10661**

Tổng số ký tự trong kết quả truy vết được bổ sung thêm chunks: **12250**

(d) Mô phỏng học tăng cường cho truy vấn và phản hồi bằng Business Logic và User Feedback

Khóa luận tích hợp khả năng tiếp nhận business logic để cá nhân hóa phản hồi cho doanh nghiệp, giúp chatbot trả lời theo đúng nghiệp vụ, quy trình của riêng từng doanh nghiệp, và tiếp nhận user feedback để chatbot học hỏi từ phản hồi của khách hàng.

Đây là một cách mô phỏng học tăng cường (Reinforcement Learning) thông qua prompt mà không phải tinh chỉnh (fine-tune) cho mô hình GPT-4o-mini.

Cách làm này giúp bot thích nghi với trường hợp trong quá khứ và đưa ra phản hồi phù hợp với nhu cầu thực tế hơn trong tương lai. Đồng thời, việc tinh chỉnh này sẽ linh hoạt, nhanh chóng hơn, vì bản chất, hoạt động này chỉ được các thông tin ngữ cảnh bổ sung vào prompt thay vì cập nhật thăng vào model.

Việc học tăng cường này được áp dụng cho trước khi truy vấn và trước khi phản hồi, giúp tối ưu hóa kết quả truy vấn và phản hồi của ChatBot:

- Học tăng cường cho truy vấn:

```
contextualize_q_system_prompt = """
```

```
...
```

Nhiệm vụ:

Viết lại câu hỏi của khách thành một truy vấn tìm kiếm sản phẩm ngắn gọn nhưng giàu ngữ cảnh, tận dụng ngữ cảnh từ lịch sử chat và hồ sơ người dùng, theo quy trình sau:

```
...
```

- Rút gọn thành câu truy vấn ngắn gọn, ưu tiên từ khóa quan trọng: sản phẩm, thương hiệu, tính năng, nhu cầu, giá,...

```
"""
```

Ví dụ 1:

raw_input: dàn karaoke bass mạnh, tầm 10 triệu

business_logic: Khi giới thiệu sản phẩm, ưu tiên hàng khuyến mãi/giảm giá, sản phẩm không bỏ mẫu

➔ Contextualized query: dàn karaoke bass mạnh trong tầm 10 triệu, **ưu tiên khuyến mãi, không bỏ mẫu**

Ví dụ 2:

raw_input: Cho hỏi mua loa jbl **state** coi

user_feedback: state ý là stage, nhưng do khách viết sai

➔ Contextualized query: mua loa jbl **stage**, ưu tiên ưu đãi

- Học tăng cường cho phản hồi:

Ví dụ:

raw_input: Tôi muốn mua loa bose 301 seri 5 hàng bãi

contextualized_query: mua loa bose 301 seri 5 bãi, **ưu tiên khuyến mãi, không bỏ mẫu**

user_feedback: Nếu khách hỏi loa bose 301 seri 5 bãi, **dùng giới thiệu mẫu 5tr3, cái đó bỏ mẫu hết hàng rồi, giới thiệu qua con chữ lớn 6tr2** ấy

➔ AI Answer: Vâng, bên tôi có bán loa bose 301 series 5 hàng bãi phù hợp với nhu cầu của Anh/Chị ạ:

Sản phẩm:

Tên sản phẩm: Loa Bose 301 Seri 5 | Hàng Xịn Mexico | Mỹ Chính Hãng

- Giá: 6,200,000 VND

- Bảo hành: 2 tháng

- Tình trạng: Còn hàng

...

Ghi chú: Loa bose 301 seri 5 bãi có nhiều loại. Nên ở ví dụ này, user_feedback gợi ý cụ thể sản phẩm nào đã bỏ mẫu và nên sử dụng sản phẩm nào thay thế, giúp chatbot phản hồi chính xác hơn với ngữ cảnh thực tế mà không phải phụ thuộc hoàn toàn vào nội dung truy vấn từ vectorstore.

3.4.2.3. Agent 3 - Dẫn Dắt Hội Thoại

Agent 3 – Followup Generator là một thành phần quan trọng trong kiến trúc ChatBot của đề tài, chịu trách nhiệm duy trì mạch hội thoại và đảm bảo khách hàng tiếp tục được dẫn dắt theo đúng định hướng quy trình bán hàng. Agent này không chỉ thực hiện chức năng tạo ra câu hỏi tiếp theo, mà còn tích hợp khả năng đánh giá ngữ cảnh và dự đoán hành vi để quyết định chiến lược tương tác phù hợp nhất.

Cụ thể, trước khi tạo câu hỏi mới, Agent 3 sẽ **đánh giá xem có cần thiết phải đặt ra câu hỏi hay không**. Điều này xuất phát từ thực tế rằng trong nhiều tình huống, phản hồi của Agent 2 (Product Responser) đã bao gồm sẵn một hoặc nhiều câu hỏi định hướng. Do đó, nếu tiếp tục đặt câu hỏi mới không đúng lúc, hệ thống có thể gây cảm giác dư thừa hoặc làm loãng trải nghiệm người dùng. Đây là điểm thể hiện rõ tư duy ngữ cảnh hóa trong quá trình tương tác.

Tiếp đến, nếu nhận định rằng việc đặt câu hỏi là cần thiết, Agent 3 sẽ **tạo ra câu hỏi follow-up phù hợp với ngữ cảnh đoạn hội thoại** sau khi Agent 2 đã trả lời truy vấn ban đầu. Các câu hỏi này được thiết kế để không chỉ duy trì sự tương tác mà còn góp phần khai thác thêm thông tin từ người dùng, phục vụ cho các giai đoạn kế tiếp như xử lý phản đối, định tính khách hàng, hoặc đề xuất mua hàng.

Cuối cùng, Agent 3 còn thực hiện chức năng **dự đoán bước tiếp theo trong quy trình bán hàng**, dựa trên các yếu tố như trạng thái hội thoại hiện tại, thông tin khách hàng đã chia sẻ, và kết quả phân tích từ các Agent trước đó. Việc dự đoán này giúp hệ thống không chỉ phản ứng một cách thụ động mà còn chủ động định hướng chiến lược dẫn dắt người dùng – yếu tố then chốt trong việc tăng tỷ lệ chuyển đổi của ChatBot.

```
followup_prompt = """
...
- Các giai đoạn bán hàng: {workflow_stages}
- Luồng hội thoại: Q_user => A_AI => follow-up (nếu cần) => Q_user => ...
..."""

Logic quyết định:
```

1. Xem lịch sử + hồ sơ người dùng.
2. Nếu A_AI chứa "?" hoặc stage hiện tại là Lời chào => không tạo follow-up, giữ stage Lời chào.
3. Ngược lại:
 - a. Dự đoán stage tiếp theo theo intent + internal_instruction.
 - b. Tạo comment ngắn phù hợp hồ sơ.
 - c. Sinh câu hỏi follow-up:
 - Nếu <2 vòng thu thập thông tin => hỏi làm rõ.
 - Nếu ≥2 vòng => chuyển sang trình bày sản phẩm, cá nhân hóa theo hồ sơ.

Trong phần thiết kế prompt này, một quy tắc quan trọng được hệ thống áp dụng, rằng không cần đưa ra câu followup khi Agent 2 đã đặt một câu hỏi trước đó. Điều này xuất phát từ thực tế rằng, Agent đôi khi có thể tự đặt câu hỏi mà không cần phải bám sát quy trình:

User Message: Tôi muốn mua loa

AI Message: Vâng, anh muốn mua loa nhu thế nào? ...

AI Followup: None

Quy tắc này giúp hệ thống tránh việc chồng chéo câu hỏi, khiến người dùng không bị hoang mang, khó hiểu. Bên cạnh đó, nếu Agent 2 đã tự đặt câu hỏi, thì hệ thống khỏi cần phải followup, giúp tối ưu chi phí từ Agent 3.

