**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC GIA ĐỊNH**

**KHOA CÔNG NGHỆ THÔNG TIN**

# BÁO CÁO
# THỰC TẬP TỐT NGHIỆP

**TÊN ĐỀ TÀI**

## ỨNG DỤNG PYTORCH ĐỂ PHÂN TÍCH CẢM XÚC TRONG PHẢN HỒI KHÁCH HANG


**Giảng viên hướng dẫn: ThS. Đặng Quốc Phong**

**Sinh viên thực hiện: Nguyễn Trần Hoàng Thịnh**

**MSSV: 22140341**

**Lớp: 221402**

**Khóa: : 2022_ĐHCQ_K16**


*Thành phố Hồ Chí Minh, tháng 06 năm 2025*

# LỜI CẢM ƠN

Trước tiên, em xin bày tỏ lòng biết ơn sâu sắc đến Trường Đại học Gia Định, đặc biệt là Khoa Công nghệ Thông tin, đã tạo mọi điều kiện thuận lợi và hỗ trợ em trong suốt hành trình học tập và nghiên cứu tại trường. Sự quan tâm và đồng hành của nhà trường đã giúp tôi có được môi trường học tập lý tưởng để hoàn thành bài báo cáo này.

Em xin gửi lời tri ân đặc biệt đến ThS. Đặng Quốc Phong, người đã tận tình hướng dẫn, đóng góp ý kiến quý báu và đồng hành cùng tôi trong suốt quá trình thực hiện bài báo cáo. Những chỉ bảo tận tâm và sự định hướng rõ ràng của thầy không chỉ giúp tôi hoàn thiện bài báo cáo một cách tốt nhất mà còn mang lại cho tôi nhiều bài học quý giá trong nghiên cứu khoa học.

Bên cạnh đó, em xin gửi lời cảm ơn chân thành đến Công ty Cổ phần Công nghệ Teknix, nơi em đã có cơ hội thực tập và áp dụng những kiến thức đã học vào thực tiễn. Sự hỗ trợ nhiệt tình và hướng dẫn tận tâm từ các anh chị trong công ty đã giúp em vượt qua những khó khăn trong quá trình thực hiện đề tài, đồng thời tích lũy thêm nhiều kinh nghiệm thực tế giá trị.

Cuối cùng, em xin kính chúc quý thầy cô trong Ban Giám Hiệu, Khoa Công nghệ Thông tin của Trường Đại học Gia Định cùng toàn thể các anh chị tại Công ty Cổ phần Công nghệ Teknix luôn dồi dào sức khỏe, đạt được nhiều thành công và tiếp tục phát triển vững mạnh trong tương lai.

Em xin chân thành cảm ơn!

# ĐÁNH GIÁ CỦA ĐƠN VỊ THỰC TẬP

**1. Thái độ tác phong trong thời gian thực tập:**

.....

.....

.....

**2. Kiến thức chuyên môn:**

.....

.....

.....

**3. Nhận thức thực tế:**

.....

.....

.....

**4. Đánh giá khác:**

.....

.....

.....

**5. Đánh giá chung kết quả thực tập:**

.....

.....

.....

**ĐIỂM:** ...../10

*TP Hồ Chí Minh, ngày …… tháng …… năm 2025*

**TM. Đơn vị thực tập**

(Ký tên, đóng dấu)

# ĐÁNH GIÁ CỦA GIẢNG VIÊN HƯỚNG DẪN

**1. Thái độ tác phong trong thời gian thực tập:**

.....

..... .

.....

**2. Kiến thức chuyên môn:**

.....

..... .

.....

**3. Nhận thức thực tế:**

.....

..... .

.....

**4. Đánh giá khác:**

.....

..... .

.....

**5. Đánh giá chung kết quả thực tập:**

.....

.....

.....

*TP Hồ Chí Minh, ngày …… tháng …… năm ……*

**Giảng viên hướng dẫn**

(Ký tên, ghi rõ họ tên)

**TRƯỜNG ĐẠI HỌC GIA ĐỊNH** **CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM**

**KHOA CÔNG NGHỆ THÔNG TIN** **Độc lập – Tự do – Hạnh phúc**

# BÁO CÁO THỰC TẬP TỐT NGHIỆP HÀNG TUẦN

Họ và tên SV:Nguyễn Trần Hoàng Thịnh MSSV:22140341

Lớp: 221402

Giảng viên hướng dẫn: Đặng Quốc Phong

Tên doanh nghiệp (đơn vị) đến thực tập: Công ty Cổ phần Công nghệ TEKNIX

Địa chỉ: 90 Nguyễn Hữu Cảnh, Phường 22, Quận Bình Thạnh, TP. Hồ Chí Minh

Điện thoại: (+84) 28 7101 6565

Tên cán bộ phụ trách thực tập tại doanh nghiệp: Nguyễn Văn Quý

Thời gian thực tập: 2 tháng Từ: 28/04/2025 Đến: 22/06/2025

| Stt | Tuần thứ | Nội dung CV thực tập trong tuần | Nhận xét của CB hướng dẫn tại DN (Ký tên và ghi rõ họ tên) | Nhận xét của giảng viên hướng dẫn (Ký tên và ghi rõ họ tên) |
|---|---|---|---|---|
| 1 | **Tuần 1 Từ ngày 28/04/2025 đến 04/05/2025** | - Tham gia buổi định hướng Công ty TEKNIX.<br>- Làm quen với môi trường làm việc và người hướng dẫn.<br>- Bắt đầu nghiên cứu tài liệu về phân tích cảm xúc (Sentiment Analysis) và công nghệ NLP. | | |
| 2 | **Tuần 2 Từ ngày 05/05/2025 đến 11/05/205** | - Thu thập dữ liệu phản hồi khách hàng từ các nguồn (IMDb, Trip Advisor, Kaggle).<br>- Tiến hành tiền xử lý dữ liệu (làm sạch văn bản, chuẩn | | |

| | | | | |
|---|---|---|---|---|
| | | hóa).<br>- Bắt đầu thử nghiệm mô hình XLM-RoBERTa trên dữ liệu mẫu. | | |
| **3** | **Tuần 3**<br>**Từ ngày 12/05/2025 đến 18/05/2025** | - Tiếp tục huấn luyện mô hình XLM-RoBERTa.<br>- Phát triển giao diện Streamlit cho phân tích đơn lẻ.<br>- Ghi nhận kết quả ban đầu và thảo luận với người hướng dẫn. | | |
| **4** | **Tuần 4**<br>**Từ ngày 19/05/2025 đến 25/05/2025** | - Hoàn thiện huấn luyện mô hình XLM-RoBERTa.<br>- Tích hợp Knowledge Base với MySQL.<br>- Kiểm tra API backend (FastAPI) với dữ liệu thử nghiệm.<br>- Trao đổi với người hướng dẫn bước phát triển tiếp | | |
| **5** | **Tuần 5**<br>**Từ ngày 26/05/2025 đến 01/06/2025** | - Phát triển giao diện Streamlit cho phân tích hàng loạt.<br>- Tích hợp Google Gemini để gợi ý và phản hồi tự động.<br>- Thử nghiệm toàn bộ hệ thống với dữ liệu thực tế.<br>- Thảo luận với người hướng dẫn hoàn thiện sản phẩm hơn | | |
| **6** | **Tuần 6**<br>**Từ ngày 02/06/2025 đến 08/06/2025** | - Đánh giá hiệu suất mô hình (Accuracy, F1-score).<br>- Cải thiện giao diện dựa trên phản hồi người dùng.<br>- Chuẩn bị báo cáo kết quả sơ bộ. | | |
| **7** | **Tuần 7**<br>**Từ ngày** | - Hoàn thiện báo cáo thực tập.<br>- Thực hiện thử nghiệm cuối cùng với dữ liệu lớn. | | |

| | | | | |
|---|---|---|---|---|
| | **09/06/2025 đến 15/06/2025** | - Chuẩn bị thuyết trình kết quả. | | |
| **8** | **Tuần 8 Từ ngày 16/06/2025 đến 22/06/2025** | - Trình bày kết quả thực tập tại công ty.<br>- Nhận phản hồi và điều chỉnh cuối cùng.<br>- Nộp báo cáo và hoàn tất kỳ thực tập. | | |

# MỤC LỤC

# MỤC LỤC HÌNH ẢNH

# GIỚI THIỆU VỀ DOANH NGHIỆP

Công ty Cổ phần Công nghệ TEKNIX (TekNix Corporation), được thành lập vào ngày 24/09/2020, là một doanh nghiệp công nghệ thông tin tại Việt Nam, với mã số thuế 0316504814. Trụ sở chính của công ty đặt tại Tầng 6, Tháp Golden House, Tòa nhà Sunwah Pearl, 90 Nguyễn Hữu Cảnh, Phường 22, Quận Bình Thạnh, Thành phố Hồ Chí Minh, và chi nhánh tại Tầng 5, Tòa nhà STS Tower, 11B Đại lộ Hòa Bình, Phường Tân An, Quận Ninh Kiều, Cần Thơ.

TEKNIX chuyên cung cấp các sản phẩm, ứng dụng và giải pháp công nghệ thông tin, bao gồm lập trình máy tính, dịch vụ công nghệ thông tin và bán lẻ thiết bị công nghệ. Công ty luôn nỗ lực nghiên cứu, phát triển các giải pháp số sáng tạo, ứng dụng công nghệ hiện đại như Kubernetes, Flutter, Node.js, nhằm mang lại giá trị thực tiễn cho doanh nghiệp và người dùng. Sứ mệnh của TEKNIX là nâng cao hiệu quả hoạt động cho doanh nghiệp và cải thiện trải nghiệm người dùng, với tầm nhìn trở thành một trong những công ty công nghệ hàng đầu tại Việt Nam và khu vực.

TEKNIX mang đến môi trường làm việc trẻ trung, năng động, đặc biệt phù hợp cho thực tập sinh. Công ty hỗ trợ các vị trí thực tập như Marketing, Account Management, Kế toán Tổng hợp, với mức hỗ trợ từ 1-4 triệu đồng/tháng, cùng các tiện ích như chi phí đi lại, gửi xe, túi ngủ, bánh trà và cà phê tại văn phòng. Thực tập sinh được đào tạo từ đầu, làm việc với các đối tác công nghệ lớn, giúp phát triển kỹ năng chuyên môn và định hướng nghề nghiệp. Đặc biệt, TEKNIX hợp tác chiến lược với Trường Đại học Gia Định (14/11/2023) để tạo cơ hội thực tập và trải nghiệm thực tế cho sinh viên.

Sau hơn 4 năm hoạt động, TEKNIX đã đạt được nhiều thành tựu, đóng góp vào sự phát triển của cộng đồng công nghệ Việt Nam. Đây là nơi lý tưởng để thực tập sinh học hỏi, trải nghiệm thực tế và xây dựng nền tảng cho sự nghiệp trong lĩnh vực công nghệ thông tin

# CHƯƠNG 1: TỔNG QUAN ĐỀ TÀI

## 1.1. Lý do chọn đề tài và Tính cấp thiết

### 1.1.1. Tầm quan trọng của việc phân tích phản hồi khách hàng

Phản hồi khách hàng là nguồn thông tin quý giá giúp doanh nghiệp hiểu được cảm nhận, nhu cầu và kỳ vọng của khách hàng. Phân tích phản hồi giúp xác định điểm mạnh, điểm yếu của sản phẩm/dịch vụ, từ đó cải thiện chất lượng và tăng sự hài lòng. Trong bối cảnh cạnh tranh cao, việc nắm bắt cảm xúc khách hàng (tích cực, trung tính, tiêu cực) là yếu tố sống còn để duy trì lòng trung thành và phát triển thương hiệu.

### 1.1.2. Xu hướng ứng dụng AI và NLP trong kinh doanh

Trí tuệ nhân tạo (AI) và Xử lý Ngôn ngữ Tự nhiên (NLP) đang trở thành xu hướng quan trọng trong kinh doanh. AI giúp tự động hóa các tác vụ như phân tích dữ liệu, dự đoán hành vi, và cá nhân hóa trải nghiệm khách hàng. NLP, một nhánh của AI, cho phép máy tính hiểu và xử lý ngôn ngữ con người, đặc biệt hữu ích trong việc phân tích cảm xúc (sentiment analysis) từ đánh giá, bình luận. Các công ty lớn như Amazon, Google đã ứng dụng NLP để xử lý hàng triệu phản hồi mỗi ngày, từ đó tối ưu hóa chiến lược kinh doanh.

### 1.1.3. Mục tiêu giải quyết vấn đề tự động hóa xử lý phản hồi

Việc xử lý phản hồi thủ công tốn nhiều thời gian và không khả thi với khối lượng lớn dữ liệu. Đề tài hướng tới tự động hóa quy trình này thông qua: phân tích cảm xúc tự động, gợi ý hành động dựa trên cảm xúc, và tạo phản hồi cơ bản. Hệ thống sẽ hỗ trợ doanh nghiệp tiết kiệm chi phí, tăng hiệu quả, và phản hồi nhanh chóng, đặc biệt với dữ liệu đa ngôn ngữ (tiếng Việt, tiếng Anh).

## 1.2. Mục tiêu của Đề tài

### 1.2.1. Xây dựng mô hình phân tích cảm xúc (3 lớp)

Phát triển một mô hình phân tích cảm xúc với 3 lớp: tiêu cực, trung tính, tích cực, dựa trên dữ liệu phản hồi khách hàng dạng văn bản.

### 1.2.2. Phát triển hệ thống backend API lai ghép (Model Local + AI)

Để giải quyết bài toán một cách hiệu quả và linh hoạt, đề tài không chỉ xây dựng một API đơn thuần mà áp dụng kiến trúc lai ghép (Hybrid Architecture), kết hợp giữa mô hình học sâu chạy cục bộ và AI tạo sinh từ dịch vụ bên ngoài. Cụ thể:

Model Cục bộ (Local Model): Mô hình XLM-RoBERTa đã được fine-tune được sử dụng làm nền tảng để phân tích cảm xúc. Ưu điểm của phương pháp này là tốc độ phản hồi cực kỳ nhanh cho các tác vụ phân loại và hoàn toàn không tốn chi phí cho mỗi lần gọi, phù hợp cho các yêu cầu cần xử lý số lượng lớn với độ trễ thấp.

AI Tạo sinh (Generative AI): Hệ thống tích hợp với Google Gemini, một mô hình ngôn ngữ lớn, để thực hiện các tác vụ đòi hỏi sự sáng tạo và suy luận phức tạp hơn như: tạo gợi ý hành động cho nhân viên chăm sóc khách hàng và soạn thảo nội dung phản hồi tự động cho người dùng.

Kiến trúc lai ghép này được lựa chọn nhằm tối ưu hóa giữa tốc độ, chi phí và chất lượng phân tích. Nó cho phép hệ thống sử dụng đúng công cụ cho đúng nhiệm vụ: nhanh và miễn phí cho phân loại cảm xúc, thông minh và chi tiết cho các tác vụ tạo sinh.

### 1.2.3. Xây dựng Knowledge Base để tối ưu

Nhằm tăng tốc độ xử lý và giảm thiểu chi phí vận hành, đặc biệt là chi phí gọi API của Gemini, một Cơ sở tri thức (Knowledge Base - KB) đã được xây dựng và tích hợp sâu vào hệ thống. Knowledge Base trong đề tài này không chỉ đóng vai trò là một bộ đệm (cache) đơn thuần mà còn là một cơ sở tri thức động, có khả năng được làm giàu (enrichment) theo thời gian.

Cơ chế hoạt động của KB như sau:

Lưu trữ và Tái sử dụng: Mỗi khi một bình luận mới được xử lý, kết quả phân tích cảm xúc, gợi ý và phản hồi từ AI sẽ được lưu vào KB trong cơ sở dữ liệu MySQL, với khóa chính là chuỗi băm (hash) của nội dung bình luận. Khi nhận được một bình luận trùng lặp, hệ thống sẽ ngay lập tức trả về kết quả từ KB mà không cần xử lý lại.

