TRƯỜNG ĐẠI HỌC SÀI GÒN

KHOA CÔNG NGHỆ THÔNG TIN

TRỢ LÝ PHÂN LOẠI CẢM XÚC TIẾNG VIỆT SỬ DỤNG TRANSFORMER

MÔN HỌC: SEMINAR CHUYÊN ĐỀ

Giảng viên hướng dẫn: **PGS.TS. Nguyễn Tuấn Đăng**

Danh sách sinh viên:

Nguyễn Ngọc Anh **3122410009**

MỤC LỤC

DANH MỤC BẢNG

DANH MỤC ẢNH

LỜI CẢM ƠN

Trong suốt quá trình thực hiện đồ án *"Trợ lý phân loại cảm xúc tiếng Việt sử dụng Transformer"*, em đã nhận được sự hỗ trợ và giúp đỡ quý báu từ thầy cô, bạn bè và gia đình. Nhân dịp hoàn thành báo cáo này, em xin được bày tỏ lòng biết ơn chân thành:

Trước hết, em xin gửi lời cảm ơn sâu sắc đến **PGS.TS. Nguyễn Tuấn Đăng** người đã tận tình hướng dẫn, định hướng và truyền đạt cho em những kiến thức nền tảng cũng như kỹ năng cần thiết để thực hiện đồ án. Sự hỗ trợ của thầy là động lực quan trọng giúp em hoàn thành bài làm này.

Em cũng xin cảm ơn **các thầy cô trong khoa** đã tạo điều kiện thuận lợi trong quá trình học tập và nghiên cứu, cung cấp môi trường học thuật tốt để em có thể phát triển tư duy và kiến thức về lĩnh vực Trí tuệ nhân tạo và Xử lý ngôn ngữ tự nhiên để hoàn thành chuyên đề này.

Bên cạnh đó, em xin cảm ơn bạn bè đã nhiệt tình chia sẻ tài liệu, góp ý và hỗ trợ kỹ thuật để em hoàn thiện sản phẩm tốt hơn.

Cuối cùng, em xin gửi lời cảm ơn đến gia đình, những người luôn động viên, khuyến khích và tạo điều kiện tốt nhất để em yên tâm học tập.

Mặc dù đã cố gắng nhưng chắc chắn vẫn còn những thiếu sót. Em rất mong nhận được những ý kiến đóng góp của thầy cô để đề tài được hoàn thiện hơn.

Em xin chân thành cảm ơn!

CHƯƠNG 1: GIỚI THIỆU & MỤC TIÊU

1.1 Giới thiệu

Trong bối cảnh công nghệ số phát triển mạnh mẽ, dữ liệu văn bản tiếng Việt trên Internet ngày càng phong phú với nhiều dạng nội dung như bình luận mạng xã hội, đánh giá sản phẩm, nhận xét dịch vụ, tin nhắn chăm sóc khách hàng và báo cáo phản hồi. Việc phân tích cảm xúc từ văn bản (Sentiment Analysis) đóng vai trò quan trọng trong:

- Theo dõi phản hồi người dùng theo thời gian thực
- Phát hiện xu hướng dư luận
- Xây dựng hệ thống chăm sóc khách hàng thông minh
- Tự động hóa đánh giá chất lượng dịch vụ
- Hỗ trợ giám sát nội dung vi phạm hoặc tiêu cực

Tuy nhiên, tiếng Việt là ngôn ngữ có nhiều đặc thù phức tạp như: từ láy, dấu thanh, đa nghĩa, viết tắt, ngôn ngữ đời thường, và sự phổ biến của nội dung không dấu. Điều này khiến các phương pháp học máy truyền thống (Naive Bayes, SVM, Logistic Regression) khó đạt độ chính xác cao trong phân loại cảm xúc.