Ngoài ra, để tăng tính cá nhân và sự tương tác, câu follow sẽ thỉnh thoảng bắt đầu với một câu nhận xét ngắn gọn, liên hệ với thông tin đã biết trong hồ sơ khách hàng hoặc nhu cầu về sản phẩm mà ChatBot thu thập được, trước khi đặt câu hỏi follow. Bằng cách tiếp cận này, cuộc hội thoại sẽ mang hơi hướng Hỏi - Kết Nối - Trả Lời, người dùng sẽ không có cảm giác bị đặt câu hỏi liên tục:

User Message: Tôi là giáo viên, muốn tìm micro để giảng dạy.

AI Message: Dạ vâng, bên em có các loại micro chuyên dùng cho giảng dạy
...

AI Followup: **Với tính chất công việc giảng dạy, chắc hẳn anh/chị cần một chiếc micro có khả năng thu âm rõ ràng và hạn chế tiếng ồn xung quanh.** Không biết anh/chị thích micro cài áo, micro cầm tay, hay micro để bàn ạ?

3.4.2.4. Agent 4 - Cập Nhật Hồ Sơ

Agent này có nhiệm vụ cập nhật thông tin khách hàng sau ở cuối chu trình chat, dựa trên các thông tin được tạo ra bởi các Agents trước. Đây là Agent thể hiện nổi bật nhất tính học hỏi kinh nghiệm từ những lần tương tác trước, khi cập nhật liên tục thông tin người dùng, để các Agents khác học hỏi chân dung khách hàng để đưa ra lựa chọn, kết quả phù hợp.

```
update_prompt = f"""
    ...
    
```

Hành động: Phân tích tin nhắn người dùng mới nhất và phản hồi AI, theo dõi để trích xuất bất kỳ thông tin mới nào về khách hàng, chẳng hạn như nghề nghiệp, sở thích, ngân sách, mối quan tâm hoặc bất kỳ thông tin chi tiết quan trọng nào khác. Cập nhật hồ sơ người dùng hiện có cho phù hợp.

Kết quả: Trả về hồ sơ người dùng đã cập nhật dưới dạng đoạn văn ngôn ngữ tự nhiên có cấu trúc tốt, kết hợp mọi chi tiết mới trong khi vẫn đảm bảo tính rõ ràng và dễ đọc.

Ví dụ:

```

    ...
    Chỉ trả về hồ sơ người dùng đã cập nhật dưới dạng JSON:
    {
        "updated_profile": "<Đoạn văn hồ sơ người dùng dưới dạng ngôn
        ngữ tự nhiên>"
    }
    """

```

Kết quả người dùng được lưu ở dạng đoạn văn, đơn giản là giúp tư vấn viên khi tiếp nhận phiên chat của khách hàng, sẽ dễ hiểu hơn khi nó được trình bày ở dạng ngôn ngữ tự nhiên.

3.4.3. Xây dựng CRUD vectorstore

Khác với các thư viện ORM phổ biến trong Python như Django hay Flask-SQLAlchemy, cung cấp các lớp và phương thức đơn giản để thao tác với cơ sở dữ liệu, Weaviate vectorstore yêu cầu người dùng thực hiện các thao tác chỉnh sửa CSDL một cách thủ công thông qua RESTful API, khi người dùng phải trực tiếp thêm, xóa và sửa các trường dữ liệu trên vectorstore. Bên cạnh đó, Weaviate còn sử dụng GraphQL API để truy xuất dữ liệu, nhằm hỗ trợ cho các thao tác CRUD thông qua RESTful API. Sự phức tạp này không chỉ đòi hỏi người dùng phải nắm vững cả hai APIs mà còn cần hiểu rõ cách thức hoạt động của vectorstore, từ đó tối ưu hóa việc quản lý và truy xuất dữ liệu trong các ứng dụng sử dụng Weaviate.

3.4.3.1. Sửa sản phẩm

Sau khi GraphQL API trả về kết quả của 1 sản phẩm, Admin có thể chỉnh sửa các thuộc tính của sản phẩm ấy. Việc chỉnh sửa các trường thông tin sản phẩm như Tên hoặc Giá khá đơn giản, nhưng các trường Nội dung và Mô tả chi tiết lại yêu cầu phức tạp hơn nhiều. Nội dung của các trường này đã được phân mảnh thành các phân mảnh (chunks) có sự chồng lặp (chunk overlap), nghĩa là các phần sau có thể lặp lại một số ký tự từ các phần trước. Do đó, khi hiển thị nội dung của 2 trường này để Admin chỉnh sửa, cần phải loại bỏ các ký tự bị lặp để đoạn văn trở nên rõ ràng và hợp lý:

Đoạn văn ban đầu (có từ lặp):

...Cd denon gcd s10ii xuất sắc, điện 100v, có remote - **hình thức, hình thức đẹp**, hàng tuyển chọn kỹ lưỡng. Cd denon gcd s10ii là dòng sản phẩm cao cấp của hãng denon, bộ...

Đoạn văn được xử lý (không có từ lắp):

...Cd denon gcd s10ii xuất sắc, điện 100v, có remote - **hình thức đẹp**, hàng tuyển chọn kỹ lưỡng. Cd denon gcd s10ii là dòng sản phẩm cao cấp của hãng denon, bộ...

Khi chỉnh sửa nội dung, ta cần xác định rõ phần nội dung nào đã được chỉnh sửa, vì các phần tiếp theo có thể chứa nội dung lắp lại từ phần trước. Điều này giúp chúng ta nắm chính xác những phần nào đã được chỉnh sửa:

```
updated_paragraph = """
```

Chi tiết sản phẩm

Mô tả

Đầu cd denon gcd qs10ii đẹp xuất sắc Xuất xứ: japan, bảo hành **2** tháng
Cd denon gcd s10ii xuất sắc, điện **200v**, có remote - **hình thức đẹp, hàng tuyển chọn kỹ lưỡng.**

```
"""
```

Trong đoạn văn trên, người dùng chỉnh sửa thời gian bảo hành từ 1 ⇒ 2 tháng, và điện 100v ⇒ 200v. Xác định các chunks được chỉnh sửa:

Token 'Chi tiết sản phẩm Mô tả Đầu cd denon gcd qs10ii đẹp xuất sắc' is unchanged

Token 'Xuất xứ: japan, bảo hành **1** tháng' has changed to 'Xuất xứ: japan, bảo hành **2** tháng'

Token 'Cd denon gcd s10ii xuất sắc, điện 100v, có remote - **hình thức**' has changed to 'Cd denon gcd s10ii xuất sắc, điện **200v**, có remote - **hình thức**'

Token 'đẹp, hàng tuyển chọn kỹ lưỡng.' is unchanged

Updated tokens:

UUID: c8f074a7-cafe-478e-9dcb-407a63730781, Updated Text: Xuất xứ: japan, bảo hành **2** tháng

UUID: 8af25cf7-0a25-4073-a672-253e1583e106, Updated Text: Cd denon gcd s10ii xuất sắc, điện **200v**, có remote - **hình thức**

Từ đó, dựa vào thuộc tính UUID của từng đối tượng, ta dùng RESTful API để cập nhật lại những chunks đã được chỉnh sửa. Kiểm tra lại thông tin sản phẩm sau khi chỉnh sửa:

Text: Tên: Đầu CD Denon QS 10 II

...

Text: Chi tiết sản phẩm

Mô tả

Đầu cd denon gcd qs10ii đẹp xuất sắc

Text: - hình thức đẹp, hàng tuyển chọn kỹ lưỡng.

Text: Cd denon gcd s10ii là dòng sản phẩm cao cấp của hãng denon, bộ

Text: Xuất xứ: japan, bảo hành **2** tháng

Text: Cd denon gcd s10ii xuất sắc, điện **200v**, có remote - hình thức

...

3.5. Kết quả đề tài

3.5.1. Dữ liệu

3.5.1.1. Dữ liệu huấn luyện

Bảng 3.7: Một vài dòng đầu trong CSDL

Tên	Link sản phẩm	Giá	Link ảnh	Tình trạng	Mô tả	Nội dung
Loa Bose 301 Seri 5 Hàng Xịn Mexico Mỹ Chính Hãng	https://769audio.vn/san-pham/132/loa-bose-301-seri-v.html	6200000	https://769audio.vn/upload/images/Bo se%2030 1%20seri %205_%205(2).jpg,...	1	Loa Bose 301 Seri 5 là gì : dòng loa nghe nhạc nổi bật của thương hiệu Bose...	Loa Bose 301 series 5 Huyền Thoại Là Sản Phẩm Chính Hãng Loa Bose 301 seri 5 nguyên zin còn đẹp, bảo hành 2 tháng...