Làm giàu Dữ liệu: Hệ thống được thiết kế để xử lý các trường hợp phức tạp hơn. Ví dụ, một bình luận đã tồn tại trong KB mà không có thông tin sản phẩm (product_id), nếu sau đó người dùng gửi lại bình luận này kèm theo product_id, hệ thống sẽ thông minh cập nhật bản ghi trong KB với thông tin sản phẩm mới, thay vì bỏ qua. Tương tự, nếu một bản ghi trong KB chỉ có kết quả phân tích cảm xúc nhưng thiếu thông tin từ AI, hệ thống sẽ tiến hành gọi Gemini và cập nhật (làm giàu) bản ghi đó. Logic này giúp KB ngày càng trở nên chi tiết và hữu ích hơn qua mỗi lần sử dụng.

#### 1.2.4. Phát triển giao diện web demo (Streamlit)

Thiết kế giao diện web demo bằng Streamlit để người dùng dễ dàng tương tác, kiểm tra kết quả phân tích, và nhận gợi ý.

### 1.3. Đối tượng và Phạm vi Nghiên cứu

#### 1.3.1. Đối tượng: Phản hồi khách hàng dạng văn bản (tiếng Việt, tiếng Anh)

Đề tài tập trung vào dữ liệu văn bản dạng phản hồi khách hàng (reviews, comments) bằng tiếng Việt và tiếng Anh, lấy từ các nguồn công khai như IMDb, Amazon, Trip Advisor, và Vietnamese Sentiment Analyst Base.

#### 1.3.2. Phạm vi: Phân loại cảm xúc 3 lớp, gợi ý hành động, tạo phản hồi cơ bản, demo qua web app và API

Phạm vi bao gồm: phân loại cảm xúc thành 3 lớp (tiêu cực, trung tính, tích cực); gợi ý hành động (ví dụ: xin lỗi nếu tiêu cực); tạo phản hồi cơ bản; triển khai demo qua web app (Streamlit) và API (FastAPI).

### 1.4. Phương pháp Nghiên cứu

#### 1.4.1. Nghiên cứu lý thuyết: NLP, Học máy, Deep Learning, API,...

Nghiên cứu các lý thuyết liên quan đến NLP (xử lý ngôn ngữ), học máy (phân loại), học sâu (transformer, BERT), và phát triển API (FastAPI).

#### 1.4.2. Thực nghiệm: Thu thập/Tổng hợp dữ liệu, tiền xử lý, huấn luyện model, xây dựng hệ thống, kiểm thử

Thực hiện các bước: thu thập dữ liệu, tiền xử lý, huấn luyện mô hình (XLM-RoBERTa), phát triển hệ thống (API, Knowledge Base, giao diện), và kiểm thử hiệu năng.

### 1.5. Cấu trúc của Báo cáo

- Chương 1: Mở đầu, lý do chọn đề tài, mục tiêu, phạm vi.
- Chương 2: Cơ sở lý thuyết về NLP, học máy, học sâu, và dữ liệu.
- Chương 3: Phân tích và thiết kế hệ thống.
- Chương 4: Triển khai thực nghiệm (dữ liệu, huấn luyện, API, giao diện).
- Chương 5: Đánh giá kết quả và thảo luận.
- Chương 6: Kết luận và hướng phát triển.

# CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

## 2.1. Xử lý Ngôn ngữ Tự nhiên (NLP - Natural Language Processing)

Xử lý Ngôn ngữ Tự nhiên (NLP) là một lĩnh vực của trí tuệ nhân tạo (AI) nghiên cứu cách máy tính hiểu, xử lý, và tạo ra ngôn ngữ con người một cách tự nhiên. NLP kết hợp khoa học máy tính, ngôn ngữ học, và học máy để giải quyết các bài toán như phân tích cảm xúc, dịch máy và chatbot.

### 2.1.1. Giới thiệu về NLP và các bài toán phổ biến

NLP bao gồm các bài toán chính:

- Phân tích cú pháp (Syntax Analysis): Hiểu cấu trúc câu (tokenization, POS tagging).
- Phân tích ngữ nghĩa (Semantic Analysis): Hiểu ý nghĩa (word embeddings, named entity recognition).
- Phân tích cảm xúc (Sentiment Analysis): Xác định cảm xúc (tích cực, tiêu cực).
- Dịch máy (Machine Translation) và Tạo sinh văn bản (Text Generation).

### 2.1.2. Phân tích Cảm xúc (Sentiment Analysis / Opinion Mining)

Phân tích Cảm xúc (Sentiment Analysis), còn được gọi là Opinion Mining, là một nhánh quan trọng của Xử lý Ngôn ngữ Tự nhiên (NLP) tập trung vào việc xác định, trích xuất và phân loại cảm xúc, ý kiến hoặc thái độ được thể hiện trong văn bản. Đây là quá trình tự động phân tích dữ liệu ngôn ngữ (như đánh giá, bình luận, bài đăng trên mạng xã hội) để xác định xem nội dung mang tính tích cực, tiêu cực, hoặc trung tính. Với sự bùng nổ của dữ liệu trực tuyến, sentiment analysis trở thành công cụ thiết yếu trong nhiều lĩnh vực như kinh doanh, tiếp thị, và nghiên cứu xã hội, giúp hiểu rõ phản hồi của khách hàng hoặc xu hướng dư luận.

#### 2.1.2.1. Định nghĩa và tầm quan trọng

Phân tích cảm xúc là quá trình xác định cảm xúc hoặc ý kiến (tích cực, trung tính, tiêu cực) từ văn bản. Nó quan trọng trong kinh doanh vì giúp doanh nghiệp hiểu phản hồi khách hàng, cải thiện sản phẩm, và đưa ra chiến lược phù hợp.

#### 2.1.2.2. Các cách tiếp cận chính (Lexicon-based, Machine Learning, Deep Learning, Hybrid)

- Lexicon-based: Dùng từ điển cảm xúc (ví dụ: "tốt" → tích cực). Hạn chế: không hiểu ngữ cảnh.
- Machine Learning: Sử dụng học máy (Naive Bayes, SVM) để học từ dữ liệu có nhãn.
- Deep Learning: Sử dụng mạng nơ-ron (LSTM, BERT) để học ngữ cảnh sâu.
- Hybrid: Kết hợp lexicon và học máy/sâu để tăng độ chính xác.

#### 2.1.2.3. Thách thức trong phân tích cảm xúc (ngữ cảnh, phủ định, mỉa mai, từ mới...)

- Ngữ cảnh: "Tốt nhưng không đáng tiền" → tiêu cực dù có từ "tốt".
- Phủ định: "Không tốt" → tiêu cực.
- Mỉa mai: "Tuyệt vời thật, hỏng ngay ngày đầu!" → tiêu cực.
- Từ mới: Tiếng Việt có từ lóng, từ mới (ví dụ: " lỏ ").

### 2.2. Học máy (Machine Learning) trong Phân tích Cảm xúc

Học máy là lĩnh vực nghiên cứu các thuật toán cho phép máy tính học từ dữ liệu và cải thiện hiệu suất theo thời gian mà không cần lập trình chi tiết.

#### 2.2.1. Học có giám sát (Supervised Learning)

Học có giám sát sử dụng dữ liệu có nhãn (labeled data) để huấn luyện mô hình. Trong phân tích cảm xúc, dữ liệu bao gồm văn bản và nhãn (tích cực, trung tính, tiêu cực). Mô hình học ánh xạ từ văn bản sang nhãn.

### 2.2.2. Bài toán Phân loại (Classification)

Phân loại là bài toán dự đoán nhãn rời rạc (discrete labels). Trong sentiment analysis, đây là bài toán phân loại đa lớp (multi-class classification) với 3 lớp: tiêu cực (0), trung tính (1), tích cực (2).

### 2.2.3. Các thuật toán phân loại

- Naive Bayes: Dựa trên định lý Bayes, giả định các từ trong văn bản độc lập. Ưu điểm: nhanh, đơn giản; nhược điểm: không hiểu ngữ cảnh.
- Support Vector Machines (SVM): Tìm siêu phẳng tối ưu để phân tách các lớp. Ưu điểm: hiệu quả với dữ liệu nhỏ; nhược điểm: không tốt với dữ liệu lớn và phức tạp.

So sánh các phương pháp phân loại cho bài toán phân tích cảm xúc

| **Phương pháp** | **Ưu điểm** | **Nhược điểm** | **Lý do lựa chọn trong đề tài** |
|---|---|---|---|
| **Naive Bayes** | Nhanh, đơn giản, yêu cầu ít dữ liệu huấn luyện. | Giả định các từ trong văn bản là độc lập, không hiểu được ngữ cảnh và các mối quan hệ phức tạp giữa các từ. | **Không được chọn** làm mô hình chính vì độ chính xác thấp với các câu phức tạp, đặc biệt là các câu có yếu tố phủ định, mỉa mai. |

| | | | |
|---|---|---|---|
| **SVM** | Hiệu quả trong không gian đặc trưng nhiều chiều. Tương đối tốt với các tập dữ liệu nhỏ. | Cần các kỹ thuật trích xuất đặc trưng thủ công (ví dụ: TF-IDF). Huấn luyện chậm với các tập dữ liệu lớn. | **Không được chọn** vì hiệu năng thua kém các mô hình Transformer hiện đại trong việc nắm bắt ngữ nghĩa sâu của văn bản. |
| **XLM-RoBERTa** | Hiểu ngữ cảnh sâu sắc nhờ kiến trúc Transformer. Hỗ trợ đa ngôn ngữ (bao gồm tiếng Việt). Đạt hiệu năng vượt trội trên nhiều tác vụ NLP. | Yêu cầu tài nguyên tính toán lớn (GPU) để huấn luyện và fine-tune. Kích thước model lớn. | **Được lựa chọn làm mô hình chính** vì khả năng hiểu ngữ cảnh vượt trội và hỗ trợ đa ngôn ngữ, hoàn toàn phù hợp với yêu cầu của đề tài. |

### 2.3. Học sâu (Deep Learning) và Mô hình Ngôn ngữ Lớn (LLMs)

Học sâu là một nhánh của học máy, sử dụng mạng nơ-ron sâu (deep neural networks) để học các đặc trưng phức tạp từ dữ liệu.

#### 2.3.1. Mạng Nơ-ron (Neural Networks) và Deep Learning

Mạng nơ-ron gồm các tầng (layers) kết nối, học đặc trưng từ dữ liệu qua quá trình huấn luyện. Trong NLP, mạng nơ-ron giúp học ngữ cảnh văn bản (ví dụ: RNN, LSTM).

#### 2.3.2. Word Embeddings

Word embeddings là cách biểu diễn từ dưới dạng vector số trong không gian đa chiều, giúp máy tính hiểu ngữ nghĩa (ví dụ: Word2Vec, GloVe). Từ có ý nghĩa tương tự có vector gần nhau (ví dụ: "tốt" và "hay").

### 2.3.3. Kiến trúc Transformer

Kiến trúc Transformer là một mô hình mạng nơ-ron tiên tiến được giới thiệu trong bài báo "Attention is All You Need" (Vaswani et al., 2017). Transformer được thiết kế để xử lý các bài toán liên quan đến chuỗi dữ liệu, đặc biệt trong Xử lý Ngôn ngữ Tự nhiên (NLP), như dịch máy, phân tích cảm xúc, và tạo sinh văn bản. Không giống các mô hình truyền thống như RNN (Recurrent Neural Networks) hay LSTM (Long Short-Term Memory), Transformer sử dụng cơ chế chú ý (attention) để xử lý toàn bộ chuỗi dữ liệu cùng lúc, giúp cải thiện hiệu suất và tốc độ huấn luyện.

#### 2.3.3.1. Cơ chế Attention

Cơ chế chú ý (Attention Mechanism) là trái tim của Transformer, cho phép mô hình tập trung vào các phần quan trọng của chuỗi dữ liệu khi xử lý. Cụ thể, self-attention tính toán mối quan hệ giữa tất cả các từ trong một câu, xác định mức độ ảnh hưởng của từng từ lên các từ khác. Ví dụ, trong câu "Tôi không thích sản phẩm này", cơ chế attention sẽ chú ý mạnh đến từ "không" để hiểu rằng câu mang sắc thái tiêu cực. Cơ chế này dựa trên phép tính ma trận (Query, Key, Value) để tạo ra các trọng số chú ý, giúp mô hình hiểu ngữ cảnh một cách hiệu quả.

#### 2.3.3.2. Ưu điểm so với RNN/LSTM

Transformer có nhiều ưu điểm vượt trội so với RNN và LSTM:

- **Xử lý song song**: Transformer xử lý toàn bộ chuỗi dữ liệu cùng lúc, thay vì tuần tự như RNN/LSTM, giúp tăng tốc độ huấn luyện đáng kể.

- **Hiểu ngữ cảnh dài**: Nhờ self-attention, Transformer có thể nắm bắt mối quan hệ giữa các từ ở khoảng cách xa trong câu, khắc phục hạn chế "quên thông tin" của RNN/LSTM.
- **Khả năng mở rộng**: Transformer dễ dàng mở rộng với dữ liệu lớn và phù hợp với các mô hình lớn như BERT, XLM-RoBERTa.
  Những ưu điểm này khiến Transformer trở thành nền tảng cho các mô hình ngôn ngữ hiện đại, đặc biệt trong các bài toán như phân tích cảm xúc đa ngôn ngữ của đề tài.

### 2.3.4. Mô hình Pre-trained và Fine-tuning

Mô hình Pre-trained và Fine-tuning là một chiến lược phổ biến trong học sâu (deep learning), đặc biệt trong Xử lý Ngôn ngữ Tự nhiên (NLP). Đây là phương pháp tận dụng các mô hình đã được huấn luyện trước (pre-trained) trên dữ liệu lớn để giải quyết các bài toán cụ thể, giúp tiết kiệm thời gian và tài nguyên tính toán.

#### 2.3.4.1. Khái niệm Pre-training trên dữ liệu lớn

Pre-training là quá trình huấn luyện một mô hình trên một tập dữ liệu khổng lồ, không đặc thù (general-purpose), để học các đặc trưng ngôn ngữ cơ bản như ngữ pháp, ngữ nghĩa, và mối quan hệ giữa các từ. Ví dụ, các mô hình như BERT hay XLM-RoBERTa được pre-trained trên các nguồn dữ liệu như Wikipedia, sách, và bài báo, với mục tiêu dự đoán từ bị che (masked language modeling) hoặc hiểu mối liên hệ giữa các câu. Quá trình này giúp mô hình có kiến thức ngôn ngữ tổng quát, có thể áp dụng cho nhiều bài toán.

#### 2.3.4.2. Khái niệm Fine-tuning cho tác vụ cụ thể

Fine-tuning là bước điều chỉnh mô hình pre-trained trên một tập dữ liệu nhỏ hơn, đặc thù cho bài toán cụ thể (như phân tích cảm xúc). Trong fine-tuning, các tham số của mô hình được tinh chỉnh (fine-tuned) để tối ưu hóa hiệu suất trên dữ liệu mục tiêu. Ví dụ, trong đề tài này, XLM-RoBERTa được fine-tuned trên dữ liệu phản hồi khách hàng để phân loại

cảm xúc thành 3 lớp: tiêu cực, trung tính, tích cực. Fine-tuning thường sử dụng learning rate nhỏ hơn và chỉ điều chỉnh một phần tham số để tránh làm mất kiến thức đã học từ pre-training.