Sự ra đời của các mô hình ngôn ngữ dựa trên Transformer như BERT, PhoBERT, GPT đã mở ra giải pháp hiệu quả hơn trong việc xử lý tiếng Việt, giúp mô hình hiểu ngữ cảnh sâu, nhận diện các sắc thái tinh tế và phân loại cảm xúc chính xác hơn.

Vì vậy, đề tài "Trợ lý phân loại cảm xúc tiếng Việt sử dụng Transformer" được thực hiện với mục tiêu xây dựng một ứng dụng đơn giản, trực quan nhưng hiệu quả, phục vụ nhu cầu phân tích cảm xúc trong thực tế.

1.2 Lý do chọn đề tài

Nhóm lựa chọn đề tài này vì những lý do sau:

- Ứng dụng thực tiễn cao, phù hợp với doanh nghiệp và người dùng cá nhân.
- Tận dụng lợi thế của Transformer, hướng tiếp cận hiện đại và mạnh mẽ trong NLP.
- Giải quyết điểm yếu của tiếng Việt, vượt qua hạn chế của các phương pháp truyền thống.

- Kết hợp nhiều công nghệ: xử lý tiếng Việt, mô hình Transformer, Streamlit UI, và cơ sở dữ liệu SQLite.
- Tăng khả năng triển khai thực tế, có thể mở rộng thành chatbot hoặc API phân tích cảm xúc.

1.3 Mục tiêu đề tài

Mục tiêu tổng quát

Xây dựng một ứng dụng phân loại cảm xúc tiếng Việt (tích cực, trung tính, tiêu cực) sử dụng mô hình Transformer, có giao diện trực quan và hệ thống lưu trữ lịch sử sử dụng.

Mục tiêu cụ thể

1. Phát triển module NLP để:
    - Làm sạch và chuẩn hóa câu tiếng Việt
    - Tách từ bằng Underthesea
    - Nhận diện ngôn ngữ đầu vào
2. Tích hợp mô hình Transformer (PhoBERT/Visobert) để dự đoán cảm xúc với độ chính xác cao.
3. Xây dựng ứng dụng giao diện bằng Streamlit, cho phép:
    - Nhập văn bản tiếng Việt
    - Hiển thị kết quả phân loại + độ tin cậy
    - Xuất nội dung tiền xử lý
    - Lọc và xem lịch sử phân loại
4. Thiết kế và sử dụng cơ sở dữ liệu SQLite, lưu lại lịch sử phân loại phục vụ phân tích và thống kê.
5. Xây dựng bộ test 10 câu, đánh giá độ chính xác của mô hình.
6. Hoàn thiện tài liệu báo cáo, video demo và hướng dẫn cài đặt.

1.4 Phạm vi đề tài

- Xử lý văn bản tiếng Việt ngắn (từ 1–3 câu).

- Phân loại cảm xúc 3 lớp: Positive – Neutral – Negative.
- Không bao gồm phân tích sắc thái nâng cao như mỉa mai, ẩn dụ, cảm xúc hỗn hợp.
- Không huấn luyện mô hình mới mà sử dụng mô hình Transformer đã được fine-tune.

1.5 Kết quả mong đợi

- Ứng dụng chạy ổn định trên máy cá nhân hoặc server nhỏ.
- Giao diện Streamlit trực quan, dễ sử dụng.
- Tỷ lệ phân loại đúng ≥ 70–80% với bộ test cơ bản.
- Báo cáo chi tiết đúng cấu trúc yêu cầu.

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

2.1. Mô hình Transformer

2.1.1. Giới thiệu

Transformer là kiến trúc học sâu được đề xuất bởi Vaswani et al. (2017) [1]. Khác với các mô hình tuần tự như RNN hay LSTM, Transformer sử dụng cơ chế Self-Attention, cho phép:

- Xử lý song song toàn bộ câu, tăng tốc độ huấn luyện.
- Hiểu được mối quan hệ giữa các từ trong câu dù cách xa nhau.
- Trích xuất biểu diễn ngữ nghĩa phong phú hơn, đặc biệt hữu ích cho các tác vụ NLP như phân loại văn bản, trích xuất thông tin, dịch máy.