Tên	Link sản phẩm	Giá	Link ảnh	Tình trạng	Mô tả	Nội dung
Loa Bose Acoustimass M10 Seri IV	https://9audio.vn/san-pham/147/loa-bose-acoustimass-m10-seri-iv.html	14500000	https://9audio.vn/upload/images/C%E1%BBA5c%20%,...	1	Loa Bose Acoustimass 10 Series IV là một trong những dòng hệ thống xem phim...	Loa Bose M10 Acoustimass Series IV Hàng Mỹ Xịn Bose nổi tiếng với công nghệ âm thanh tiên tiến,...
Amply Accuphas e E405 Đẹp Hoàn Hảo	https://9audio.vn/san-pham/2447/amply-accuphas-e-e405-hang-bai.html	52500000	https://9audio.vn/upload/images/405_8.jpg,...	1	Amply Accuphase E-405 là sự kết hợp hoàn hảo giữa thiết kế thanh lịch,...	AMPLY ACCUPHASE E405 hàng bãi đẹp Long Lanh ...

3.5.1.2. Dữ liệu kiểm tra

Bảng 3.8: Một vài dòng đầu trong nội dung kiểm tra hiệu quả ChatBot

ID	Questions	Real responses
2	Bên em amply jarguar ko	dạ còn amply Jarguar 203 gold limited
	tôi muốn mua amply Jarguar 203 gold limited	Mẫu này hiện tại còn hàng nha anh, giá 7.2 triệu, mới bao zin

ID	Questions	Real responses
4	Cho mình hỏi con jamo d590 còn hàng không Bạn	không còn anh
	Thấy mình để hàng trung bày còn hàng	Ngưng sản xuất lâu rồi anh
5	Cho m hỏi bose s1 pro có hàng lại rồi hả	Dạ có, loa Bose S1 Pro hiện đang còn hàng a. Sản phẩm này có giá là 14,500,000 VNĐ và được bảo hành 12 tháng. Nếu bạn cần thêm thông tin chi tiết hoặc muốn mua sản phẩm, hãy cho tôi biết nhé!
6	Micro D2 Alpha	alpha hết rồi anh, bên em về hàng micro rồi nhé anh, Giá 1 triệu 950k
8	Loa Bose 101 cũ xịn Mỹ là một sản phẩm âm thanh nổi tiếng, sản phẩm loa, Chú muốn mua cặp loa lời này, Hàng xịn	dạ có

3.5.2. Sơ lược về các độ đo kết quả của mô hình

Để đánh giá toàn diện hiệu quả vận hành của mô hình chatbot đa tác vụ, khóa luận xây dựng một hệ thống các thước đo kết quả được chia thành hai nhóm chính:

Một là, **Đánh giá tầng mô hình (Task-Level Evaluation)**: tập trung vào hiệu suất kỹ thuật của các agent trong việc xử lý phản hồi.

Hai là, **Đánh giá tầng phiên hội thoại (Session-Level Evaluation)**: tập trung vào trải nghiệm người dùng và sự hài lòng của họ trong suốt phiên tương tác.

Việc phân tầng này nhằm đảm bảo quá trình đánh giá vừa tuân thủ các chuẩn kỹ thuật, vừa phản ánh đúng giá trị thực tế khi triển khai chatbot vào quy trình bán hàng.

3.5.3. Đánh giá tầng mô hình (Task-Level Evaluation)

3.5.3.1. Giới thiệu Precision Soft

Đánh giá tầng mô hình tập trung vào việc đo lường độ chính xác và khả năng phản hồi đúng ngữ cảnh của chatbot ở từng tác vụ cụ thể. Các kết quả đánh giá được thực hiện bởi các người dùng thử và các nhân viên bán hàng có kinh nghiệm, giúp đảm bảo tính khách quan trong kiểm tra chất lượng kết quả trả lời.

Trong khuôn khổ đề tài này, **Precision Soft** được dùng để thay thế chỉ số Recall truyền thống. Lý do là trong các kết quả đánh giá, chúng không chỉ bao gồm kết quả nhị phân là đúng hay sai (1 hoặc 0), mà được chấm theo tính phù hợp so với câu hỏi và ngữ cảnh. Vì thế Precision Soft sẽ cho phép hệ thống được đánh giá một cách tương đối hơn theo các mức được quy định.

Mỗi phản hồi của chatbot được chấm điểm theo thang:

- 1: Hoàn toàn phù hợp và chính xác.
- 0.5: Tương đối phù hợp, cần cải thiện nhẹ.
- 0: Không phù hợp, sai ngữ cảnh.

Độ chính xác của mô hình được tính bằng công thức:

$$\text{Precision}_{\text{soft}} = \frac{\sum \text{Score}}{\text{Responses}} \times 100\%$$

Trong đó:

- Precision Soft: phản ánh tỷ lệ phản hồi chất lượng tốt và chấp nhận được trên tổng số phản hồi.
- Score: Tổng số điểm của các phản hồi
- Responses: Số lượng các phản hồi

Dưới đây là chính xác của Agents theo tỷ lệ Precision:

Bảng 3.9: Độ chính xác Precision Soft theo Agent

Agent	Số lượng phản hồi	Tổng điểm đánh giá	Precision Soft (%)
Agent 2 (Trả lời sản phẩm)	472	371.0	78.6

Agent 3 (Follow-up)	229	193.5	84.5
Trung bình (Average)	701	564.5	80.58

Trung bình Precision Soft toàn hệ thống đạt **80.58%**, cho thấy rằng dù với hai tác vụ khác nhau (trả lời sản phẩm và follow-up), chatbot vẫn duy trì được tỷ lệ phản hồi đạt chất lượng từ mức chấp nhận được trở lên ở mức khá cao.

3.5.3.2. Đánh giá

Nhìn chung, Agent 2 cho thấy khả năng trả lời câu hỏi ở mức khá tốt với **78.6%**, giúp giải đáp được các câu truy vấn đơn giản và trung bình. Tuy nhiên, trong quá trình phân tích, Agent 2 vẫn còn tồn đọng một vài hạn chế:

Một là, **xử lý các câu truy vấn phức tạp**: ChatBot gặp vấn đề với những câu truy vấn phức tạp yêu cầu xử lý nhiều thực thể (thường là sản phẩm), dẫn đến việc trả về thông tin không chính xác.

Hai là, **phân tích nhu cầu kỹ thuật của khách hàng**: Khả năng của bot trong việc hiểu và phân tích nhu cầu của khách hàng là chưa tốt. Điều này đa phần do training data của model chỉ bao gồm thông tin sản phẩm mà ít chứa các thông tin về kỹ thuật lắp đặt, cấu hình,...

Ba là, **hạn chế của truy vết dữ liệu**: LangChain Retriever chủ yếu thu thập các chunks dựa trên từ khóa của câu truy vấn, đa phần các chunks này được lấy từ cột “Thể” và các cột về Nội dung chi tiết, và các mẫu chunks này cũng khá “rải rác”, khó có thể tập trung vào những sản phẩm cần được truy vết. Vì thế, đôi khi các yêu cầu quá sâu vào trong sản phẩm, thì Chatbot vẫn sẽ bỏ sót, dẫn đến trả về kết quả sai.

Bốn là, **nhận biết từ khóa**: Bot thường gặp khó khăn với những từ “lóng” mà người dùng hay sử dụng hoặc từ viết sai chính tả, mà không được đề cập trong bài viết sản phẩm, dẫn đến khó khăn trong việc truy vết chính xác hay nhận biết sản phẩm cần thiết.