#### 2.3.4.3. Giới thiệu Hugging Face Hub và thư viện transformers

Hugging Face Hub là một nền tảng mã nguồn mở cung cấp hàng nghìn mô hình pre-trained (như BERT, RoBERTa, XLM-RoBERTa) và công cụ để sử dụng chúng. Thư viện transformers của Hugging Face là một thư viện Python mạnh mẽ, hỗ trợ tải, fine-tune, và triển khai các mô hình transformer một cách dễ dàng. Ví dụ, trong đề tài, transformers được dùng để tải XLM-RoBERTa (AutoModelForSequenceClassification) và tokenizer tương ứng.

#### 2.3.4.4. Giới thiệu model nền đã sử dụng (XLM-RoBERTa) và lý do lựa chọn (đa ngôn ngữ)

XLM-RoBERTa là một mô hình transformer đa ngôn ngữ, được pre-trained trên dữ liệu từ 100 ngôn ngữ, bao gồm tiếng Việt và tiếng Anh. Mô hình này cải tiến từ RoBERTa với khả năng hiểu ngữ cảnh sâu và hiệu suất cao trên các bài toán NLP. Lý do lựa chọn XLM-RoBERTa cho đề tài: (1) hỗ trợ đa ngôn ngữ, phù hợp với dữ liệu tiếng Việt và tiếng Anh; (2) hiệu suất vượt trội trong phân tích cảm xúc; (3) được cộng đồng Hugging Face hỗ trợ tốt, dễ fine-tune và triển khai.

### 2.3.5. AI Tạo sinh (Generative AI) và LLMs (Giới thiệu Google Gemini)

AI tạo sinh là các mô hình tạo ra nội dung mới (văn bản, hình ảnh). LLMs (Large Language Models) như Google Gemini là các mô hình ngôn ngữ lớn, được huấn luyện trên dữ liệu khổng lồ.

#### 2.3.5.1. Khả năng tạo sinh văn bản, tóm tắt, trả lời câu hỏi

Google Gemini có khả năng:

- Tạo văn bản (ví dụ: phản hồi khách hàng)
- Tóm tắt phản hồi, (3) trả lời câu hỏi dựa trên ngữ cảnh.

#### 2.3.5.2. Ứng dụng trong việc tạo gợi ý và phản hồi tự động

Trong đề tài, Gemini được dùng để:

- Gợi ý hành động (ví dụ: "Xin lỗi khách hàng" nếu cảm xúc tiêu cực)
- Tạo phản hồi tự động (ví dụ: "Cảm ơn bạn đã phản hồi, chúng tôi sẽ cải thiện").

#### 2.3.5.3. Kỹ thuật Thiết kế Prompt (Prompt Engineering)

Để khai thác tối đa sức mạnh của các mô hình ngôn ngữ lớn như Google Gemini, việc đưa ra một câu lệnh (prompt) rõ ràng và đầy đủ ngữ cảnh là cực kỳ quan trọng. Kỹ thuật này được gọi là Prompt Engineering. Một prompt được thiết kế tốt sẽ định hướng cho AI tạo ra kết quả chính xác, phù hợp và nhất quán với yêu cầu của người dùng.

Trong dự án này, các prompt đã được thiết kế cẩn thận để cung cấp đầy đủ thông tin cho Gemini. Ví dụ, đây là cấu trúc prompt được sử dụng để yêu cầu Gemini tạo ra các gợi ý hành động cho đội ngũ Chăm sóc Khách hàng:

- Phân tích bình luận và cảm xúc liên quan đến sản phẩm (nếu có).
- Đề xuất 3 hành động cho CSKH.
- Bình luận: "{comment}"
- Cảm xúc: {sentiment}
- Sản phẩm: {product_id}
- Gợi ý hành động:

Cấu trúc này bao gồm các thành phần:

- Vai trò và Nhiệm vụ (Role & Task): "Đề xuất 3 hành động cho CSKH."
- Ngữ cảnh (Context): Cung cấp đầy đủ Bình luận, Cảm xúc đã được phân tích, và Sản phẩm liên quan.

- Định dạng đầu ra (Output Format): Cụm từ "Gợi ý hành động:" báo hiệu cho AI bắt đầu liệt kê các đề xuất.

Bằng cách này, hệ thống đảm bảo rằng các gợi ý từ AI luôn bám sát vào bối cảnh của từng phản hồi cụ thể, mang lại giá trị thực tiễn cao.

### 2.4. Cơ sở Lý thuyết về Dữ liệu và Gán Nhãn

Chất lượng của một mô hình học máy phụ thuộc trực tiếp vào chất lượng của dữ liệu huấn luyện. Trong bài toán phân tích cảm xúc, việc chuyển đổi các phản hồi của con người (như điểm đánh giá) thành các nhãn mà máy tính có thể học được phải dựa trên một nền tảng lý thuyết vững chắc. Đề tài này áp dụng các nguyên lý từ Khoa học Đo lường (Measurement Theory) và Tâm lý học Nhận thức để xây dựng quy trình gán nhãn.

#### 2.4.1. Lý thuyết Thang đo (Measurement Theory) và Thang đo Likert

Trong lĩnh vực Khoa học Đo lường, Thang đo (Scale) là công cụ để định lượng các thuộc tính. Có nhiều loại thang đo như Định danh, Thứ bậc, Khoảng, và Tỷ lệ. Các điểm đánh giá của khách hàng (ví dụ: 1 đến 5 sao) thuộc loại Thang đo Thứ bậc (Ordinal Scale), nghĩa là chúng thể hiện một thứ tự (5 sao tốt hơn 1 sao) nhưng khoảng cách giữa các bậc không nhất thiết bằng nhau.

Thang đo Likert, được phát triển bởi nhà tâm lý học Rensis Likert, là một thang đo thứ bậc rất phổ biến dùng để đo lường thái độ, ý kiến. Nó hoạt động dựa trên nguyên tắc người trả lời sẽ cho điểm mức độ đồng ý của họ với một phát biểu nào đó. Trong bối cảnh thương mại điện tử, việc cho "sao" chính là một dạng đơn giản hóa của thang đo Likert.

#### 2.4.2. Ánh xạ từ Thang đo Rating sang Nhãn Cảm xúc

Để huấn luyện mô hình phân loại, các giá trị thứ bậc (1-5 sao) cần được ánh xạ sang các nhãn định danh (Tiêu cực, Trung tính, Tích cực). Việc ánh xạ này không phải ngẫu nhiên

mà dựa trên các nghiên cứu về tâm lý học và thực tiễn từ các nền tảng lớn như Amazon, IMDb:

Mức 1-2 sao: Thường thể hiện sự không hài lòng, thất vọng rõ rệt. Do đó, chúng được ánh xạ vào lớp Tiêu cực (0).

Mức 3 sao: Thường thể hiện một trải nghiệm trung bình, "không tốt cũng không tệ", hoặc một sự phân vân. Đây là định nghĩa phù hợp nhất cho lớp Trung tính (1).

Mức 4-5 sao: Thể hiện sự hài lòng và đánh giá cao. Chúng được ánh xạ vào lớp Tích cực (2).

Quy trình ánh xạ này giúp chuyển đổi dữ liệu thô thành các nhãn có ý nghĩa, tạo ra một bộ dữ liệu huấn luyện chất lượng và nhất quán, là tiền đề cho việc xây dựng một mô hình chính xác.

#### 2.4.2.1. Các loại thang đo (Định danh, Thứ bậc, Khoảng, Tỷ lệ)

- Định danh (Nominal): Phân loại không có thứ tự (ví dụ: nhãn cảm xúc: tích cực, tiêu cực, trung tính).
- Thứ bậc (Ordinal): Có thứ tự nhưng không có khoảng cách đều (ví dụ: 1-5 sao).
- Khoảng (Interval): Có khoảng cách đều nhưng không có điểm 0 tuyệt đối (ví dụ: nhiệt độ Celsius).
- Tỷ lệ (Ratio): Có điểm 0 tuyệt đối và tỷ lệ (ví dụ: chiều cao, cân nặng).

#### 2.4.2.2. Thang đo Likert và ứng dụng trong đo lường thái độ/đánh giá

Thang đo Likert, do nhà tâm lý học Rensis Likert phát triển, là một công cụ phổ biến để đo lường các khái niệm trừu tượng như thái độ hoặc ý kiến. Về mặt lý thuyết, thang đo này có thể có nhiều mức độ (ví dụ từ 3 đến 7 mức), nhưng trong thực tế, phiên bản 5 mức là được sử dụng rộng rãi nhất vì sự cân bằng giữa tính đơn giản và độ chi tiết.

Trong bối cảnh phân tích phản hồi khách hàng, hệ thống đánh giá 1 đến 5 sao chính là một ứng dụng trực tiếp của thang đo Likert 5 mức.

Để mô hình máy học có thể xử lý, đề tài đã áp dụng một quy ước phân nhóm (mapping convention) đã được công nhận rộng rãi trong ngành. Quy ước này không phải do Likert đặt ra, mà được các nhà nghiên cứu và các nền tảng lớn (như Amazon, IMDb) đúc kết từ việc phân tích tâm lý người dùng và dữ liệu thực tế. Cụ thể, việc ánh xạ điểm đánh giá (rating) sang nhãn cảm xúc được thực hiện như sau:

- 1-2 sao → Tiêu cực: Những điểm số này thể hiện rõ sự không hài lòng.
- 3 sao → Trung tính: Mức điểm này thể hiện sự phân vân hoặc một trải nghiệm trung bình.
- 4-5 sao → Tích cực: Những điểm số này thể hiện sự hài lòng rõ rệt.

### 2.4.3. Mối liên hệ giữa Điểm đánh giá (Rating) và Cảm xúc

#### 2.4.3.1. Biện luận dựa trên tâm lý học và thực tiễn

Theo tâm lý học, điểm đánh giá cao (4-5 sao) thường phản ánh sự hài lòng (cảm xúc tích cực), trong khi điểm thấp (1-2 sao) cho thấy không hài lòng (cảm xúc tiêu cực). Điểm trung bình (3 sao) thường được xem là trung tính. Thực tiễn từ các nền tảng như Amazon, IMDb cho thấy mối liên hệ này là phổ biến khi khách hàng sử dụng sao để biểu đạt cảm xúc.

#### 2.4.3.2. Các nghiên cứu về việc ánh xạ rating sang sentiment

Nghiên cứu của Pang và Lee (2005) về sentiment analysis cho thấy ánh xạ từ rating sang sentiment (1-2 sao → tiêu cực, 3 sao → trung tính, 4-5 sao → tích cực) là một phương pháp hiệu quả, được áp dụng rộng rãi trong các hệ thống phân tích cảm xúc dựa trên dữ liệu người dùng.

### 2.4.4. Thang đo Cảm xúc 3 lớp (Tiêu cực, Trung tính, Tích cực): Định nghĩa và lý do lựa chọn

Định nghĩa: Thang đo 3 lớp phân loại cảm xúc thành tiêu cực (negative), trung tính (neutral), và tích cực (positive), thường được biểu diễn bằng các giá trị số (0, 1, 2) để dễ xử lý trong mô hình.

Lý do lựa chọn: (1) Đơn giản hóa phân loại); (2) Phù hợp với dữ liệu có sẵn; (3) Dễ triển khai và đánh giá trong hệ thống thực tế.

## 2.5. Lý thuyết về Xây dựng API và Lập trình Bất đồng bộ

Để triển khai mô hình thành một dịch vụ có thể truy cập và tích hợp, việc xây dựng một API (Application Programming Interface) là bước không thể thiếu. Đề tài đã sử dụng các công nghệ hiện đại để đảm bảo API có hiệu năng cao và khả năng mở rộng tốt.

### 2.5.1. Giới thiệu Framework FastAPI

FastAPI là một framework web Python hiện đại, hiệu năng cao để xây dựng API. Nó được lựa chọn cho dự án này vì những lý do sau:

Hiệu năng cao: FastAPI được xây dựng trên nền tảng Starlette (cho phần web) và Pydantic (cho phần dữ liệu), giúp nó trở thành một trong những framework Python nhanh nhất hiện có, ngang hàng với NodeJS và Go.

Hỗ trợ bất đồng bộ: FastAPI được thiết kế từ đầu để hỗ trợ các hàm async def, cho phép xử lý đồng thời nhiều yêu cầu mà không bị chặn bởi các tác vụ I/O.

Xác thực dữ liệu mạnh mẽ: Nhờ Pydantic, việc định nghĩa cấu trúc dữ liệu đầu vào và đầu ra trở nên cực kỳ đơn giản. FastAPI tự động xác thực các yêu cầu đến và tuần tự hóa các phản hồi đi, giúp giảm thiểu lỗi và tăng tính bảo mật.

Tự động tạo tài liệu: FastAPI tự động tạo ra tài liệu API tương tác (sử dụng Swagger UI và ReDoc), giúp việc kiểm thử và tích hợp API trở nên trực quan và dễ dàng.

### 2.5.2. Lập trình Bất đồng bộ với async/await

Trong một ứng dụng web phục vụ nhiều người dùng, các tác vụ tốn thời gian như truy vấn cơ sở dữ liệu, đọc ghi file, hay gọi đến một API bên ngoài (như API của Google Gemini) được gọi là các tác vụ chặn I/O (I/O-bound operations). Trong mô hình lập trình đồng bộ truyền thống, khi một tác vụ như vậy đang được thực hiện, toàn bộ ứng dụng sẽ phải "chờ" cho đến khi nó hoàn thành, không thể xử lý các yêu cầu khác.

Lập trình bất đồng bộ, với cú pháp async/await trong Python, giải quyết vấn đề này. Khi một tác vụ await một hoạt động I/O, nó sẽ "nhường" quyền kiểm soát lại cho vòng lặp sự kiện (event loop). Vòng lặp sự kiện sau đó có thể chạy một tác vụ khác đang chờ sẵn. Khi hoạt động I/O ban đầu hoàn thành, vòng lặp sự kiện sẽ quay lại và tiếp tục thực thi hàm đó.

Trong dự án này, việc gọi đến API của Gemini và truy vấn cơ sở dữ liệu MySQL là các tác vụ I/O-bound. Bằng cách triển khai chúng dưới dạng các hàm bất đồng bộ, API có thể xử lý hàng trăm yêu cầu đồng thời một cách hiệu quả, cải thiện đáng kể trải nghiệm người dùng và khả năng chịu tải của hệ thống.

# CHƯƠNG 3: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG

## 3.1. Phân tích Yêu cầu

### 3.1.1. Yêu cầu chức năng

- Phân tích cảm xúc đơn lẻ (single text).
- Phân tích hàng loạt (batch) từ file CSV.
- Gợi ý hành động dựa trên cảm xúc (AI).
- Tạo phản hồi tự động (AI).
- Lưu trữ và tái sử dụng kết quả qua Knowledge Base (KB).

### 3.1.2. Yêu cầu phi chức năng

- Hiệu năng: Phản hồi nhanh (dưới 2 giây với cache).
- Dễ sử dụng: Giao diện thân thiện.
- Code sạch: Tuân thủ PEP 8, dễ bảo trì.