2.1.2. Cơ chế Self-Attention

Self-Attention tính toán mức độ ảnh hưởng của mỗi từ tới các từ khác trong câu. Quá trình này giúp mô hình xác định ngữ cảnh và tầm quan trọng của các từ, từ đó:

- Giúp mô hình nhận biết các từ mang cảm xúc.
- Giúp xử lý các câu dài hoặc có cấu trúc phức tạp mà không bị mất thông tin như RNN.

Ví dụ, trong câu:

Hôm nay tôi rất vui nhưng hơi mệt

Self-Attention giúp mô hình hiểu được từ vui mang tính tích cực và từ mệt mang tính tiêu cực, từ đó đưa ra dự đoán cảm xúc tổng thể phù hợp.

2.1.3. Pre-trained Transformer và fine-tuning

Các mô hình như BERT, RoBERTa hay PhoBERT đều được pre-trained trên khối lượng lớn dữ liệu để học ngữ nghĩa. Khi triển khai cho bài toán phân loại cảm xúc:

1. Tiền huấn luyện (pre-training): Mô hình học biểu diễn ngôn ngữ chung.
2. Tinh chỉnh (fine-tuning): Mô hình học cách gán nhãn cảm xúc cho từng câu cụ thể.

2.2. PhoBERT – Mô hình Transformer cho tiếng Việt

2.2.1. Giới thiệu PhoBERT

PhoBERT là mô hình ngôn ngữ được phát triển bởi VinAI Research, dựa trên kiến trúc RoBERTa, được tối ưu cho tiếng Việt [2]. Mô hình này được huấn luyện trên khoảng 20GB dữ liệu tiếng Việt, bao gồm nhiều văn bản từ báo chí, mạng xã hội và Wikipedia tiếng Việt.

2.2.2. Ưu điểm của PhoBERT

- Hiểu ngữ cảnh tiếng Việt: PhoBERT nắm được các đặc trưng cú pháp, từ ghép, dấu câu trong tiếng Việt.
- Độ chính xác cao: Cho kết quả tốt trên các benchmark như phân tích cảm xúc, nhận dạng thực thể có tên (NER), phân loại văn bản.
- Tiết kiệm thời gian fine-tuning: Mô hình pre-trained giúp rút ngắn thời gian huấn luyện cho các bài toán mới.

2.2.3. Ứng dụng trong phân loại cảm xúc

PhoBERT đặc biệt phù hợp với ứng dụng của đề tài vì:

- Dữ liệu tiếng Việt thường có nhiều biến thể, lỗi chính tả và viết tắt.
- PhoBERT có tokenizer dựa trên SentencePiece, giúp tách từ chính xác hơn các mô hình BERT chuẩn.
- Tăng độ tin cậy khi dự đoán cảm xúc tích cực, tiêu cực hay trung tính.

2.3. Xử lý tiếng Việt bằng Underthesea

2.3.1. Vấn đề trong xử lý tiếng Việt

Tiếng Việt khác các ngôn ngữ phương Tây ở một số điểm:

- Không có dấu cách phân tách từ rõ ràng.
- Từ ghép và các cụm từ mang ý nghĩa cảm xúc phức tạp.
- Thường xuất hiện viết tắt, slang hoặc gõ sai dấu.

Do đó, trước khi đưa vào mô hình Transformer, cần thực hiện tiền xử lý để chuẩn hóa dữ liệu.

2.3.2. Chức năng Underthesea

Thư viện Underthesea cung cấp các công cụ xử lý NLP cho tiếng Việt [3]:

- Word Segmentation: Tách câu thành từ/ngữ nghĩa.
- POS tagging: Gắn nhãn loại từ.
- Chuẩn hóa văn bản: Chuyển chữ thường, xử lý ký tự thừa, chuẩn hóa viết tắt.