Về phần Agent 3, độ chính xác đạt **84.5%** cho thất chất lượng follow-up hợp lý của chatbot là ở mức tốt, dù đây là tác vụ khó hơn do phải dựa vào ngữ cảnh hội thoại mở. Do tính phức tạp của nó, cũng như việc phải đọc qua rất nhiều thông tin từ bộ nhớ

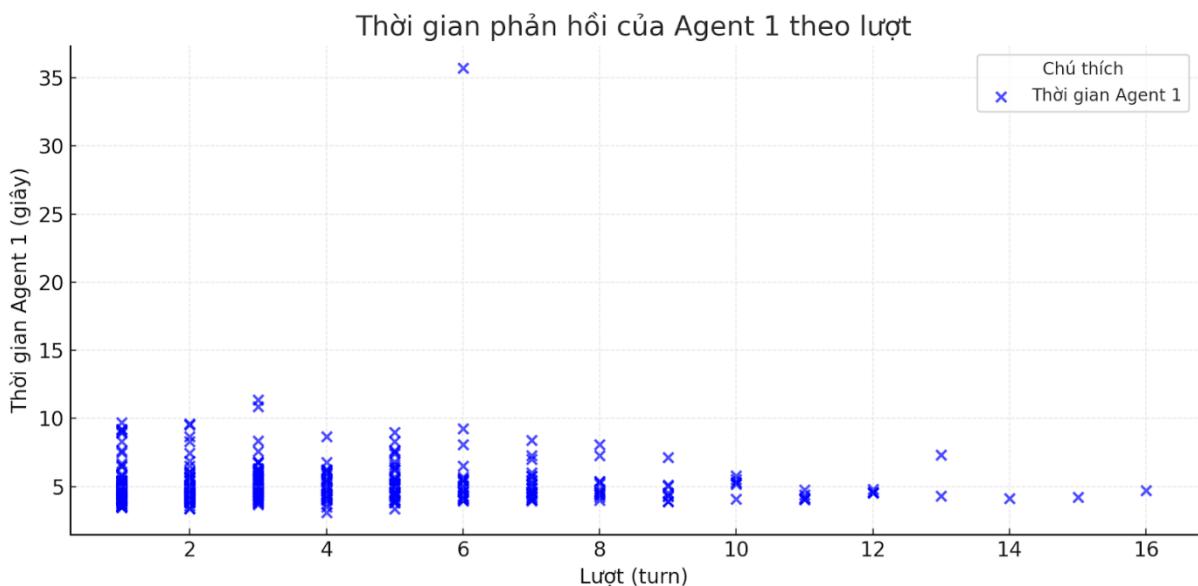
ngoài, lịch sử chat, nên kết quả phản hồi đôi khi cũng gặp phải sự sai lệch thông tin, lặp lại câu followup, mà chỉ riêng việc Prompt Engineering cũng khó giải quyết.

3.5.4. Đánh giá tầng phiên hội thoại (Session-Level Evaluation)

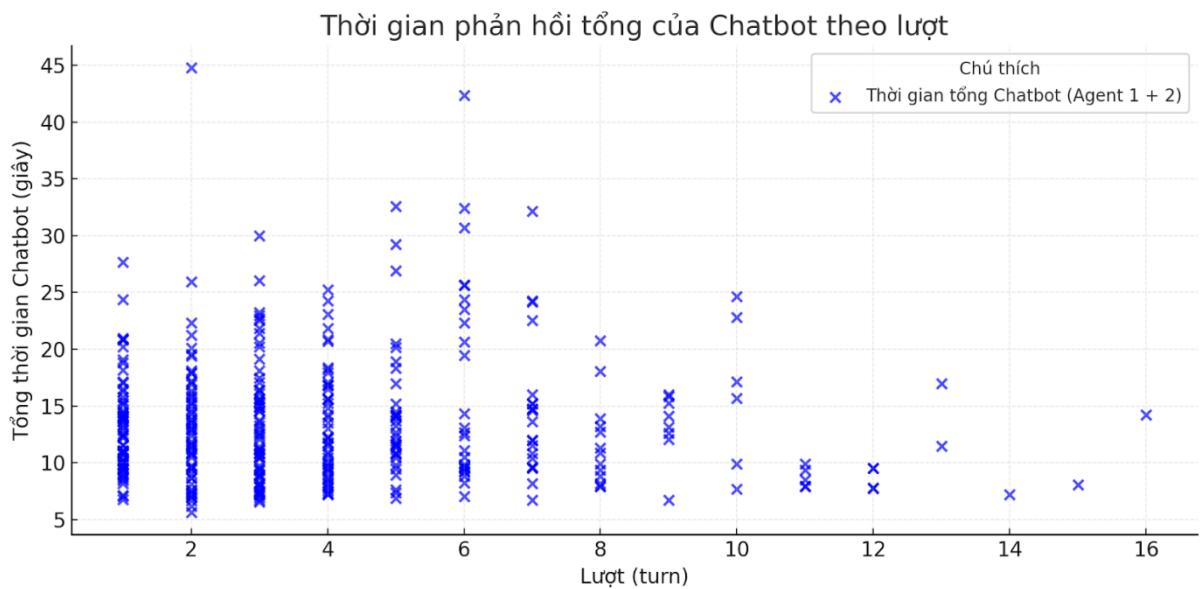
Đánh giá tầng phiên hội thoại tập trung vào cảm nhận tổng thể và sự hứng thú muôn tiếp tục của người dùng sau một phiên tương tác với chatbot.

(a) Thời gian phản hồi (Response Time)

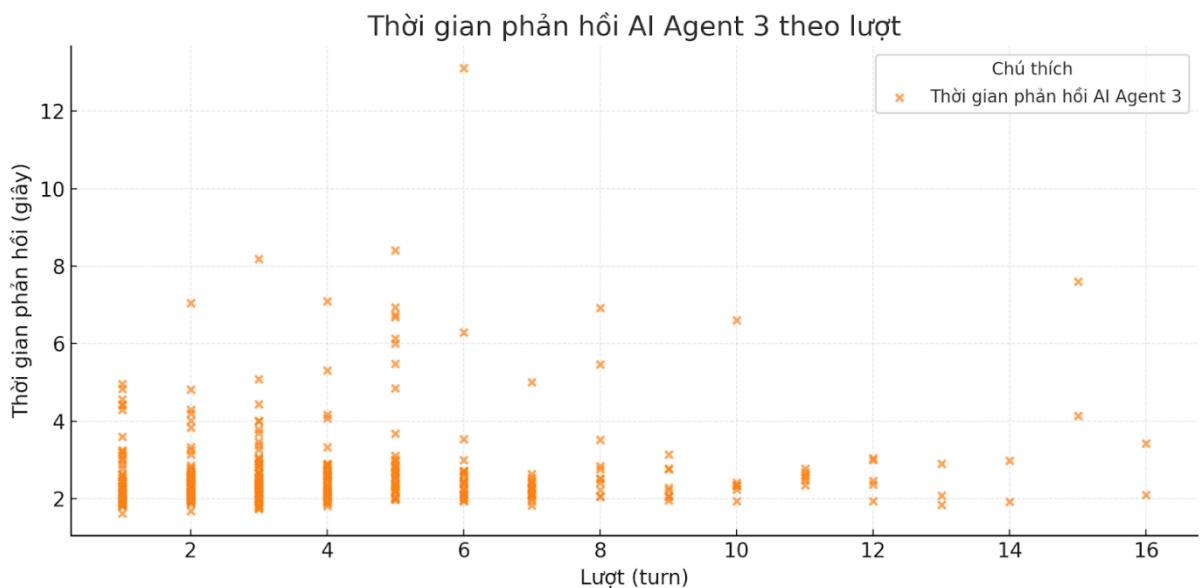
Thời gian phản hồi của chatbot được đo bằng đơn vị giây, tính từ lúc người dùng gửi câu hỏi đến khi chatbot trả lời hoàn chỉnh.