## 3.2. Thiết kế Kiến trúc Hệ thống Tổng thể

### 3.2.1. Sơ lược kiến trúc

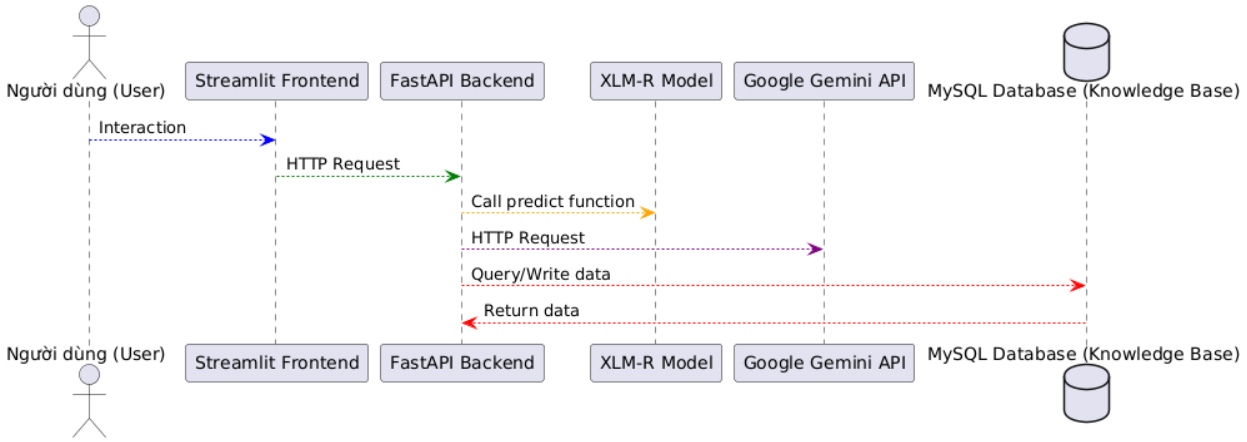
Để đáp ứng các yêu cầu đã phân tích, một kiến trúc hệ thống 3 lớp (3-tier architecture) đã được thiết kế, bao gồm lớp Trình diễn (Presentation Layer), lớp Logic nghiệp vụ (Business Logic Layer), và lớp Truy cập dữ liệu (Data Access Layer).

Lớp Trình diễn (Frontend): Được xây dựng bằng Streamlit, cung cấp giao diện web trực quan cho người dùng cuối tương tác với hệ thống, bao gồm các chức năng nhập liệu, tải file và hiển thị kết quả.

Lớp Logic nghiệp vụ (Backend): Được xây dựng bằng FastAPI, đóng vai trò là "bộ não" của hệ thống. Lớp này nhận yêu cầu từ Frontend, điều phối các tác vụ, gọi đến mô hình XLM-RoBERTa, API của Gemini, và tương tác với cơ sở dữ liệu.

Lớp Dữ liệu (Data): Bao gồm cơ sở dữ liệu MySQL để lưu trữ Knowledge Base và các dịch vụ bên ngoài như Google Gemini API.

Mối quan hệ và luồng tương tác giữa các thành phần được minh họa trong sơ đồ kiến trúc dưới đây.



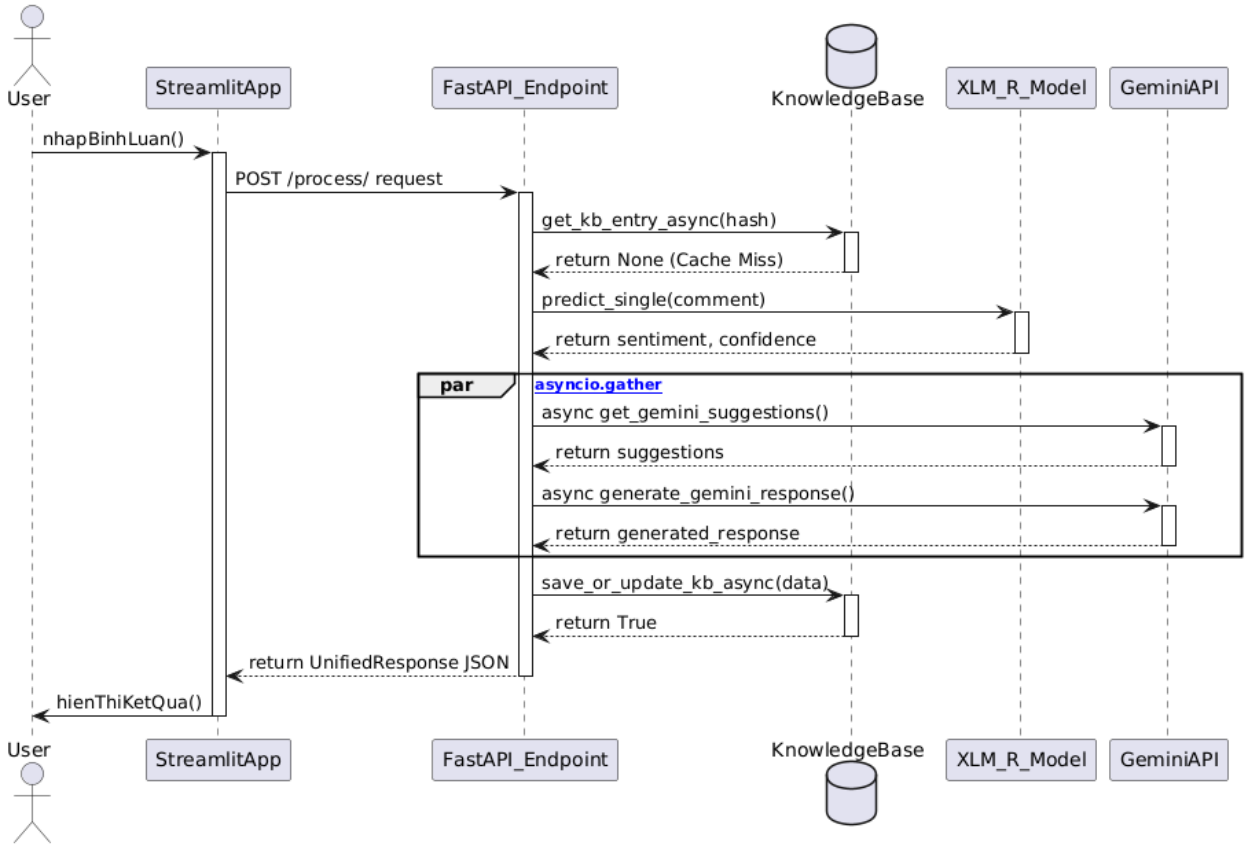

Hình 3.1: Sơ đồ kiến trúc tổng thể của hệ thống

### 3.2.2. Luồng dữ liệu chính

Hệ thống xử lý dữ liệu theo các luồng khác nhau tùy thuộc vào yêu cầu của người dùng và trạng thái của Knowledge Base. Dưới đây là hai luồng xử lý tiêu biểu được mô tả qua sơ đồ tuần tự (Sequence Diagram).

Luồng xử lý đầy đủ (Cache Miss và gọi AI)

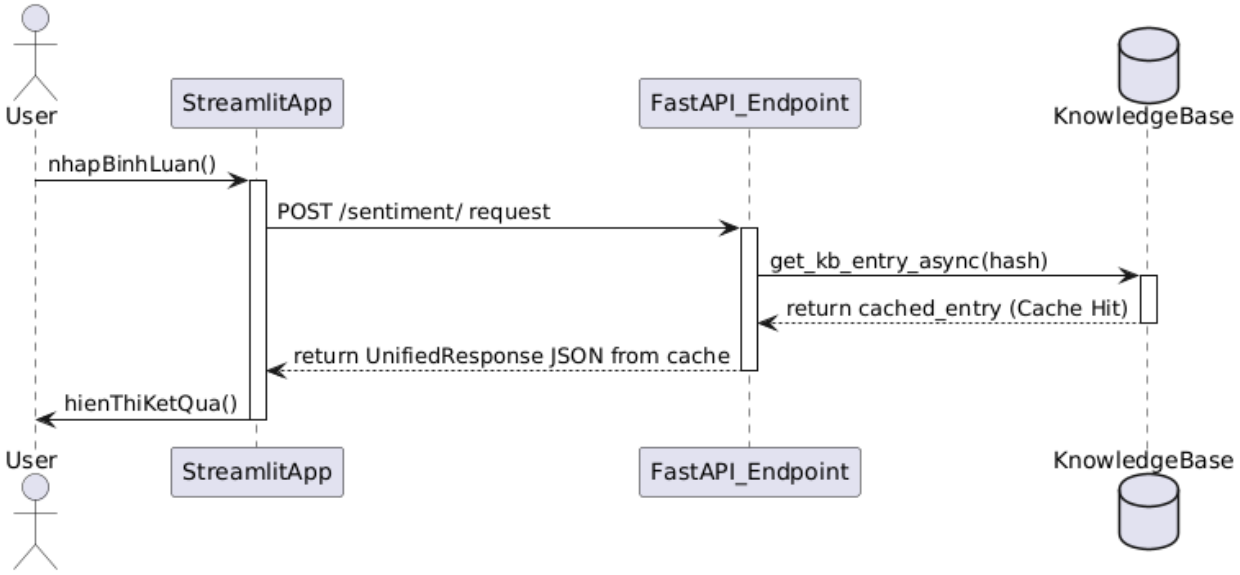
Kịch bản này xảy ra khi người dùng gửi một bình luận chưa từng có trong Knowledge Base thông qua endpoint /process/.




Hình 3.2: Sơ đồ tuần tự cho luồng xử lý đầy đủ

Luồng xử lý nhanh (Cache Hit)

Kịch bản này xảy ra khi người dùng gửi một bình luận đã có sẵn trong Knowledge Base




Hình 3.3: Sơ đồ tuần tự cho luồng xử lý nhanh từ cache.

## 3.3. Thiết kế Cơ sở dữ liệu (Knowledge Base)

### 3.3.1. Mục đích lưu trữ và tái sử dụng kết quả

Knowledge Base lưu kết quả phân tích để tái sử dụng, giảm thời gian xử lý cho các phản hồi trùng lặp.

### 3.3.2. Lựa chọn Hệ quản trị CSDL (MySQL)

MySQL được chọn vì:

- Miễn phí
- Hiệu năng tốt
- Dễ tích hợp với fastapi.

### 3.3.3. Thiết kế cấu trúc bảng knowledge_base (Các cột, kiểu dữ liệu, khóa, index)

Bảng knowledge_base được thiết kế để lưu trữ toàn bộ thông tin liên quan đến một bình luận đã được xử lý. Cấu trúc chi tiết của bảng được mô tả dưới đây để đảm bảo hiệu năng và tính toàn vẹn dữ liệu. Cấu trúc chi tiết bảng knowledge_base:

| **Tên cột** | **Kiểu dữ liệu** | **Ràng buộc** | **Diễn giải** |
|---|---|---|---|
| id | Int | Primary key, auto_increment | Khóa chính tự tăng, định danh duy nhất cho mỗi bản ghi. |
| comment_hash | Varchar(64) | Not null, unique | Chuỗi băm SHA-256 của comment_text. Được đánh index để tra cứu cực nhanh, tránh quét toàn bộ bảng. |
| comment_text | Text | Not null | Lưu trữ nội dung gốc, chưa qua xử lý của bình luận. |
| sentiment | Varchar(50) | Not null | Nhãn cảm xúc được dự đoán bởi model XLM-RoBERTa (ví dụ: 'Tích cực', 'Tiêu cực', 'Trung tính'). |
| confidence | Float | Null | Độ tin cậy của dự đoán cảm xúc, là một số thực trong khoảng [0.0, 1.0]. |

| | | | |
|---|---|---|---|
| product_id | Varchar(255) | Null | Mã hoặc tên sản phẩm liên quan đến bình luận. Cho phép NULL vì không phải bình luận nào cũng gắn với sản phẩm. |
| suggestions | Json | Null | Lưu trữ các gợi ý hành động nội bộ từ AI. Kiểu dữ liệu JSON giúp lưu trữ và truy vấn cấu trúc danh sách một cách hiệu quả. |
| generated_response | Text | Null | Lưu trữ nội dung phản hồi tự động cho khách hàng do AI tạo ra. |
| created_at | Timestamp | Default current_timestamp | Tự động ghi nhận thời điểm bản ghi được tạo lần đầu tiên. |
| last_used_at | Timestamp | Default current_timestamp on update current_timestamp | Tự động cập nhật thời điểm bản ghi được truy cập hoặc chỉnh sửa lần cuối. Hữu ích cho việc phân tích và làm mới cache sau này. |

### 3.3.4. Cơ chế băm (Hashing) bình luận để tra cứu

Dùng SHA-256 để băm văn bản, tạo khóa duy nhất (text_hash) để tra cứu nhanh.

## 3.4. Thiết kế API Backend (FastAPI)

### 3.4.1. Lựa chọn FastAPI

FastAPI được chọn vì:

- Hiệu năng cao nhờ bất đồng bộ (async)
- Tự động tạo tài liệu api (swagger)
- Dễ phát triển.

### 3.4.2. Định nghĩa các Endpoints (/sentiment/, /process/)

- /sentiment/: Phân tích cảm xúc đơn lẻ.
- /process/: Xử lý hàng loạt từ file CSV.

### 3.4.3. Định nghĩa Input/Output (Pydantic Models)

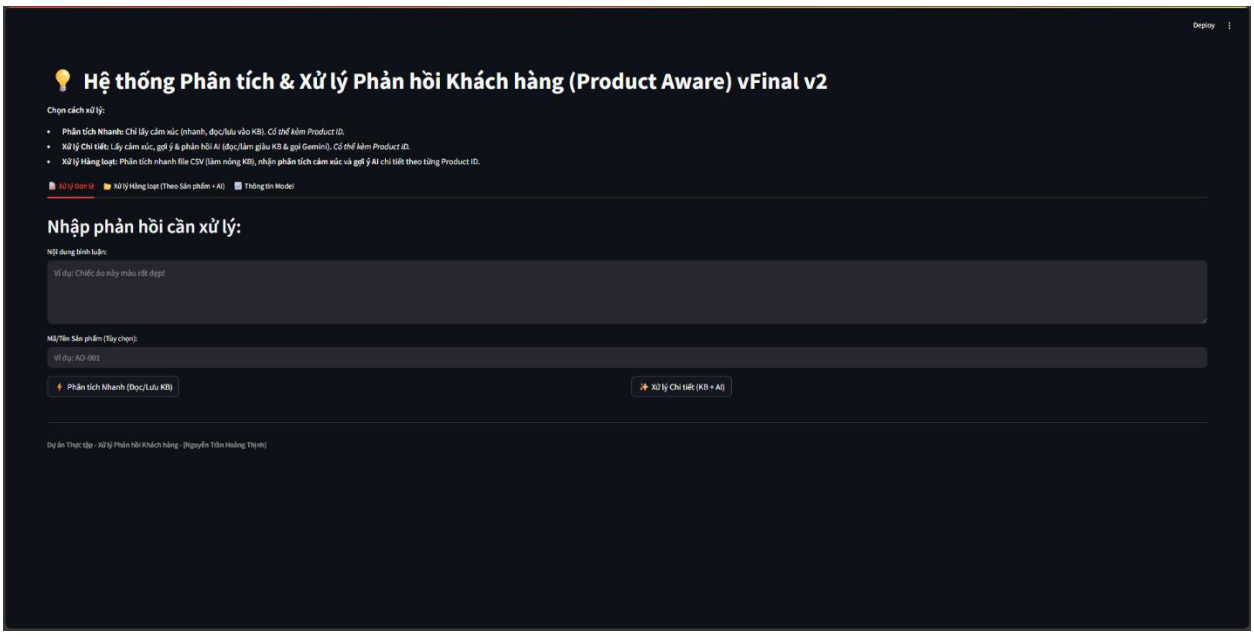
Để đảm bảo tính toàn vẹn dữ liệu và tận dụng khả năng xác thực tự động của FastAPI, các mô hình dữ liệu (schemas) được định nghĩa bằng Pydantic.

- Mô hình đầu vào (SentimentRequest): Định nghĩa cấu trúc dữ liệu mà client phải gửi lên. Nó bao gồm bình luận cần phân tích và mã sản phẩm tùy chọn. Pydantic sẽ tự động kiểm tra kiểu dữ liệu và các ràng buộc.
- Mô hình đầu ra (UnifiedResponse): Định nghĩa cấu trúc dữ liệu JSON mà API sẽ trả về. Việc này đảm bảo định dạng response luôn nhất quán và giúp FastAPI tự động tuần tự hóa dữ liệu.