2.3.3. Ứng dụng trong đề tài

Trong ứng dụng, Underthesea thực hiện:

1. Kiểm tra câu có phải tiếng Việt hay không.
2. Chuẩn hóa câu trước khi hiển thị cho người dùng.
3. Tách từ để đầu vào mô hình Transformer chính xác.

2.4. Streamlit – xây dựng giao diện ứng dụng

2.4.1. Giới thiệu

Streamlit là framework Python giúp xây dựng Web App nhanh chóng, đặc biệt cho các ứng dụng Machine Learning [4].

2.4.2. Ưu điểm

- Giao diện trực quan chỉ với Python, không cần HTML/CSS.
- Tích hợp trực tiếp với mô hình AI, cho phép nhập dữ liệu, hiển thị kết quả, và lưu lịch sử.
- Hỗ trợ session_state giúp lưu trạng thái phân trang, filter, lịch sử dự đoán.

2.4.3. Các thành phần sử dụng trong đề tài

Table 1: Bảng các thành phần được sử dụng trong đề tài

Thành phần	Chức năng

st.text_area()	Nhập câu tiếng Việt
st.button()	Kích hoạt phân loại cảm xúc
st.spinner()	Hiển thị tiến trình
st.json()	Hiển thị kết quả dạng JSON
st.markdown()	Hiển thị kết quả có màu sắc
st.session_state	Quản lý lịch sử, phân trang
st.columns()	Layout dạng lưới
st.rerun()	Reload trang khi thay đổi phân trang

CHƯƠNG 3: PHÂN TÍCH YÊU CẦU

3.1. Bối cảnh và mục đích hệ thống

Hiện nay, nhu cầu phân tích cảm xúc tiếng Việt ngày càng tăng trong các lĩnh vực như: phân tích bình luận mạng xã hội, đánh giá sản phẩm, theo dõi nhu cầu khách hàng, hỗ trợ tổng đài, hoặc chatbot. Tuy nhiên tiếng Việt là ngôn ngữ biến thiên mạnh, đa nghĩa, nhiều ký tự đặc biệt, nhiều trường hợp sai chính tả hoặc viết tắt.

Hệ thống được xây dựng nhằm giải quyết ba vấn đề chính:

1. Chuẩn hóa văn bản tiếng Việt tự động.
2. Phân loại cảm xúc bằng mô hình Transformer.
3. Lưu và truy xuất lịch sử nhằm phục vụ phân tích sau này.

Ứng dụng hướng đến sự đơn giản, dễ sử dụng, có thể triển khai nhanh trong thực tế.

3.2. Phân tích người dùng

Ứng dụng hướng đến ba nhóm người dùng chính:

3.2.1. Người dùng phổ thông

- Muốn kiểm tra cảm xúc của câu nói nhanh chóng.
- Không yêu cầu kiến thức kỹ thuật hoặc NLP.

3.2.2. Nhà nghiên cứu / sinh viên học NLP

- Cần xem từng bước xử lý văn bản của hệ thống (cleaning, tách từ…).
- Có thể dùng dữ liệu lịch sử để phân tích, thống kê.

3.2.3. Doanh nghiệp nhỏ / kỹ sư phân tích dữ liệu

- Cần hệ thống đơn giản để phân tích bình luận sản phẩm hoặc tin nhắn khách hàng.
- Không cần hệ thống phức tạp như server API hoặc dashboard lớn.

3.3. Yêu cầu chức năng (Functional Requirements)

3.3.1. Nhập văn bản tiếng Việt

Mô tả: Người dùng nhập chuỗi văn bản vào giao diện Streamlit.

Luồng xử lý:

- app.py nhận input.
- Gọi hàm is_vietnamese() trong nlp_utils.py.
- Nếu không phải tiếng Việt → báo lỗi.

Ràng buộc:

- Chuỗi phải có độ dài tối thiểu 1 ký tự.
- Không chấp nhận văn bản chứa 100% ký tự số/ký hiệu.