Hình 3.3: Biểu đồ phân tán hiển thị thời gian phản hồi của Agent 1 theo lượt



Hình 3.4: Biểu đồ phân tán biểu diễn thời gian phản hồi của Agent 1 và Agent 2 theo lượt



Hình 3.5: Biểu đồ phân tán biểu diễn thời gian phản hồi của Agent 3 theo lượt

Việc thể hiện biểu đồ thời gian của Agent 1 và tổng thời gian của Agent 1 và 2 là do kiến trúc của hệ thống Chatbot, khi thời gian nên được tính kể từ khi người dùng nhập câu truy vấn cho bot, Agent 1 nhận câu truy vấn, phân phối Agents rồi mới tới Agent 2 phản hồi lại cho người dùng.

Đối với Agent 1, hầu hết thời gian phân phối Agents duy trì khoảng từ 3-10 giây. Đây là một kết quả tương đối khá, khi việc phân phối yêu cầu agent suy luận nhiều hơn để

ra quyết định phân phối phù hợp. Tuy nhiên, các trường hợp ở gần mốc 10 giây vẫn là quá lâu, khi phải tính cả thời gian phản hồi thật sự cho người dùng ở các Agents sau.

Khi kết hợp với Agent 2, đa phần thời gian phản hồi của bot tập trung ở 5-20 giây trong vòng 8 lượt chat đầu, cho thấy yêu cầu phải xử lý lượng thông tin ngữ cảnh phức tạp, bao gồm các phần lớn các thông tin về lịch sử trò chuyện, cũng như thông tin sản phẩm. Việc truy xuất thông tin sản phẩm thực chất không mất nhiều thời gian xử lý, vì theo một vài quan sát sơ bộ trong quá trình debug, tổng thời gian truy xuất thông tin sản phẩm chỉ mất dưới 2 giây với $k = 20$. Điều này minh chứng cho thời gian phản hồi đa phần đến từ việc xử lý thông tin ngữ cảnh trong prompt, khi LLM phải phân tích kỹ các thông tin sản phẩm được cung cấp trước khi phản hồi về.

Ngược lại, Agent 3 phản hồi nhanh hơn rất nhiều, dưới 5 giây ở đa số các lượt. Điều này phản ánh đúng nghiệp vụ của agent, khi không cần phải xử lý quá nhiều thông tin sản phẩm, mà chỉ cần hiểu đúng ngữ cảnh hiện tại và bước tiếp theo của quy trình bán hàng là đã có thể phản hồi.

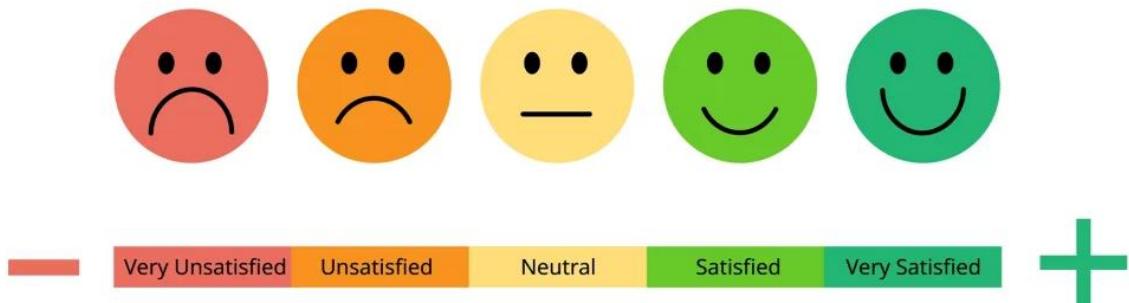
Một vài điểm dữ liệu ngoại lai (outliers) bất thường của các Agents có thể được giải thích do model phải kiểm tra logic nhiều hơn bình thường, cũng có thể do đường truyền mạng làm kéo dài thời gian phản hồi.

(b) Sự hài lòng của khách hàng (CSAT - Customer Satisfaction Score)

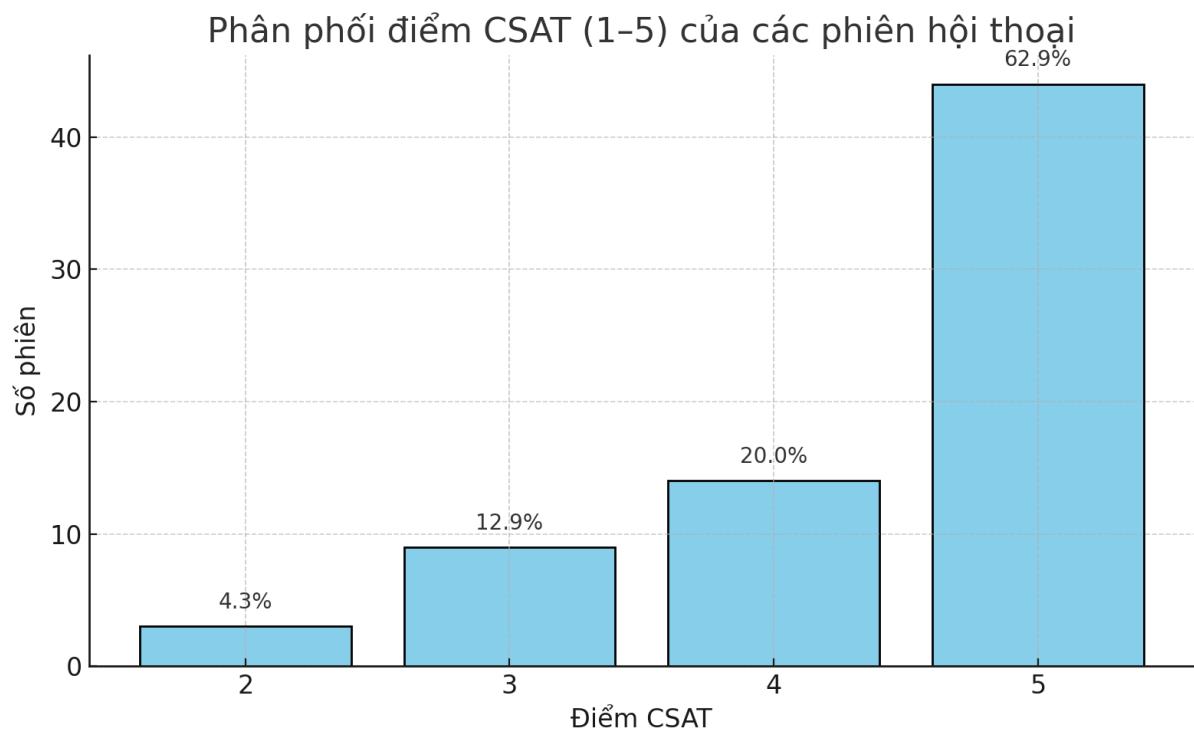
Độ hài lòng người dùng sau phiên chat được đo bằng thang điểm 5, cao nhất là 5 điểm và giảm dần về thấp nhất là 1:

- 5: Rất hài lòng, muốn tiếp tục mua hàng hoặc được tư vấn thêm.
- 1: Rất không hài lòng, muốn kết thúc phiên chat.

Customer Satisfaction Score (CSAT)



Hình 3.6: Thang điểm cho CSAT (Nguồn: <https://base.vn/blog/csat-la-gi/>)



Hình 3.7: Biểu đồ cột thể hiện sự hài lòng của khách hàng qua cuộc hội thoại

Hơn 60% phiên đạt điểm 5, chứng tỏ sự hài lòng và muốn tiếp tục của người dùng khi trao đổi với Chatbot. 82.9% tổng phiên đạt điểm mức 4 và 5, càng cho thấy khả năng dẫn dắt tốt của chatbot và trải nghiệm tốt của khách hàng.

Trong khi đó, lần lượt 12.9% và 4.3% phiên đạt điểm 3 và 2, do chatbot cung cấp sai thông tin yêu cầu, chưa nắm bắt rõ thông tin người dùng cung cấp, dẫn đến gợi ý sản phẩm, đặt câu hỏi kém chính xác, không đúng trọng tâm.