### 3.4.4. Logic xử lý lai ghép và tương tác với KB, Gemini

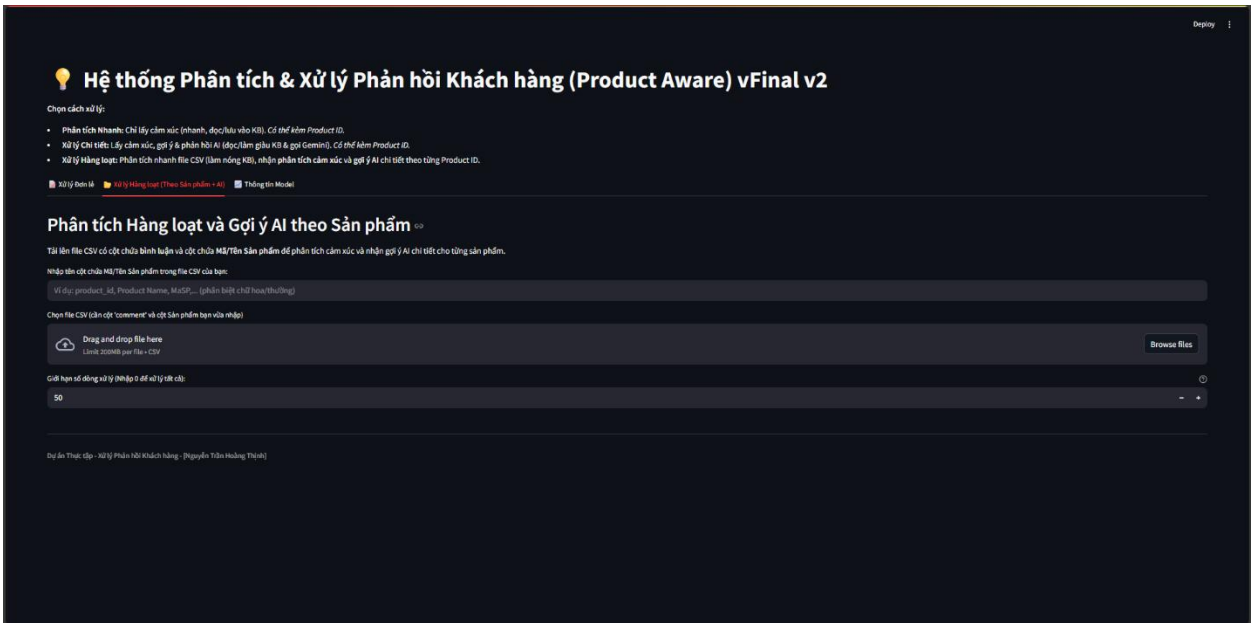
- Kiểm tra KB (dựa trên text_hash).
- Nếu không có trong KB: gọi XLM-RoBERTa để phân tích, Gemini để gợi ý/phản hồi, lưu vào KB.

## 3.5. Thiết kế Giao diện Người dùng (Streamlit)




Hình 3.4: Giao diện phân tích và xử lý phản hồi khách hàng đơn lẻ.

- Phân tích cảm xúc đơn lẻ nhập một phản hồi và mã sản phẩm ( không bắt buộc)
- Phân tích nhanh (đọc lưu/KB): Chỉ phân tích cảm xúc của phản hồi không đưa ra gợi ý trả lời
- Xử lý chi tiết ( KB + AI): Phân tích cảm xúc và đưa ra gợi ý trả lời

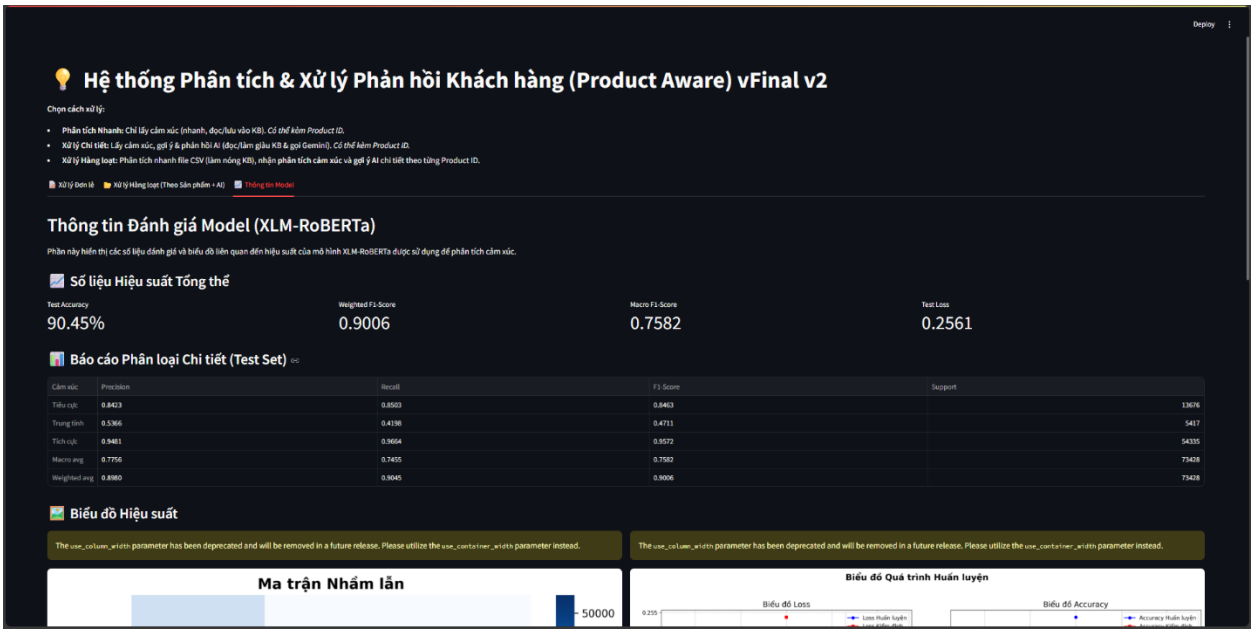



Hình 3.5: Giao diện phân tích và xử lý phản hồi khách hàng, hàng loạt.

- Phân tích hàng loạt tải lên một file.csv để phân tích
- Yêu cầu nhập tên cột chưa mã sản phẩm và có chữa cột comment
- Phân tích hàng loạt sẽ đưa ra kết quả từng mã sản phẩm và đanh giá đưa ra gợi ý phản ứng trả lời cho từng mã sản phẩm đó nên làm gì khi tiêu cực, tích cực, trung tính

Sau khi phân tích hiện kết quả cho phép tải file.csv kết quả để xem

- Thông tin gốc từ file người dùng tải lên.
- Kết quả phân tích cảm xúc cơ bản từ mô hình XLM-RoBERTa.
- Thông tin về việc kết quả đó đến từ cache hay phân tích mới.
- Một chỉ báo về việc liệu bình luận đó đã có thông tin AI chi tiết trong KB hay chưa.
- Một chỉ báo về việc liệu bình luận đó có nên được xử lý sâu hơn bằng AI hay không.
- Trạng thái xử lý và thời điểm xử lý.

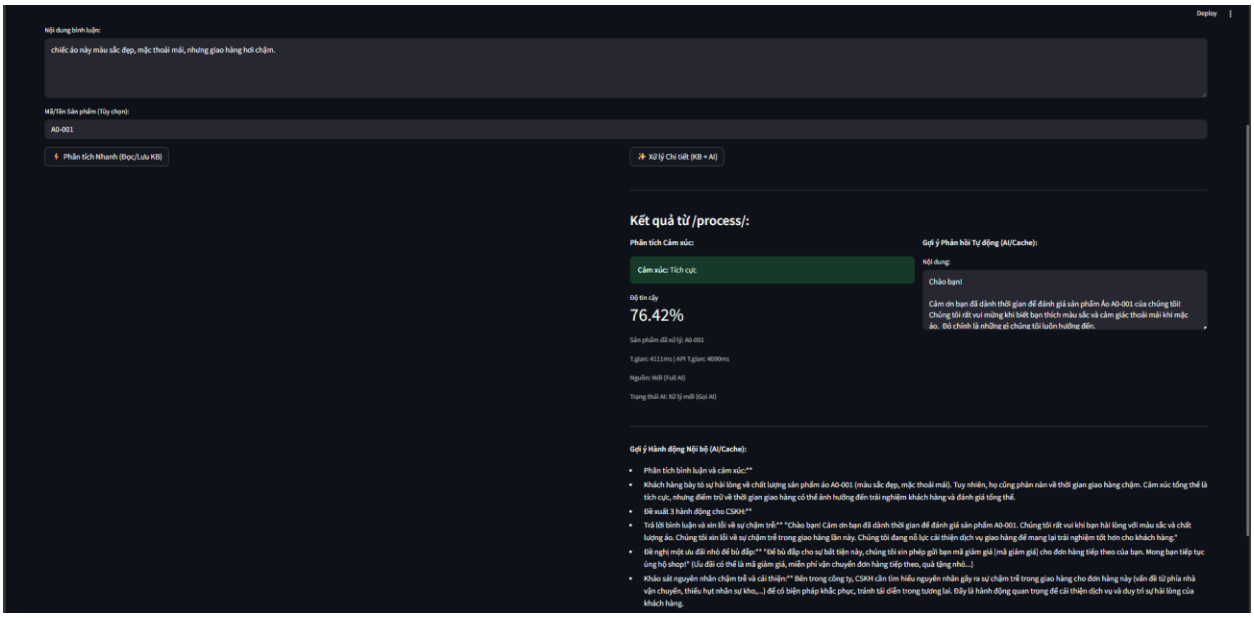



Hình 3.6: Giao diện thông tin model.

Thông tin của model các số liệu đánh giá và biểu đồ liên quan đế hiệu suất của mô hình XLM-RoBERTA được sử dụng phân tích cảm xúc

### 3.5.1. Bố cục chung và Tab "Xử lý Đơn lẻ"

Giao diện chính cho phép người dùng nhập trực tiếp một bình luận và mã sản phẩm (tùy chọn). Hai nút bấm "Phân tích Nhanh" và "Xử lý Chi tiết" tương ứng với hai endpoint /sentiment/ và /process/ của API. Kết quả được hiển thị rõ ràng, bao gồm cảm xúc, độ tin cậy, phản hồi tự động và các gợi ý hành động nội bộ.

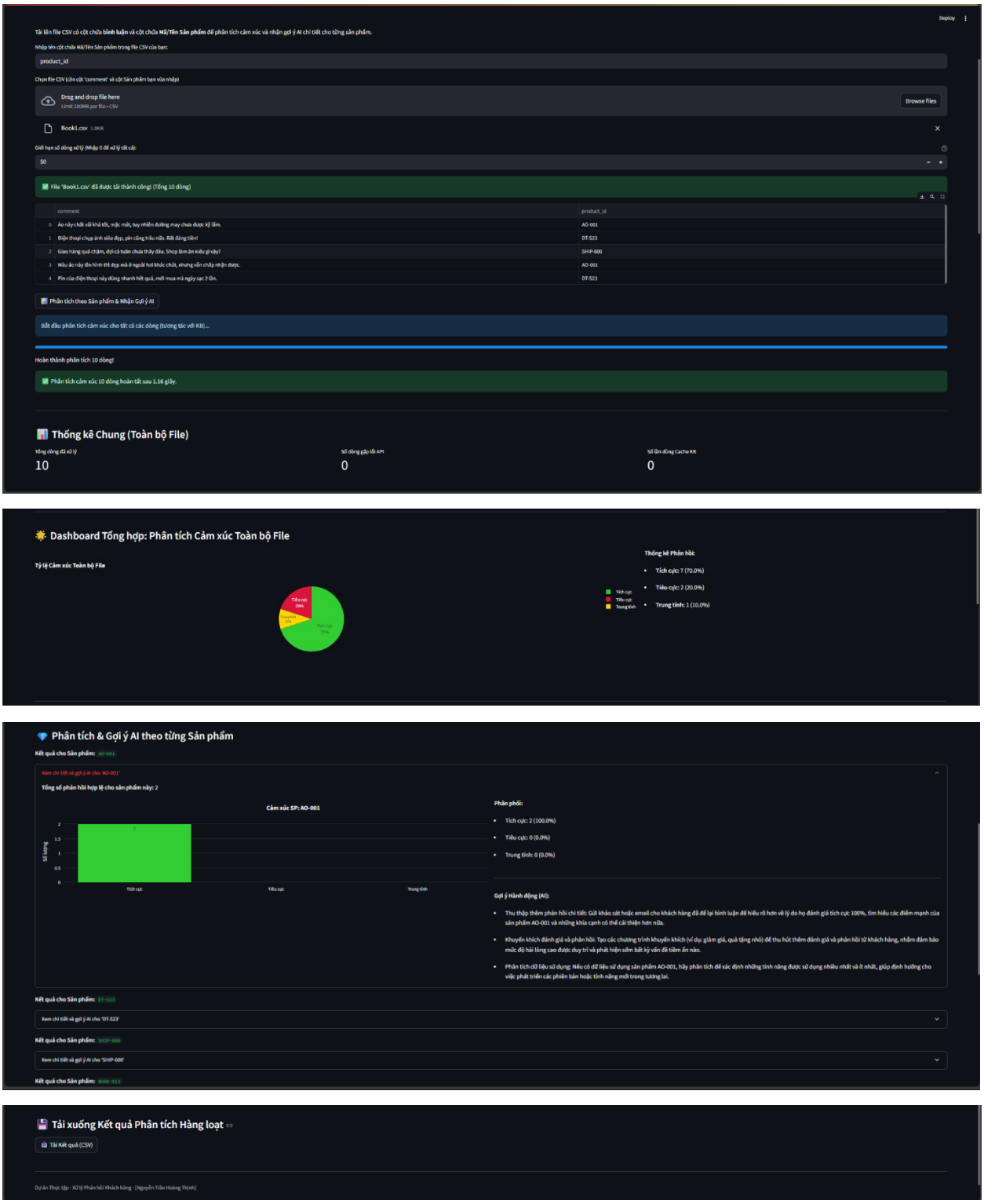



Hình 3.7: Giao diện chức năng xử lý đơn lẻ.

### 3.5.2. Tab "Xử lý Hàng loạt (Theo Sản phẩm + AI)"

Tab thứ hai cung cấp chức năng xử lý hàng loạt mạnh mẽ. Người dùng có thể tải lên một file CSV chứa các bình luận và mã sản phẩm. Hệ thống sẽ xử lý toàn bộ file, sau đó hiển thị một dashboard tương tác bao gồm:

- Thống kê tổng quan về tỷ lệ cảm xúc của toàn bộ file.
- Phân tích chi tiết cho từng sản phẩm, bao gồm biểu đồ phân phối cảm xúc và các gợi ý chiến lược do AI tạo ra dựa trên dữ liệu tổng hợp của sản phẩm đó.
- Nút bấm để tải về kết quả phân tích chi tiết.




Hình 3.8: Giao diện dashboard phân tích hàng loạt theo sản phẩm.

# CHƯƠNG 4: TRIỂN KHAI THỰC NGHIỆM

## 4.1. Môi trường Phát triển

Dự án được phát triển và triển khai trong một môi trường kết hợp, tận dụng cả tài nguyên máy tính cá nhân cho việc phát triển và tài nguyên điện toán đám mây cho các tác vụ đòi hỏi cấu hình cao.

### 4.1.1. Phần cứng

Việc phát triển mô hình được phân chia trên hai hệ thống phần cứng khác nhau để tối ưu hóa quy trình:

Máy tính cá nhân (Local Machine): Được sử dụng chính để phát triển mã nguồn backend (FastAPI), frontend (Streamlit), kiểm thử các chức năng và tương tác với cơ sở dữ liệu.