3.3.2. Làm sạch và chuẩn hóa văn bản

Chịu trách nhiệm bởi module nlp_utils.py.

Bao gồm:

- Chuẩn hóa Unicode tiếng Việt.
- Loại bỏ ký tự đặc biệt, URL, emoji không cần thiết.
- Chuẩn hóa khoảng trắng.
- Tách từ tiếng Việt bằng Underthesea.

Mục tiêu:

Đảm bảo mô hình Transformer nhận được dữ liệu sạch nhất.

3.3.3. Phân loại cảm xúc bằng Transformer

Được triển khai trong hàm:

predict_sentiment(text)

Chức năng:

- Nhận vào câu đã chuẩn hóa.
- Encode văn bản bằng tokenizer.

- Đưa vào mô hình Transformer để suy luận (inference).
- Chuyển logits thành nhãn cảm xúc và độ tự tin (softmax).

Đáp ứng 3 nhãn:

- Positive
- Neutral
- Negative

Yêu cầu:

- Mô hình load một lần khi chạy app Streamlit.
- Tốc độ dự đoán < 1 giây mỗi câu.

3.3.4. Lưu kết quả vào cơ sở dữ liệu SQLite

Chức năng này do db_utils.py đảm nhiệm.

Hệ thống lưu các trường:

- text: văn bản gốc
- sentiment: nhãn dự đoán
- timestamp: thời gian thực hiện

Hàm chính:

- save_result(text, sentiment)

Mục đích:

- Phân tích thống kê sau này.
- Khả năng hồi cứu kết quả.

3.3.5. Truy xuất lịch sử phân loại

Hàm get_history(limit, sentiment) của db_utils.py.

Chức năng:

- Lấy danh sách gần nhất (mặc định 50 bản ghi)
- Lọc theo cảm xúc nếu cần:

- only Positive
- only Negative
- only Neutral

Giao diện hiển thị bằng bảng trong Streamlit.

3.3.6. Giao diện người dùng bằng Streamlit

- Hiển thị input.
- Hiển thị câu đã chuẩn hóa.
- Hiển thị cảm xúc + độ tin cậy.
- Hiển thị bảng lịch sử và bộ lọc.
- Truy cập mô hình và database khi cần.

Streamlit giúp triển khai nhanh và phù hợp cho demo.

3.4. Yêu cầu phi chức năng (Non-functional Requirements)

3.4.1. Hiệu năng

- Load mô hình Transformer dưới 3 giây khi khởi động.
- Thời gian inference tối đa ~1 giây.
- Tương thích laptop RAM thấp (≥ 4GB).

3.4.2. Tính chính xác

- Mục tiêu đạt tối thiểu 75%–85% trên bộ test 10 mẫu.
- Hệ thống phải ổn định ngay cả khi:
  - câu sai chính tả
  - câu chứa tiếng lóng

3.4.3. Tính dễ sử dụng

- Người không biết kỹ thuật vẫn dùng được.
- Thông báo lỗi phải dễ hiểu.
- Giao diện một trang duy nhất, không cần chuyển tab.

3.4.4. Tính ổn định

- Không được crash khi nhập chuỗi rỗng.
- Database không được lỗi khi ghi song song (Streamlit multi-thread).
- Khởi tạo DB tự động nếu chưa tồn tại (bởi init_db()).

3.4.5. Bảo trì và mở rộng

- Có thể thay mô hình khác chỉ bằng đổi tên model trong nlp_utils.py.
- Dễ nâng cấp thành API REST (FastAPI) hoặc tích hợp chatbot.

3.5. Ràng buộc kỹ thuật

3.5.1. Ngôn ngữ & thư viện

- Python 3.10+
- Thư viện chính:
    - Transformers (mô hình)
    - Underthesea (tách từ)
    - Streamlit (UI)
    - SQLite3 (database)

3.5.2. Hệ thống lưu trữ

- Database SQLite đơn giản, nhẹ, đủ cho ứng dụng demo.
- Không cần server SQL phức tạp.