3.5.5. Giải thích về mất cân bằng lớp (Class Imbalance) giữa các loại câu hỏi khi kiểm tra hệ thống Chatbot

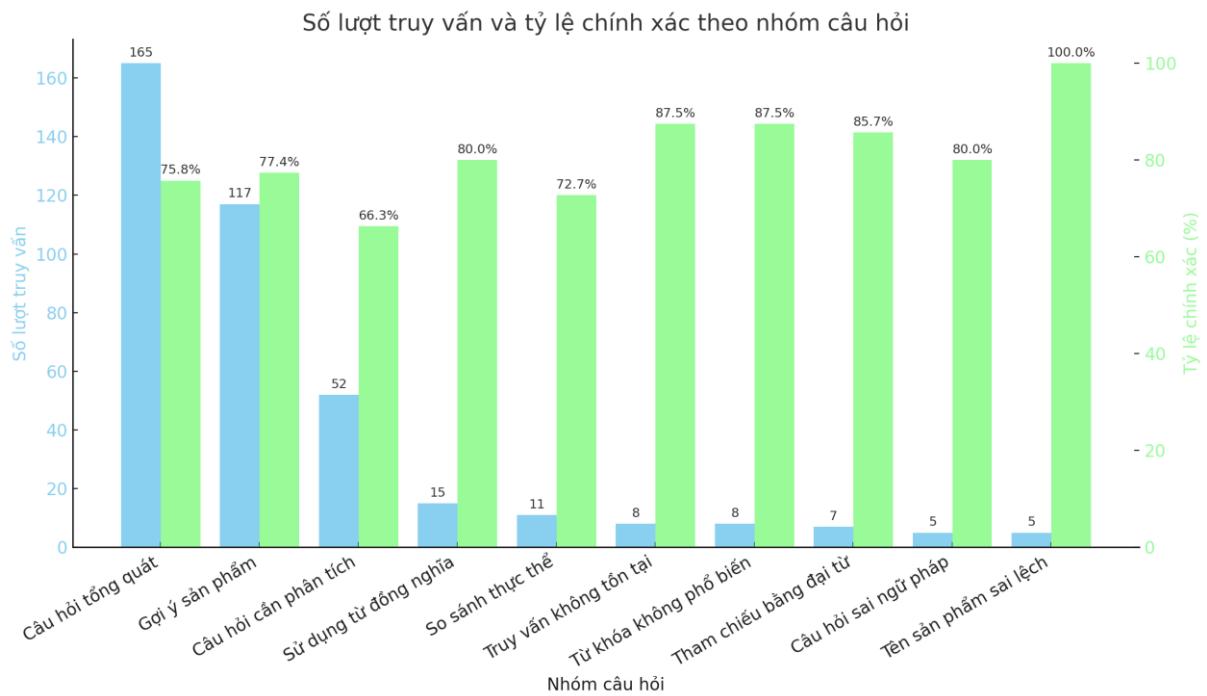
Đây là tình hình thực tế trong nghiệp vụ bán hàng, khi đa phần khách hàng chỉ muốn đặt các câu hỏi tổng quát như sản phẩm, giá thành,... và số ít sẽ hỏi về các trường hợp sâu hơn, hiếm gặp hơn như cách phối ghép sản phẩm,...

Bảng 3.10: Ngẫu nhiên 8 câu hỏi thuộc nhóm Câu hỏi tổng quát

Câu hỏi
Kết nối dễ ko anh?
em để trong phòng khác hát karaoke
Ko co vo đen hả e
Màu đen giá thế nào a, Có hàng xách tay khg?
Với amply này mình bảo hành bao lâu anh
chính hãng nhập khẩu ở đâu
Vang số Karaoke JBL KX180A Black, nhiêu tiền shop
Micro không dây JBL VM300 (BS) Black
Vậy mua sub khả thi hơn nhỉ

Việc ChatBot tập trung vào giải quyết đa số các nhu cầu của khách hàng mà vẫn đảm bảo độ chính xác tốt, và những câu hỏi khó hơn, ít người yêu cầu hơn sẽ được nhân sự trực tiếp trao đổi, sẽ giúp giảm gánh nặng chi phí tư vấn khách hàng của nhân viên bán hàng.

Do đó, dữ liệu kiểm tra phản ánh đúng hành vi của khách hàng, khi gần 42% trên tổng 392 câu truy vấn thuộc vào các câu hỏi tổng quát mà khách hàng thường đặt cho người bán hàng:



Hình 3.8: Biểu đồ cột đôi thể hiện Số lượt truy vấn và tỷ lệ chính xác theo nhóm câu hỏi

Các câu hỏi tổng quát chiếm số lượng áp đảo với 165 truy vấn, thể hiện qua việc khách hàng thường đặt những câu hỏi chung chung như tên sản phẩm, giá, màu sắc, bảo hành,... với độ chính xác là 75.8% - mức chấp nhận được, nhưng cũng cần được cải thiện thêm thông qua tối ưu Prompt Engineering, cung cấp thêm feedback và thông tin học thêm về sản phẩm cụ thể hơn,...

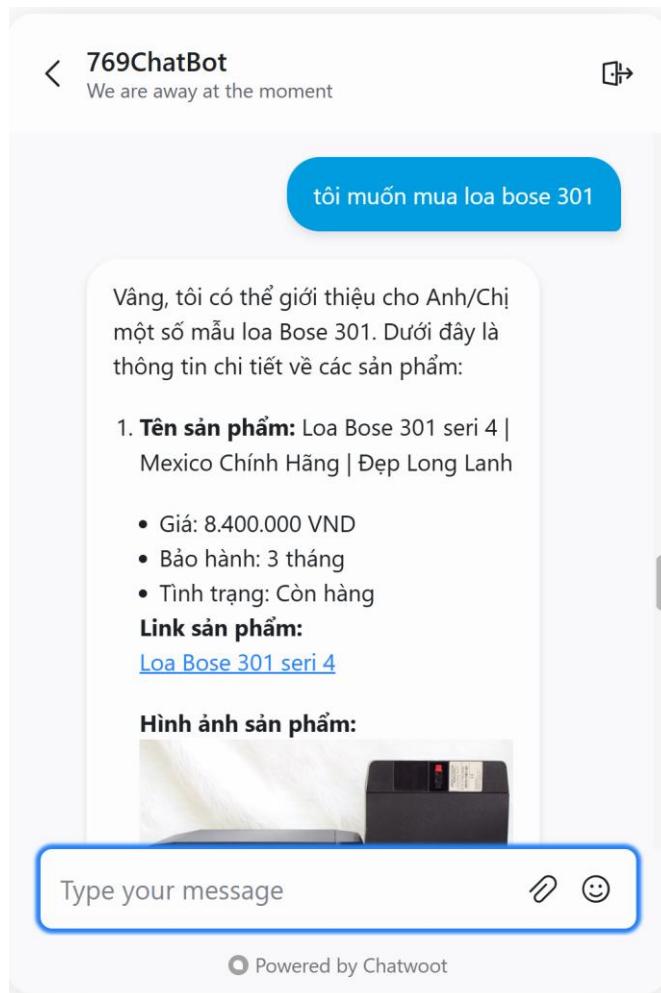
Hai nhóm chiếm đa số tiếp theo là nhóm câu hỏi Gợi ý sản phẩm và các Câu hỏi cần phân tích, cũng có độ chính xác tạm ổn, do các câu hỏi này yêu cầu độ hiểu biết rõ về sản phẩm để tư vấn, so sánh mà việc truy vấn thông thường khó có thể trả về các thông tin cần thiết để phản hồi.

Các nhóm câu hỏi thiểu số còn lại, tuy chiếm số lượng ít, khá khó để đánh giá một cách khách quan, nhưng chúng thể hiện rằng đây là các trường hợp người dùng ít đặt ra, nên được ưu tiên tối ưu cuối cùng.