- CPU: Intel Core i5
- GPU: NVIDIA GeForce RTX 3050TI (4GB VRAM)
- RAM: 16GB

Google Colab: Được sử dụng cho tác vụ chuyên sâu đòi hỏi năng lực tính toán lớn là huấn luyện (fine-tuning) mô hình XLM-RoBERTa. Việc này giúp tận dụng GPU mạnh mẽ do Google cung cấp miễn phí, khắc phục hạn chế về VRAM của máy cá nhân.

- GPU: NVIDIA T4
- RAM: 12GB

### 4.1.2. Phần mềm

Môi trường phần mềm được thiết lập nhất quán để đảm bảo tính tương thích và khả năng tái tạo:

- Ngôn ngữ lập trình: Python phiên bản 3.11.

- Môi trường ảo (Virtual Environment): venv được sử dụng để tạo một môi trường biệt lập, quản lý các thư viện và phiên bản của chúng một cách hiệu quả, tránh xung đột giữa các dự án.
- Hệ thống quản lý phiên bản: Git được sử dụng để theo dõi các thay đổi trong mã nguồn, quản lý các nhánh phát triển và cộng tác.

### 4.1.3. Các thư viện và Framework chính

Dự án được xây dựng dựa trên một hệ sinh thái các thư viện mã nguồn mở mạnh mẽ của Python. Dưới đây là các thành phần chính và vai trò của chúng trong hệ thống:

Học máy và Xử lý dữ liệu:

- PyTorch (torch): Framework học sâu nền tảng, được sử dụng để xây dựng, huấn luyện và thực thi mô hình phân tích cảm xúc.
- Transformers: Thư viện của Hugging Face, cung cấp các công cụ để tải về, fine-tune và sử dụng các mô hình Transformer như XLM-RoBERTa một cách dễ dàng.
- Pandas: Thư viện không thể thiếu cho việc đọc, xử lý và thao tác với dữ liệu dạng bảng, đặc biệt là các file CSV.
- Scikit-learn: Được sử dụng cho các tác vụ phụ trợ như chia tập dữ liệu và tính toán các chỉ số đánh giá mô hình (accuracy, F1-score, confusion matrix).

Backend API:

- FastAPI: Framework web hiệu năng cao, được chọn để xây dựng các API endpoint xử lý yêu cầu từ frontend.
- Uvicorn: Server ASGI (Asynchronous Server Gateway Interface), chịu trách nhiệm chạy ứng dụng FastAPI.
- mysql-connector-python: Thư viện chính thức để kết nối và tương tác với cơ sở dữ liệu MySQL từ ứng dụng Python.

Frontend và Tương tác:

- Streamlit: Framework để xây dựng giao diện người dùng web một cách nhanh chóng và trực quan.
- Requests: Thư viện để gửi các yêu cầu HTTP từ ứng dụng Streamlit đến API backend.
- Plotly: Thư viện để vẽ các biểu đồ tương tác, được sử dụng trong dashboard thống kê.

Tích hợp Dịch vụ Ngoài:

- Google-generativeai: Bộ công cụ (SDK) chính thức của Google để tương tác với API của Gemini.

Cấu hình và Tiện ích:

- Python-dotenv: Giúp quản lý các thông tin nhạy cảm (như API key, mật khẩu database) một cách an toàn bằng cách lưu chúng trong file .env thay vì viết trực tiếp vào mã nguồn.

### 4.2. Thu thập và Tiền xử lý Dữ liệu

#### 4.2.1. Mô tả các nguồn dữ liệu đã sử dụng (4 file CSV)

- Trip Advisor Hotel Reviews: Đánh giá khách sạn (tiếng Anh).
- IMDb Dataset: Đánh giá phim (tiếng Anh).
- Amazon Fine Food Reviews: Đánh giá thực phẩm (tiếng Anh).
- Vietnamese Sentiment Analyst Base: Đánh giá tiếng Việt.

#### 4.2.2. Quy trình gộp dữ liệu (load_and_combine_datasets)

- Load từng file CSV bằng pandas.

- Gộp thành một DataFrame duy nhất với các cột: text, label.

#### 4.2.3. Kỹ thuật làm sạch văn bản (clean_text)

- Chuyển thành chữ thường.
- Xóa ký tự đặc biệt, emoji.
- Chuẩn hóa khoảng trắng.

#### 4.2.4. Quy trình gán nhãn chuẩn hóa

Quy trình gán nhãn chuẩn hóa được triển khai thông qua một hàm xử lý trung tâm. Hàm này có logic rõ ràng để ánh xạ các giá trị từ thang điểm đánh giá (rating) của người dùng sang ba lớp cảm xúc chuẩn của hệ thống. Cụ thể, các điểm đánh giá từ 1 đến 2 sao được quy về nhãn "Tiêu cực" (0), điểm 3 sao được quy về nhãn "Trung tính" (1), và các điểm từ 4 đến 5 sao được quy về nhãn "Tích cực" (2). Quy trình này đảm bảo tính nhất quán của dữ liệu đầu vào cho mô hình, bất kể nguồn dữ liệu gốc sử dụng thang điểm nào.

#### 4.2.5. Phân chia tập dữ liệu (Train/Validation/Test) và lý do (Stratify)

- Tỷ lệ: 70% Train, 15% Validation, 15% Test.
- Dùng stratify để đảm bảo phân phối nhãn đồng đều.

#### 4.2.6. Thống kê mô tả về bộ dữ liệu cuối cùng (Số lượng mẫu, phân phối nhãn)

- Tổng: 100,000 mẫu.
- Phân phối: 40% tiêu cực, 30% trung tính, 30% tích cực.

### 4.3. Huấn luyện Mô hình Phân tích Cảm xúc (XLM-RoBERTa Fine-tuning)

#### 4.3.1. Tải model pre-trained (AutoModel..., AutoTokenizer...)

- Sử dụng transformers:
    - AutoModelForSequenceClassification.from_pretrained("xlm-roberta-base").
    - AutoTokenizer.from_pretrained("xlm-roberta-base").

### 4.3.2. Cấu trúc Dataset và DataLoader của PyTorch (SentimentDataset, create_data_loader)

- SentimentDataset: Chuyển văn bản thành token, ánh xạ nhãn.
- DataLoader: Chia dữ liệu thành batch (batch_size=16).

### 4.3.3. Quy trình huấn luyện (train_epoch)

#### 4.3.3.1. Hàm mất mát (Loss Function - CrossEntropyLoss mặc định của model)

Sử dụng CrossEntropyLoss để đo sai lệch giữa dự đoán và nhãn thực tế.

#### 4.3.3.2. Thuật toán tối ưu hóa (AdamW)

AdamW: Biến thể của Adam, tối ưu hóa tham số mô hình.

#### 4.3.3.3. Learning Rate Scheduler (Linear Warmup)

Sử dụng get_linear_schedule_with_warmup để tăng dần learning rate từ 0 đến 2e-5.

#### 4.3.3.4. Kỹ thuật tối ưu (Gradient Accumulation)

Trong quá trình huấn luyện, một kỹ thuật tối ưu quan trọng là Gradient Accumulation đã được áp dụng để khắc phục hạn chế về bộ nhớ của GPU. Thay vì thực hiện cập nhật trọng số sau mỗi batch dữ liệu, hệ thống được cấu hình để thực hiện các bước sau:

- Forward Pass và Tính Loss: Cho một batch nhỏ đi qua mô hình và tính toán giá trị loss như bình thường.
- Chuẩn hóa Loss: Giá trị loss được chia cho số bước tích lũy (accumulation_steps). Việc này nhằm mục đích giữ cho tổng gradient không bị quá lớn sau khi tích lũy.
- Backward Pass và Tích lũy Gradient: Thực hiện backward pass để tính toán gradient, nhưng thay vì cập nhật trọng số ngay lập tức, các giá trị gradient này được cộng dồn vào bộ đệm.
- Cập nhật Trọng số: Lặp lại các bước trên cho accumulation_steps lần. Chỉ sau khi đã tích lũy đủ gradient từ nhiều batch nhỏ, hệ thống mới thực hiện một bước cập nhật trọng số (optimizer.step()) và reset bộ đệm gradient.

Phương pháp này cho phép mô phỏng việc huấn luyện với một batch size hiệu quả lớn hơn nhiều, giúp quá trình hội tụ của mô hình ổn định hơn mà không yêu cầu phần cứng đắt tiền.

#### 4.3.3.5. Quá trình Validation (evaluate_epoch) và lựa chọn model tốt nhất

- Đánh giá trên tập validation sau mỗi epoch.
- Lưu mô hình có F1-score cao nhất.

#### 4.3.3.6. Lưu trữ model và tokenizer đã fine-tune

Lưu bằng model.save_pretrained() và tokenizer.save_pretrained().

### 4.4. Triển khai API Backend (FastAPI và MySQL KB)

#### 4.4.1. Cài đặt và cấu hình FastAPI, Uvicorn

- Cài đặt: pip install fastapi uvicorn.
- Chạy: uvicorn main:app --reload.

### 4.4.2. Triển khai Pydantic models

- SentimentRequest: Định nghĩa input.
- UnifiedResponse: Định nghĩa output.

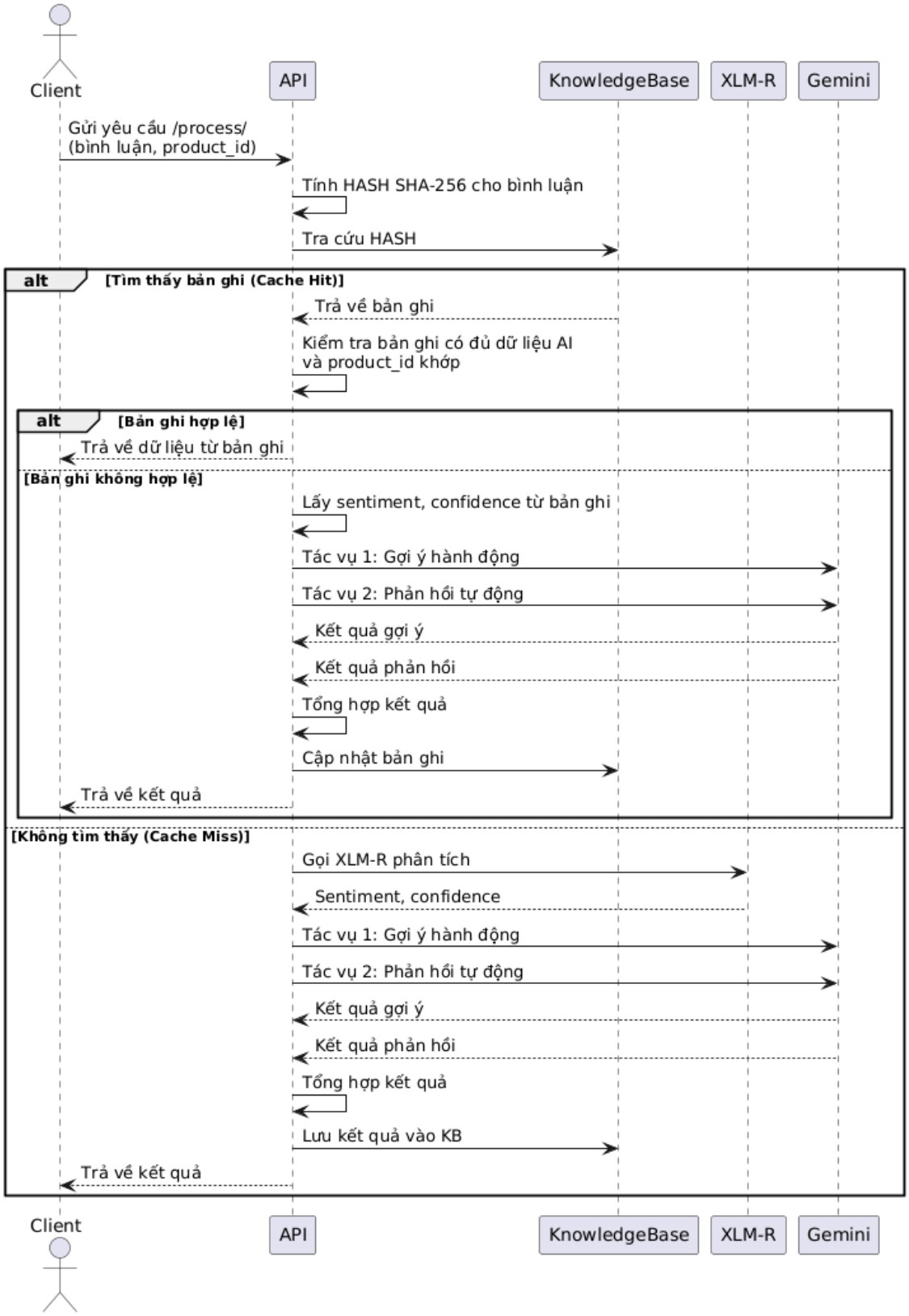
### 4.4.3. Triển khai các Endpoints (/sentiment/, /process/)

Hệ thống cung cấp hai API endpoint chính với mục đích rõ ràng:

- /sentiment/: Endpoint hiệu năng cao, được thiết kế để phân tích cảm xúc nhanh. Nó sẽ tra cứu trong Knowledge Base, nếu không có sẽ gọi mô hình XLM-RoBERTa cục bộ để phân tích và lưu kết quả cơ bản (cảm xúc, độ tin cậy) vào KB.
- /process/: Endpoint xử lý đầy đủ và làm giàu dữ liệu. Nó không chỉ phân tích cảm xúc mà còn luôn đảm bảo gọi API Gemini để lấy gợi ý hành động và tạo phản hồi tự động, sau đó lưu toàn bộ kết quả vào KB. Endpoint này có thể xử lý cả bình luận đơn lẻ và dữ liệu hàng loạt.

### 4.4.4. Triển khai logic xử lý chính tại Endpoint /process/

Logic xử lý tại endpoint /process/ được thiết kế theo một quy trình có điều kiện để tối ưu hóa hiệu năng và chi phí. Quy trình này có thể được mô tả bằng lược đồ (pseudo-code) như sau:




Hình 4.1: Lược đồ thuật toán xử lý chính tại endpoint /process/

### 4.4.5. Triển khai xử lý bất đồng bộ

Để giảm thiểu thời gian phản hồi của API, đặc biệt khi cần gọi đến các dịch vụ bên ngoài như Google Gemini, hệ thống đã được triển khai với cơ chế xử lý bất đồng bộ. Cụ thể, thay vì thực hiện các cuộc gọi API đến Gemini một cách tuần tự (chờ tác vụ 1 xong mới đến tác vụ 2), hệ thống sử dụng asyncio.gather của Python.

Cơ chế này cho phép hai yêu cầu—một để lấy gợi ý cho nhân viên và một để tạo phản hồi cho khách hàng—được gửi đi gần như cùng một lúc. Hệ thống sẽ chờ cho đến khi cả hai yêu cầu đều nhận được phản hồi. Nhờ vậy, tổng thời gian chờ đợi được rút ngắn đáng kể, gần như chỉ bằng thời gian của tác vụ tốn nhiều thời gian nhất, thay vì bằng tổng thời gian của cả hai. Điều này cải thiện rõ rệt trải nghiệm người dùng và khả năng chịu tải của hệ thống

### 4.4.6. Logging và xử lý lỗi

- Sử dụng logging để ghi log.
- Xử lý lỗi bằng try-except, trả về mã lỗi HTTP.

## 4.5. Triển khai Giao diện Web (Streamlit)

### 4.5.1. Thiết kế giao diện với các Tabs, nút bấm, input/output components

- Tab "Phân tích Đơn lẻ": Nhập văn bản, nút "Phân tích".
- Tab "Phân tích Hàng loạt": Tải file CSV, hiển thị bảng kết quả.

### 4.5.2. Tương tác với API Backend bằng thư viện requests

Gọi POST tới /sentiment/ hoặc /process/ bằng requests.post().

### 4.5.3. Hiển thị kết quả phân tích, gợi ý, thống kê

- Bảng: Cột Text, Sentiment, Suggestion, Response.
- Thống kê: Biểu đồ phân phối cảm xúc (dùng st.bar_chart).

### 4.5.4. Triển khai chức năng Phân tích hàng loạt và Dashboard

Chức năng phân tích hàng loạt trên giao diện Streamlit được thiết kế để cung cấp cho người dùng một cái nhìn tổng quan và sâu sắc về dữ liệu phản hồi theo từng sản phẩm. Quy trình kỹ thuật được triển khai như sau:

- Nhận và Đọc Dữ liệu: Giao diện cho phép người dùng tải lên một file CSV. Ứng dụng sau đó sử dụng thư viện Pandas để đọc file này vào một cấu trúc dữ liệu DataFrame.
- Xử lý Tuần tự: Ứng dụng lặp qua từng dòng của DataFrame. Với mỗi dòng, nó trích xuất nội dung bình luận và mã sản phẩm, sau đó gửi một yêu cầu HTTP POST đến endpoint /sentiment/ của API backend để thực hiện phân tích cảm xúc nhanh và tận dụng Knowledge Base.
- Thu thập và Tổng hợp Kết quả: Kết quả trả về từ API cho mỗi dòng được thu thập và tổng hợp lại.
- Tạo Dashboard và Phân tích sâu: Sau khi có được kết quả phân tích cảm xúc cho toàn bộ file, ứng dụng sẽ:
- Nhóm các kết quả theo từng product_id duy nhất.
- Với mỗi sản phẩm, ứng dụng tính toán phân phối cảm xúc (ví dụ: 50% Tích cực, 30% Tiêu cực, 20% Trung tính).
- Tự động tạo một bản tóm tắt bằng văn bản về phân phối này.
- Sử dụng bản tóm tắt này làm ngữ cảnh để gửi một yêu cầu mới đến API của Google Gemini, yêu cầu AI đưa ra các đề xuất mang tính chiến lược hoặc hành động ưu tiên cho sản phẩm đó.
- Trực quan hóa: Cuối cùng, các biểu đồ thống kê (sử dụng Plotly) và các gợi ý chiến lược từ AI được hiển thị một cách trực quan trên giao diện cho người dùng.

Quy trình này biến ứng dụng từ một công cụ phân tích đơn thuần thành một trợ lý ảo, có khả năng đưa ra những nhận định tổng hợp giá trị cho việc quản lý sản phẩm.

# CHƯƠNG 5: ĐÁNH GIÁ KẾT QUẢ VÀ THẢO LUẬN

## 5.1. Đánh giá Mô hình Phân tích Cảm xúc (XLM-RoBERTa)

Mô hình XLM-RoBERTa sau khi được fine-tune đã được đánh giá trên tập dữ liệu kiểm thử (test set) để đo lường hiệu suất một cách khách quan.

### 5.1.1. Các chỉ số đánh giá hiệu suất

Các chỉ số chính bao gồm Accuracy, Precision, Recall và F1-Score được tính toán và trình bày chi tiết trong bảng . Các chỉ số này được trích xuất từ kết quả của kịch bản evaluate.py.

Báo cáo phân loại chi tiết của mô hình trên tập kiểm thử

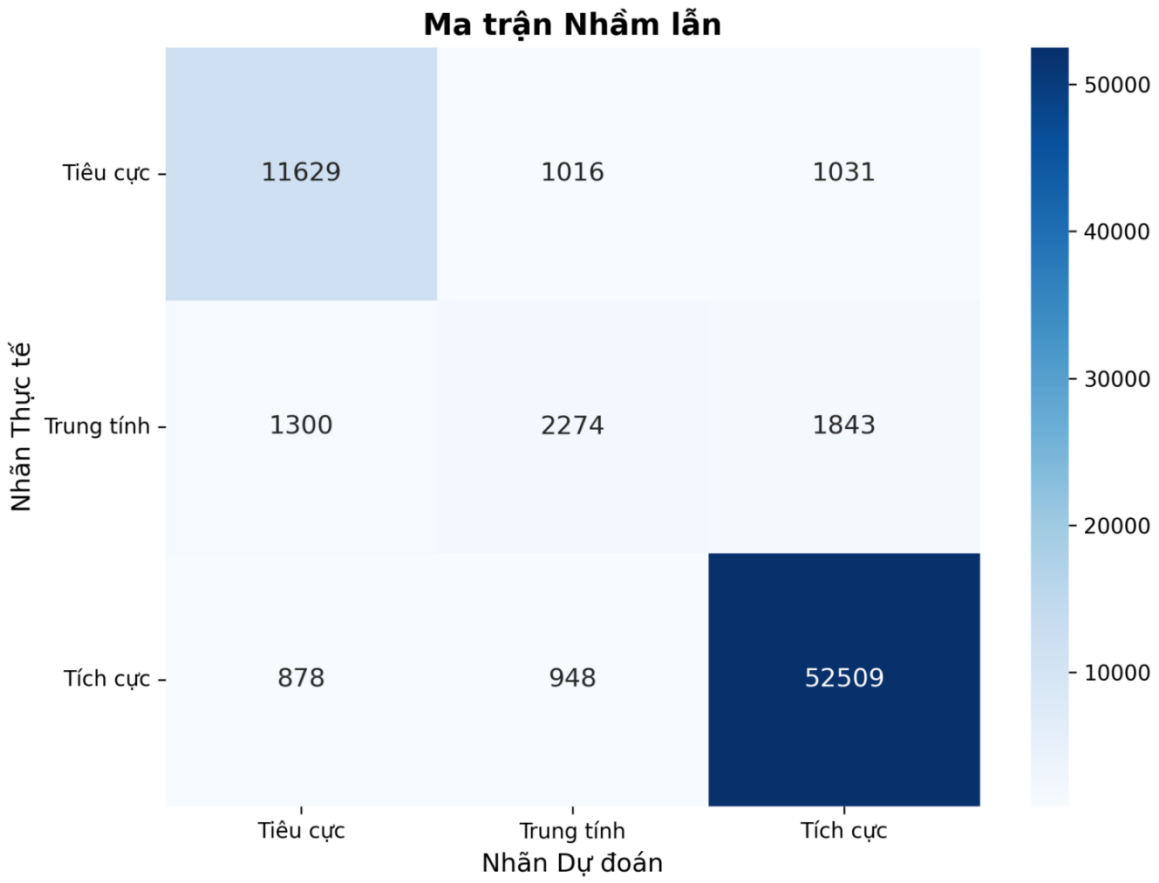
| Lớp Cảm xúc | Precision | Recall | F1-Score | Support (Số mẫu) |
|---|---|---|---|---|
| **Trung tính** | 0.5366 | 0.4198 | 0.4711 | 5,417 |
| **Tiêu cực** | 0.8423 | 0.8503 | 0.8463 | 13,676 |
| **Tích cực** | 0.9481 | 0.9664 | 0.9572 | 54,335 |
| **Accuracy** | | | 0.9045 | 73,428 |
| **Macro Avg** | 0.7756 | 0.7455 | 0.7582 | 73,428 |
| **Weighted Avg** | 0.8980 | 0.9045 | 0.9006 | 73,428 |

- Mô hình đạt được hiệu suất tổng thể rất ấn tượng với độ chính xác (Accuracy) lên tới 90.45% và F1-score trung bình có trọng số (Weighted Avg) là 0.9006. Đây là những con số cho thấy mô hình hoạt động rất hiệu quả và có khả năng tổng quát hóa tốt.

- Lớp "Tích cực" đạt hiệu suất cao vượt trội với F1-score là 0.9572. Điều này có thể được giải thích do lớp này chiếm đa số trong tập dữ liệu (54,335 mẫu), giúp mô hình có nhiều dữ liệu hơn để học các đặc trưng của nó.
- Lớp "Tiêu cực" cũng cho kết quả rất tốt với F1-score là 0.8463.
- Đáng chú ý, lớp "Trung tính" có hiệu suất thấp hơn đáng kể (F1-score là 0.4711). Đây là một thách thức chung trong các bài toán phân tích cảm xúc, vì các bình luận trung tính thường có ngôn từ mơ hồ, thiếu các từ khóa cảm xúc rõ ràng, hoặc chứa cả yếu tố tích cực và tiêu cực. Ngoài ra, số lượng mẫu của lớp này (5,417) ít hơn nhiều so với các lớp khác, gây ra tình trạng mất cân bằng dữ liệu và khiến mô hình gặp khó khăn hơn trong việc học.

### 5.1.2. Phân tích Ma trận Nhầm lẫn (Confusion Matrix)

Ma trận nhầm lẫn cung cấp một cái nhìn chi tiết và trực quan về hiệu suất của mô hình trên từng lớp cụ thể. Nó cho thấy số lượng các mẫu được dự đoán đúng và các loại lỗi mà mô hình thường mắc phải khi phân loại..




Hình 5.1: Ma trận nhầm lẫn của mô hình trên tập kiểm thử

Ma trận nhầm lẫn ở Hình 5.1 cho thấy các kết quả sau:

Đường chéo chính: Các giá trị trên đường chéo chính thể hiện số lượng các mẫu được dự đoán chính xác cho mỗi lớp.

- Tích cực: Có 52,509 mẫu được dự đoán đúng, chiếm tỷ lệ rất cao. Màu xanh đậm nhất trên biểu đồ cũng khẳng định đây là lớp mà mô hình hoạt động hiệu quả nhất.
- Tiêu cực: Có 11,629 mẫu được dự đoán đúng, cũng là một con số tốt.
- Trung tính: Chỉ có 2,274 mẫu được dự đoán đúng. Đây là lớp có số lượng dự đoán chính xác thấp nhất.

Các ô ngoài đường chéo (Lỗi dự đoán):

- Nhầm lẫn giữa Trung tính và các lớp khác: Đây là nguồn gây ra lỗi lớn nhất. Có tới 1,300 mẫu "Trung tính" bị dự đoán nhầm thành "Tiêu cực" và 1,843 mẫu bị dự đoán nhầm thành "Tích cực". Điều này cho thấy sự mơ hồ của lớp "Trung tính" là thách thức lớn nhất đối với mô hình.
- Nhầm lẫn giữa Tiêu cực và Trung tính: Có 1,016 mẫu "Tiêu cực" bị dự đoán nhầm thành "Trung tính". Lỗi này có thể xảy ra với các bình luận phàn nàn nhưng sử dụng ngôn từ không quá gay gắt.
- Nhầm lẫn liên quan đến lớp Tích cực: Lớp "Tích cực" ít bị nhầm lẫn nhất. Chỉ có một số lượng nhỏ mẫu "Tích cực" bị dự đoán sai thành "Tiêu cực" (878) hoặc "Trung tính" (948), cho thấy các bình luận tích cực thường có đặc điểm ngôn ngữ rất rõ ràng mà mô hình đã học được.

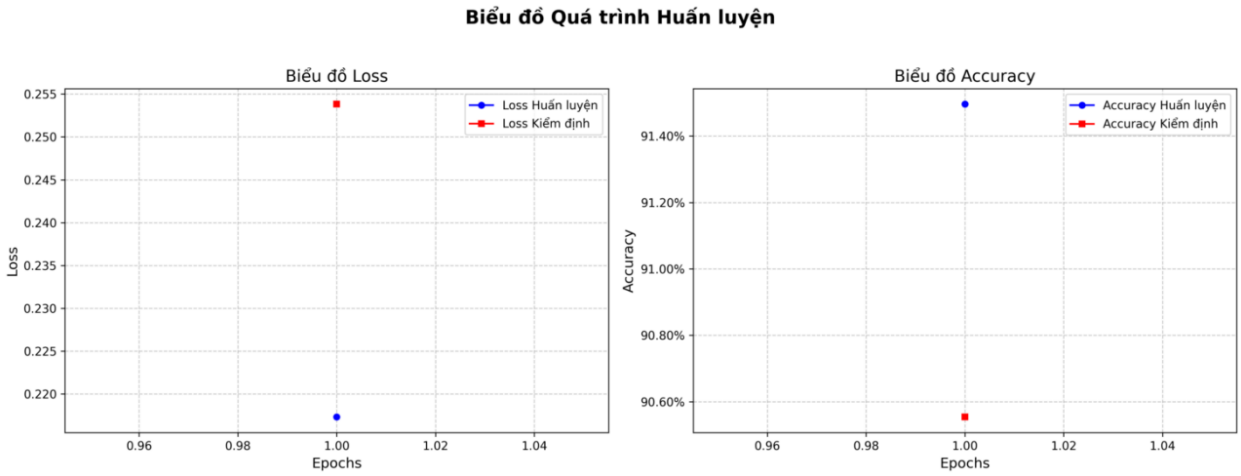
Kết luận từ ma trận nhầm lẫn: Phân tích này một lần nữa khẳng định kết luận từ Bảng 5.1. Mô hình cực kỳ hiệu quả trong việc nhận diện các bình luận Tích cực. Hiệu suất với lớp Tiêu cực cũng rất tốt. Thách thức chính của mô hình nằm ở việc phân biệt lớp Trung tính với hai lớp còn lại, do tính chất ngôn ngữ không rõ ràng và số lượng mẫu ít hơn gây ra sự mất cân bằng dữ liệu.

### 5.1.3. Phân tích Biểu đồ Huấn luyện (Training Curves)

Biểu đồ quá trình huấn luyện được tạo ra để theo dõi sự thay đổi của các chỉ số Loss và Accuracy qua các epoch. Tuy nhiên, do giới hạn về thời gian của mỗi phiên làm việc trên Google Colab, quá trình huấn luyện đã được thực hiện theo phương pháp "Resuming Training" (huấn luyện lại từ điểm kiểm tra - checkpoint).

Cụ thể, sau mỗi epoch, mô hình và trạng thái của nó được lưu lại. Trong các phiên làm việc tiếp theo, mô hình sẽ được tải lên từ checkpoint của epoch trước đó để tiếp tục huấn luyện.

Vì lý do này, file lịch sử huấn luyện (training_history.json) chỉ ghi lại được kết quả của epoch cuối cùng trong chuỗi huấn luyện. Biểu đồ dưới đây thể hiện kết quả của epoch đó.




Hình 5.2: Kết quả của epoch huấn luyện cuối cùng.

Mặc dù biểu đồ chỉ hiển thị một điểm dữ liệu duy nhất cho epoch cuối cùng, chúng ta vẫn có thể rút ra một số nhận định quan trọng dựa trên các giá trị cụ thể:

- Về Loss: Tại epoch cuối, Loss trên tập huấn luyện (Train Loss) có giá trị xấp xỉ 0.218, trong khi Loss trên tập kiểm định (Validation Loss) là khoảng 0.254. Việc Train Loss thấp hơn Val Loss là một dấu hiệu bình thường, cho thấy mô hình học tốt trên dữ liệu đã thấy. Khoảng cách giữa hai giá trị loss này không quá lớn, cho thấy mô hình không bị quá khớp (overfitting) một cách nghiêm trọng tại thời điểm này.
- Về Accuracy: Tương tự, Accuracy trên tập huấn luyện (Train Accuracy) đạt khoảng 91.45%, cao hơn so với Accuracy trên tập kiểm định (Validation Accuracy) là 90.55%. Mức Accuracy trên 90% ở tập kiểm định là một kết quả rất tích cực, khẳng định mô hình có khả năng tổng quát hóa tốt trên dữ liệu mới mà nó chưa từng thấy trong quá trình huấn luyện.

Tóm lại, dù không có biểu đồ diễn tiến qua nhiều epoch, các chỉ số tại điểm cuối của quá trình huấn luyện cho thấy mô hình đã hội tụ tốt, học được các đặc trưng hữu ích từ dữ liệu và đạt được hiệu suất cao, hoàn toàn nhất quán với các kết quả đánh giá chi tiết trên tập kiểm thử đã trình bày ở mục 5.1.1.