3.5.6. Các chức năng hoàn chỉnh của hệ thống

3.5.6.1. Chatbot tư vấn bán hàng

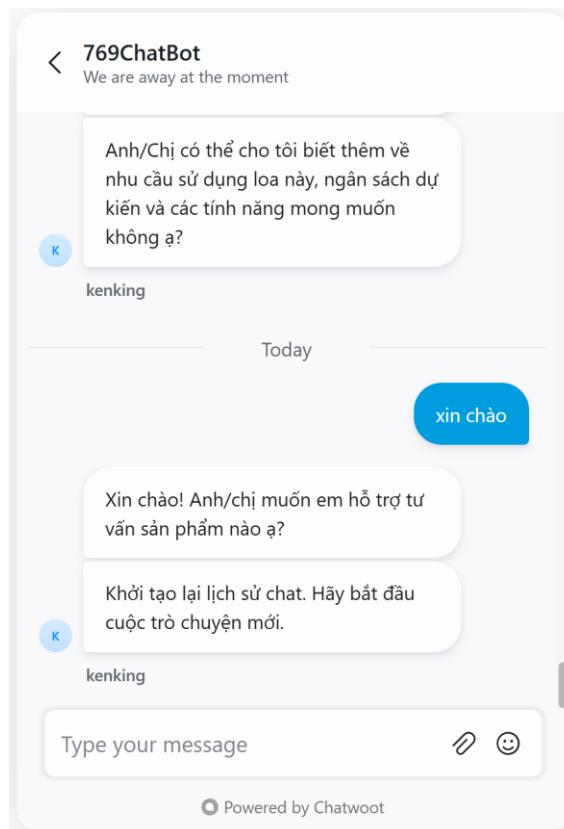
Khách hàng có thể chat trực tiếp với ChatBot thông qua hộp hội thoại bất kỳ câu hỏi hay vấn đề gì, tương tự như việc nhắn tin với nhân viên bán hàng. ChatBot sẽ ghi nhận lịch sử chat, nên khi dùng những từ ở ngôi thứ ba như “nó”, ChatBot sẽ hiểu và trả lời cho chủ thể người dùng và ChatBot đang đề cập



Hình 3.9: Hộp hội thoại giữa khách hàng và ChatBot

3.5.6.2. Xóa lịch sử hội thoại

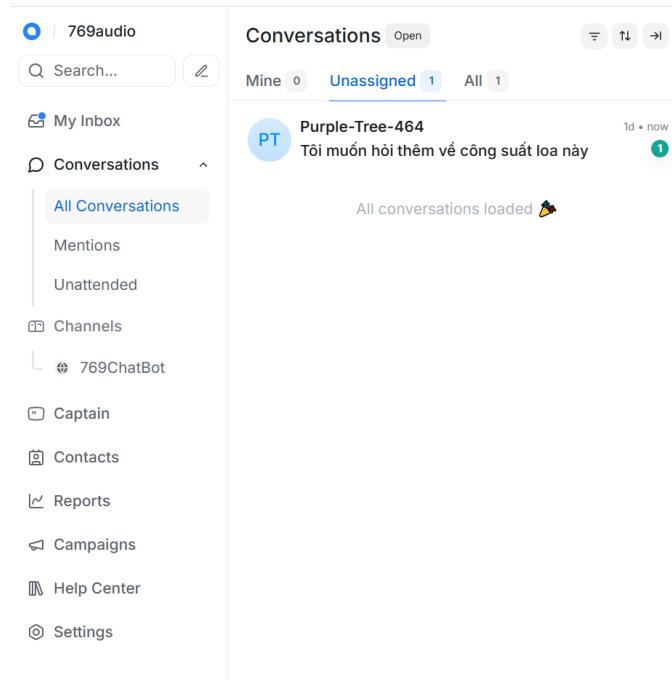
Khi người dùng bấm vào nút ở góc trên bên phải, lịch sử hội thoại sẽ được xóa đi, nhưng thông tin ngữ cảnh về người dùng vẫn được giữ lại để chatbot tư vấn chính xác hơn cho lần chat sau.



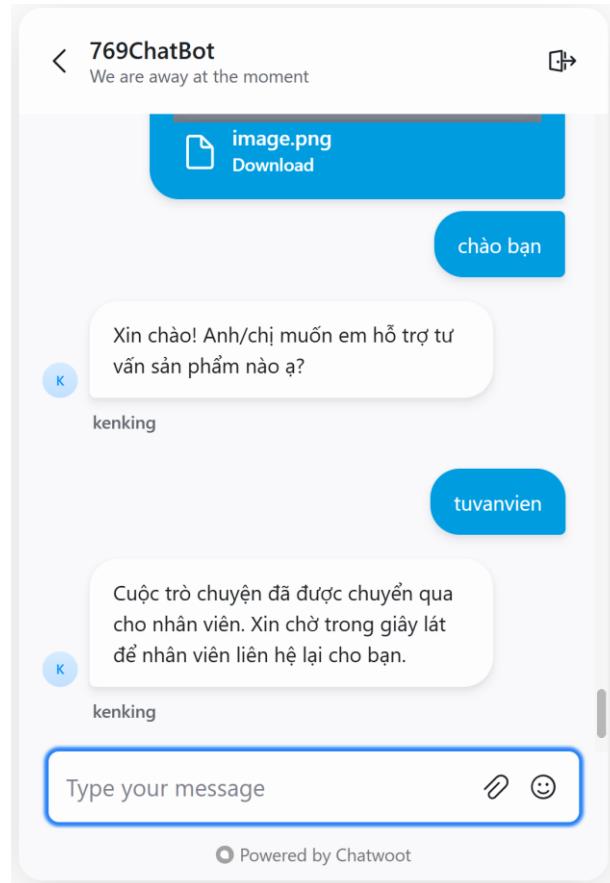
Hình 3.10: Khi xóa cuộc hội thoại, thông báo sẽ hiện ra để người dùng bắt đầu cuộc trò chuyện mới

3.5.6.3. Chuyển sang tư vấn viên

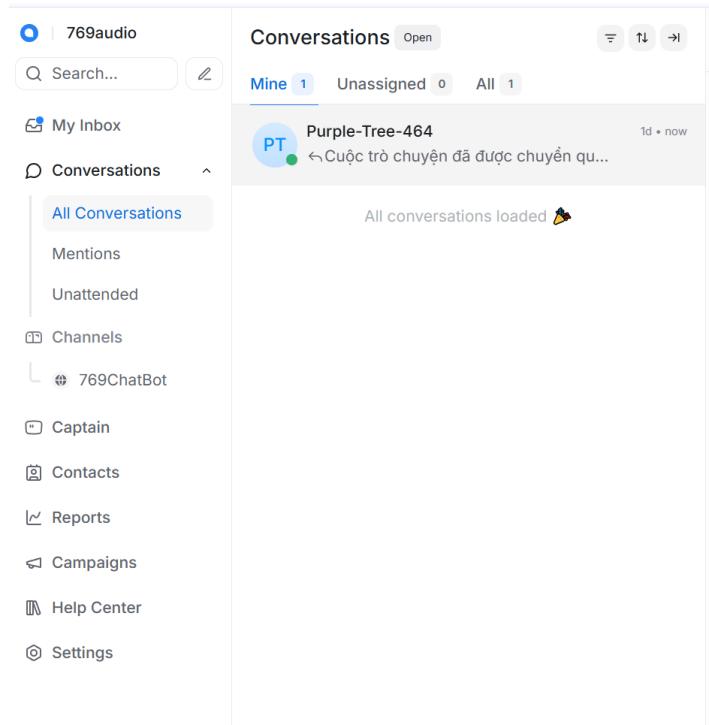
Nếu người dùng không muốn tiếp tục trò chuyện với chatbot, chỉ cần chat vào từ khóa “tuvanvien”, thì cuộc trò chuyện sẽ được chuyển sang chat với tư vấn viên thật, cuộc trò chuyện và thông tin người dùng sẽ có thể được theo dõi để tư vấn viên tiếp tục cuộc trò chuyện



Hình 3.11: Cuộc trò chuyện với Chatbot sẽ không được gán (unassigned) cho tư vấn viên nào



Hình 3.12: Khi người dùng nhập từ khóa “tuvanvien”, cuộc hội thoại được chuyển cho tư vấn viên



Hình 3.13: Khi này, cuộc hội thoại chuyển cho tư vấn viên (gán cho Mine – hộp thoại của tôi)

Người dùng có thể chọn lại nút góc trên bên phải để không chỉ xóa lịch sử chat, mà còn quay lại chat với chatbot.

Chương 4. KẾT LUẬN & HƯỚNG PHÁT TRIỂN

4.1. Kết luận

Thông qua đề tài hệ thống ChatBot tư vấn mua hàng trực tiếp, khóa luận này đã thiết kế một hệ thống Chatbot không chỉ có khả năng giải đáp cơ bản các nhu cầu thiết yếu của khách hàng, từ đó hỗ trợ đáng kể trong việc tư vấn bán hàng, giúp doanh nghiệp tiết kiệm chi phí và nâng cao chất lượng dịch vụ, mà còn có khả năng đặt câu hỏi khai thác nhu cầu tiềm ẩn, dẫn dắt luồng trò chuyện tới việc chốt đơn hiệu quả hơn. Ngoài ra, đề tài còn áp dụng kiến trúc AI Agents, giúp hệ thống được phân chia nghiệp vụ rõ ràng, xử lý thông tin ngữ cảnh tốt hơn, cá nhân hóa trải nghiệm, cũng như dễ dàng tích hợp các Agents khác mà không bị ảnh hưởng tới hệ thống.

Hệ thống đã tích hợp kiến trúc AI Agents điều phối công việc cụ thể, giúp suy luận các bước cần thiết cho một chu trình trao đổi, cho phép Chatbot tư vấn phản hồi chính xác cho đa dạng các yêu cầu tư vấn, đồng thời chủ động đặt câu hỏi khai thác thông tin khách hàng theo quy trình bán hàng cụ thể. Đồng thời, thu thập và sử dụng các thông tin ngữ cảnh của doanh nghiệp, người dùng thông qua cơ chế bộ nhớ ngữ cảnh, giúp nâng cao tính nhất quán trong ngữ cảnh và nâng cao trải nghiệm người dùng, đồng thời mở rộng linh hoạt các chức năng mới trong tương lai bằng cách bổ sung các Agents mới mà không làm gián đoạn các chức năng hiện hữu của hệ thống.

Tuy nhiên, dự án ChatBot này vẫn gặp phải một số khó khăn cần phải được giải quyết. Đầu tiên, ChatBot vẫn gặp khó khăn trong việc xử lý các câu truy vấn phức tạp do cơ chế truy vấn chưa được khai thác tối ưu, còn hạn chế trong việc truy vết chính xác các dữ liệu mang tính kỹ thuật, yêu cầu so sánh nhiều giữa các thực thể, dẫn đến câu trả không chính xác. Thứ hai, Agent 1 và Agent 2 thường phản hồi khá lâu do phải xử lý lượng lớn thông tin ngữ cảnh và thông tin sản phẩm trong prompt, gây gián đoạn trải nghiệm khách hàng. Thứ ba, kiến trúc đa agent tuy linh hoạt và đa nhiệm, nhưng việc sử dụng nhiều lần gọi LLMs thường khiến độ trễ cho phản hồi bị kéo dài, cũng như gây phát sinh chi phí token. Ngoài ra, hệ thống được xây dựng trên flask - framework khá đơn giản, nên việc triển khai trên môi trường thực tế sẽ kém hiệu quả hơn, khi có thể còn thiếu hệ thống bảo mật, cơ sở hạ tầng đủ mạnh mẽ, và dễ xảy ra các nút thắt cho hệ thống.

4.2. Hướng phát triển

Khóa luận đã xây dựng hệ thống Chatbot đa agent hỗ trợ tư vấn mua hàng trực tuyến, cho phép tư vấn chính xác các yêu cầu thiết yếu, chủ động đặt câu hỏi khai thác thông tin khách hàng, giúp nâng cao chất lượng dịch vụ, giảm chi phí vận hành cho doanh nghiệp. Dù thế, hệ thống còn nhiều khuyết điểm về độ chính xác khi gặp các câu hỏi phức tạp, thời gian phản hồi chưa quá nhanh, cũng như hệ thống tổng thể còn đơn giản, chưa thể triển khai trên môi trường thực tế với lượng truy cập lớn.

Trong thời gian tới, đề tài ChatBot tư vấn bán hàng sẽ tiếp tục được phát triển, dựa trên những gì đề tài này chưa đạt được, cũng như các nhu cầu đưa vào doanh nghiệp thực tế:

Đầu tiên, tiếp tục cải tiến mô hình, kiểm tra phản hồi của bot với lượng dữ liệu lớn hơn và đồng đều các nhãn hơn.

Bên cạnh đó, hệ thống cần tối ưu prompt, các bộ nhớ ngữ cảnh, nhằm cải thiện tính cá nhân hóa cho phản hồi, đồng thời giảm độ trễ và chi phí token cho mỗi lần gọi LLM.

Nâng cấp framework và cơ sở hạ tầng để tương thích tốt hơn với nhiều nền tảng và đạt hiệu năng tốt cho môi trường thực tế với số lượng truy cập cùng lúc lớn.

Và không thể thiếu, Tích hợp các chức năng bảo mật và phân quyền, giúp bảo vệ thông tin người dùng, cũng như đảm bảo khả năng scale linh hoạt cho hệ thống.

TÀI LIỆU THAM KHẢO

- [1] G. Calderini and S. J. a. K. McGarry, "A Literature Survey of Recent Advances in Chatbots," 17 January 2022. [Online]. Available: <https://arxiv.org/pdf/2201.06657.pdf>. [Accessed 05 May 2025].
- [2] A. F. C. T. C. A. B. Isabel Kathleen Fornell Haugeland, "Understanding the user experience of customer service chatbots: An experimental study of chatbot interaction design," *International Journal of Human - Computer Studies*, vol. 161, p. 102788, 2022.
- [3] C. S. Ivan Belcic, "AI Agents in 2025: Expectations vs. Reality," Microsoft, 4 March 2025. [Online]. Available: <https://www.ibm.com/think/insights/ai-agents-2025-expectations-vs-reality>. [Accessed 5 May 2025].
- [4] LangChain, "LangChain," [Online]. Available: <https://www.langchain.com/>. [Accessed 5 May 2025].
- [5] "Build a Retrieval Augmented Generation (RAG) App," [Online]. Available: <https://python.langchain.com/docs/tutorials/rag/>. [Accessed 10 2024].
- [6] OpenAI, "GPT-4o mini: advancing cost-efficient intelligence," 18 July 2024. [Online]. Available: <https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/>. [Accessed 5 May 2025].
- [7] Z. J. N. C. Pengyu Zhao, "An In-depth Survey of Large Language Model-based Artificial Intelligence Agents," 23 Sep 2023. [Online]. Available: <https://arxiv.org/pdf/2309.14365.pdf>. [Accessed 5 May 2025].
- [8] A. Gutowska, "What are AI agents?," IBM, 3 July 2024. [Online]. Available: <https://www.ibm.com/think/topics/ai-agents>. [Accessed 5 May 2025].
- [9] L. Varanasi, "Don't Get Too Excited About AI Agents. They Make a Lot of Mistakes," 17 April 2025. [Online]. Available: <https://www.businessinsider.com/ai-agents-errors-hallucinations-compound-risk-2025-4>. [Accessed 5 May 2025].

- [10] K. Gomez, "Multi-Agent vs Single Agent. Understanding the Limitations of Individual LLM Agents and the Superiority of Multi-Agent Collaboration," 17 June 2024. [Online]. Available: <https://medium.com/@kyeg/multi-agent-vs-single-agent-a72713812b68>. [Accessed 5 May 2025].
- [11] D. J. K. Gill, "Mitigating the Top 10 Vulnerabilities in AI Agents," 11 April 2025. [Online]. Available: <https://www.xenonstack.com/blog/vulnerabilities-in-ai-agents>. [Accessed 5 May 2025].
- [12] "Weaviate Cloud," [Online]. Available: <https://weaviate.io/developers/wcs#serverless>. [Accessed 10 2024].
- [13] Qdrant, "Vector Database Benchmarks," 15 January 2024. [Online]. Available: <https://qdrant.tech/benchmarks>. [Accessed 5 May 2025].
- [14] Qdrant, "Pricing," [Online]. Available: <https://qdrant.tech/pricing/>. [Accessed 5 May 2025].