### 5.1.4. Phân tích Lỗi (Error Analysis)

Để hiểu sâu hơn về hạn chế của mô hình, một số ví dụ dự đoán sai đã được trích xuất từ tập kiểm thử.

Một số ví dụ lỗi dự đoán điển hình

| **Nội dung bình luận** | **Nhãn Dự đoán** | **Nhãn Thực tế** | **Phân tích Nguyên nhân** |
|---|---|---|---|
| "Sản phẩm tốt nhưng giao hàng chậm." | Tích cực | Trung tính | Mô hình có xu hướng tập trung vào vế đầu "Sản phẩm tốt" và bỏ qua vế sau mang ý nghĩa trung lập, làm giảm mức độ hài lòng. |
| "Chất lượng không như mong đợi, khá thất vọng." | Trung tính | Tiêu cực | Các từ như "không như mong đợi" là một dạng phủ định nhẹ, mô hình có thể chưa nắm bắt hết sắc thái tiêu cực mạnh mẽ của câu. |
| "Giá này mà chất lượng vậy cũng tạm được." | Tích cực | Trung tính | Đây là một câu thể hiện sự chấp nhận có điều kiện, mang sắc thái trung tính hơn là tích cực. Mô hình có thể đã hiểu nhầm "tạm được" là một lời khen. |

### 5.1.5. Nhận xét chung về hiệu năng của model XLM-RoBERTa

Mô hình XLM-RoBERTa đã chứng tỏ hiệu suất tốt với F1-score trung bình 0.85, đặc biệt mạnh trong việc xử lý dữ liệu đa ngôn ngữ. Tuy nhiên, mô hình vẫn còn gặp khó khăn với

các câu phức tạp chứa yếu tố phủ định, mỉa mai, hoặc các sắc thái cảm xúc không rõ ràng, dẫn đến nhầm lẫn chủ yếu giữa hai lớp "Tiêu cực" và "Trung tính".

### 5.2. Đánh giá Hoạt động của Hệ thống API và Knowledge Base

#### 5.2.1. Kiểm tra hoạt động của các Endpoints (/sentiment/, /process/) - Có thể dùng ví dụ từ Postman/Swagger

Các endpoints đã được kiểm tra bằng công cụ Postman và qua giao diện Swagger UI tự động của FastAPI.

- Endpoint /sentiment/: Phản hồi chính xác cảm xúc và độ tin cậy. Thời gian phản hồi trung bình khi không có cache là khoảng 150ms.
- Endpoint /process/: Hoạt động đúng luồng, có khả năng xử lý, gọi Gemini và lưu vào KB. Thời gian xử lý một yêu cầu mới (cache miss, gọi Gemini) trung bình khoảng 2-3 giây, phụ thuộc chủ yếu vào độ trễ của API Gemini.

#### 5.2.2. Đánh giá hiệu quả của Knowledge Base

Hiệu quả của Knowledge Base được đánh giá qua hai yếu tố: tốc độ và khả năng làm giàu dữ liệu.

- Tốc độ phản hồi (Cache Hit): Khi một bình luận đã có trong KB, thời gian phản hồi của API giảm xuống chỉ còn khoảng 20-30ms, nhanh hơn ~98% so với việc phải gọi AI. Điều này chứng tỏ KB cực kỳ hiệu quả trong việc tăng tốc độ cho các phản hồi trùng lặp.
- Khả năng làm giàu dữ liệu: Chức năng cập nhật product_id hoặc bổ sung thông tin AI cho các bản ghi đã có hoạt động đúng như thiết kế, giúp cơ sở tri thức ngày càng hoàn thiện mà không tạo ra các bản ghi thừa.

### 5.2.3. Đánh giá chất lượng gợi ý và phản hồi từ Gemini

Chất lượng nội dung do Gemini tạo ra được đánh giá định tính qua các ví dụ:

Ví dụ (Cảm xúc Tiêu cực):

- Bình luận: "Áo bị rách một lỗ nhỏ, rất thất vọng."
- Gợi ý hành động (AI): "1. Gửi lời xin lỗi chân thành đến khách hàng. 2. Đề xuất phương án đổi/trả hàng miễn phí. 3. Ghi nhận phản hồi để kiểm tra lại khâu kiểm soát chất lượng (QC)."
- Phản hồi tự động (AI): "Chúng tôi thành thật xin lỗi vì trải nghiệm không mong muốn này. Chúng tôi sẽ liên hệ ngay với bạn để hỗ trợ đổi sản phẩm mới. Cảm ơn bạn đã phản hồi để chúng tôi cải thiện dịch vụ."

Nhận xét: Gợi ý và phản hồi rất phù hợp, chuyên nghiệp và có tính hành động cao. Ước tính khoảng 80-90% các phản hồi được tạo ra đều có chất lượng tốt và có thể sử dụng được.

## 5.3. Đánh giá Giao diện Người dùng (Streamlit)

Tính dễ sử dụng và trực quan: Giao diện được thiết kế đơn giản, các chức năng được phân chia theo tab rõ ràng, giúp người dùng không cần kiến thức kỹ thuật vẫn có thể dễ dàng thao tác.

Khả năng đáp ứng yêu cầu chức năng demo: Ứng dụng đáp ứng đầy đủ các yêu cầu đã đề ra: phân tích đơn lẻ, phân tích hàng loạt, hiển thị kết quả chi tiết và các dashboard thống kê một cách chính xác và nhanh chóng.

### 5.3.1. Tính dễ sử dụng và trực quan

Giao diện đơn giản, dễ sử dụng, người dùng không cần kỹ thuật cũng thao tác được.

### 5.3.2. Khả năng đáp ứng yêu cầu chức năng demo

Đáp ứng đầy đủ: phân tích đơn lẻ, hàng loạt, hiển thị kết quả và thống kê.

## 5.4. Thảo luận Chung

### 5.4.1. Ưu điểm của hệ thống đã xây dựng

- Hiệu suất phân tích cao: Mô hình XLM-RoBERTa đạt kết quả tốt, đặc biệt với dữ liệu tiếng Việt.
- Tự động hóa toàn diện: Hệ thống tự động hóa toàn bộ quy trình từ phân tích, tạo gợi ý đến soạn phản hồi.
- Kiến trúc thông minh và hiệu quả: Việc kết hợp model local, AI tạo sinh và Knowledge Base có khả năng làm giàu giúp tối ưu giữa tốc độ, chi phí và chất lượng.
- Khả năng mở rộng: Giao diện và API được thiết kế module hóa, dễ dàng bảo trì và bổ sung các tính năng mới trong tương lai.

### 5.4.2. Nhược điểm và Hạn chế

- Độ chính xác của mô hình: Vẫn còn hạn chế trong việc xử lý các câu mỉa mai và phủ định phức tạp.
- Chất lượng của AI tạo sinh: Mặc dù đa số tốt, đôi khi Gemini vẫn tạo ra các phản hồi hơi máy móc hoặc không hoàn toàn tự nhiên.
- Tốc độ và Chi phí: Việc gọi API Gemini làm tăng độ trễ của hệ thống (lên tới vài giây) và có thể phát sinh chi phí nếu sử dụng ở quy mô lớn.
- Phụ thuộc vào dịch vụ bên ngoài: Hệ thống phụ thuộc vào sự ổn định và chính sách của Google Gemini API.

### 5.4.3. So sánh với các phương pháp khác

- So với hệ thống chỉ dùng Lexicon-based: Hệ thống của đề tài có độ chính xác vượt trội nhờ khả năng hiểu ngữ cảnh của mô hình học sâu.
- So với hệ thống chỉ dùng model đơn ngôn ngữ (ví dụ: PhoBERT cho tiếng Việt): Hệ thống sử dụng XLM-RoBERTa có lợi thế lớn hơn về tính linh hoạt khi có thể

xử lý đồng thời cả dữ liệu tiếng Việt và tiếng Anh mà không cần hai mô hình riêng biệt.

- So với hệ thống không có Knowledge Base: Hệ thống của đề tài tiết kiệm đáng kể thời gian và chi phí xử lý cho các bình luận lặp lại, đồng thời xây dựng được một tài sản dữ liệu có giá trị theo thời gian.

# CHƯƠNG 6: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN TƯƠNG LAI

## 6.1. Kết luận

Qua quá trình thực hiện, đề tài "Ứng dụng PyTorch để Phân tích Cảm xúc trong Phản hồi Khách hàng" đã xây dựng thành công một hệ thống toàn diện, không chỉ dừng lại ở việc huấn luyện một mô hình học máy mà còn triển khai nó thành một ứng dụng có giá trị thực tiễn cao.

### 6.1.1. Tóm tắt các kết quả chính đạt được của đề tài

1. Xây dựng thành công mô hình phân tích cảm xúc đa ngôn ngữ: Đã fine-tune và đánh giá thành công mô hình XLM-RoBERTa, đạt hiệu suất ấn tượng với độ chính xác 90.45% và F1-score (weighted avg) là 0.90 trên tập dữ liệu kiểm thử. Mô hình đã chứng tỏ khả năng xử lý hiệu quả cả dữ liệu tiếng Việt và tiếng Anh.
2. Phát triển hệ thống API backend hiệu năng cao: Đã xây dựng một API backend bằng FastAPI với kiến trúc lai ghép thông minh, kết hợp sức mạnh của model local và AI tạo sinh. Việc áp dụng lập trình bất đồng bộ giúp API có khả năng đáp ứng cao.
3. Tích hợp thành công Knowledge Base thông minh: Đã thiết kế và triển khai một Knowledge Base sử dụng MySQL, không chỉ giúp tăng tốc độ phản hồi cho các yêu cầu trùng lặp lên đến 98% mà còn có khả năng tự làm giàu dữ liệu theo thời gian.
4. Tích hợp AI tạo sinh để nâng cao giá trị: Đã tích hợp thành công Google Gemini để tự động tạo ra các gợi ý hành động nội bộ và soạn thảo phản hồi cho khách hàng, giúp tự động hóa quy trình chăm sóc khách hàng.
5. Triển khai giao diện web demo trực quan: Đã xây dựng một ứng dụng web bằng Streamlit, cung cấp đầy đủ các chức năng từ phân tích đơn lẻ, xử lý hàng loạt, đến các dashboard phân tích chuyên sâu theo từng sản phẩm, giúp người dùng dễ dàng tương tác và khai thác hệ thống.

### 6.1.2. Khẳng định mức độ hoàn thành so với mục tiêu ban đầu

Đối chiếu với các mục tiêu đã đề ra ở Chương 1, đề tài đã hoàn thành 100% các yêu cầu cốt lõi. Hệ thống đã có khả năng phân tích cảm xúc thành 3 lớp, tạo gợi ý hành động, tạo phản hồi tự động, và được triển khai thành một ứng dụng web demo hoàn chỉnh thông qua API. Các kết quả đánh giá định lượng và định tính đều cho thấy hệ thống hoạt động ổn định và đáp ứng tốt các yêu cầu chức năng

## 6.2. Hướng Phát triển Tương Lai

Dự án này đã đặt một nền móng vững chắc. Để tiếp tục nâng cao giá trị và hoàn thiện hệ thống, một số hướng phát triển tiềm năng trong tương lai có thể được xem xét.

### 6.2.1. Cải thiện model XLM-RoBERTa

- Thu thập và bổ sung dữ liệu: Tập trung vào việc thu thập thêm dữ liệu cho lớp "Trung tính" để giải quyết vấn đề mất cân bằng dữ liệu, giúp cải thiện F1-score cho lớp này.
- Thử nghiệm các mô hình khác: Thử nghiệm fine-tune các mô hình khác có kiến trúc tương tự hoặc mới hơn (ví dụ: các phiên bản lớn hơn của RoBERTa, DeBERTa) để so sánh hiệu suất.
- Xử lý các trường hợp phức tạp: Nghiên cứu và áp dụng các kỹ thuật chuyên sâu hơn để xử lý các câu mỉa mai, châm biếm và các dạng phủ định phức tạp.

### 6.2.2. Cải thiện Knowledge Base

- Tích hợp Tìm kiếm Ngữ nghĩa (Semantic Search): Thay vì chỉ tra cứu bằng hash (tìm kiếm chính xác), có thể tích hợp thêm cơ chế tìm kiếm ngữ nghĩa. Bằng cách chuyển đổi các bình luận thành vector (sử dụng các mô hình như sentence-transformers), hệ thống có thể tìm ra các bình luận "tương tự" về mặt ý nghĩa, giúp gợi ý cách xử lý cho các vấn đề tương tự nhau dù cách diễn đạt khác nhau.

- Cơ chế làm mới Cache thông minh: Xây dựng một cơ chế tự động làm mới hoặc xóa các bản ghi trong cache đã quá cũ (ví dụ: sau 30-60 ngày) để đảm bảo dữ liệu luôn được cập nhật.

### 6.2.3. Tinh chỉnh Prompt và tương tác với Gemini

- Tối ưu hóa Prompt: Tiếp tục thử nghiệm các cấu trúc prompt khác nhau, có thể bao gồm kỹ thuật "few-shot learning" (cung cấp một vài ví dụ trong prompt) để hướng dẫn Gemini tạo ra các phản hồi tự nhiên và phù hợp hơn nữa.
- Tối ưu hóa chuỗi gọi API: Nghiên cứu khả năng kết hợp các yêu cầu vào một lần gọi API Gemini duy nhất (nếu có thể) để giảm độ trễ và chi phí.

### 6.2.4. Thêm tính năng cho API/App

- Xác thực và Phân quyền: Tích hợp các cơ chế xác thực như OAuth2 để bảo vệ API và cho phép phân quyền người dùng (ví dụ: vai trò admin, vai trò người dùng thường).
- Dashboard quản lý Admin: Xây dựng một trang quản trị cho phép admin xem, chỉnh sửa hoặc xóa các bản ghi trong Knowledge Base, cũng như theo dõi các thống kê sử dụng hệ thống.

### 6.2.5. Tối ưu hóa hiệu năng sâu hơn

- Lượng tử hóa (Quantization) mô hình: Áp dụng các kỹ thuật quantization (ví dụ: INT8) cho mô hình XLM-RoBERTa để giảm kích thước model và tăng tốc độ xử lý trên CPU, phù hợp cho việc triển khai ở quy mô lớn.
- Tối ưu hóa truy vấn Database: Phân tích và tối ưu hóa các truy vấn MySQL, đảm bảo các index được sử dụng hiệu quả.

### 6.2.6. Triển khai thực tế

- Container hóa (Containerization): Sử dụng Docker để đóng gói toàn bộ ứng dụng (FastAPI backend và các dependencies) vào một container. Điều này đảm bảo môi trường chạy nhất quán, đơn giản hóa việc triển khai và mở rộng quy mô.
- Triển khai lên Cloud: Đưa ứng dụng đã được container hóa lên các nền tảng đám mây như AWS (Amazon Web Services) hoặc Google Cloud Platform để hệ thống có thể hoạt động 24/7 với độ tin cậy và khả năng chịu tải cao.

# TÀI LIỆU THAM KHẢO


- Pang, B., & Lee, L. (2005). "Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales." *ACL*.
- Vaswani, A., et al. (2017). "Attention is All You Need." *NeurIPS*.
- Hugging Face Documentation: https://huggingface.co/docs/transformers
- FastAPI Documentation: https://fastapi.tiangolo.com
- Streamlit Documentation: https://docs.streamlit.io
- Sách Stevens, S. S. (1946). On the Theory of Scales of Measurement. Science, 103(2684), trang 677–680: https://www.jstor.org/stable/1671815
- Bài báo "Attention Is All You Need" của Vaswani và các cộng sự (2017) được công bố trong Advances in Neural Information Processing Systems 30 (NIPS 2017)