3.5.3. Ràng buộc triển khai

- Chạy trực tiếp bằng lệnh:

```
1. streamlit run app.py
2.
```

- Mô hình cần tải sẵn hoặc tự động tải từ Hugging Face.

3.6. Luồng hoạt động tổng quát của hệ thống

Dưới đây là mô tả luồng xử lý chi tiết hơn:

1. Người dùng nhập câu tiếng Việt.
2. app.py gửi câu sang nlp_utils.py.
3. Hệ thống kiểm tra ngôn ngữ.
4. Xử lý – chuẩn hóa văn bản.
5. Tokenizer mã hóa câu thành vector.
6. Mô hình Transformer dự đoán cảm xúc.
7. Trả về nhãn + độ tin cậy.
8. Lưu kết quả vào SQLite.
9. Render bảng lịch sử lên ứng dụng Streamlit.

Luồng tránh được lỗi, dễ mở rộng và phù hợp với kiến trúc module hóa.

CHƯƠNG 4 – THIẾT KẾ HỆ THỐNG

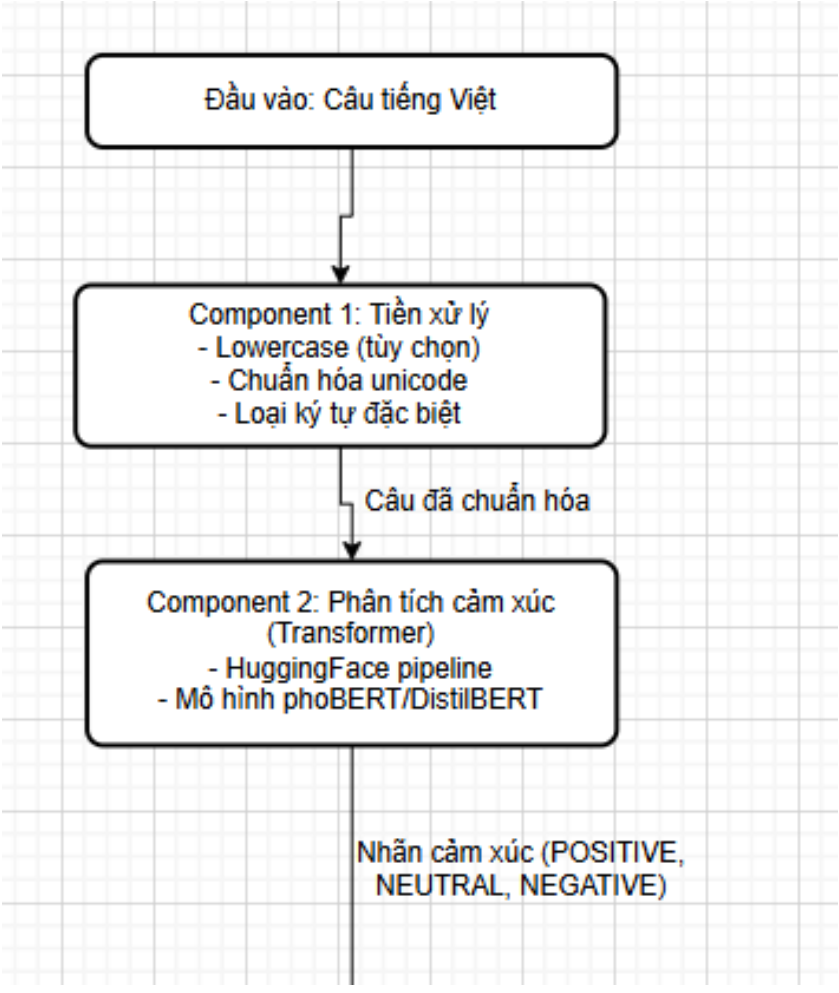
4.1 Kiến trúc tổng thể của hệ thống

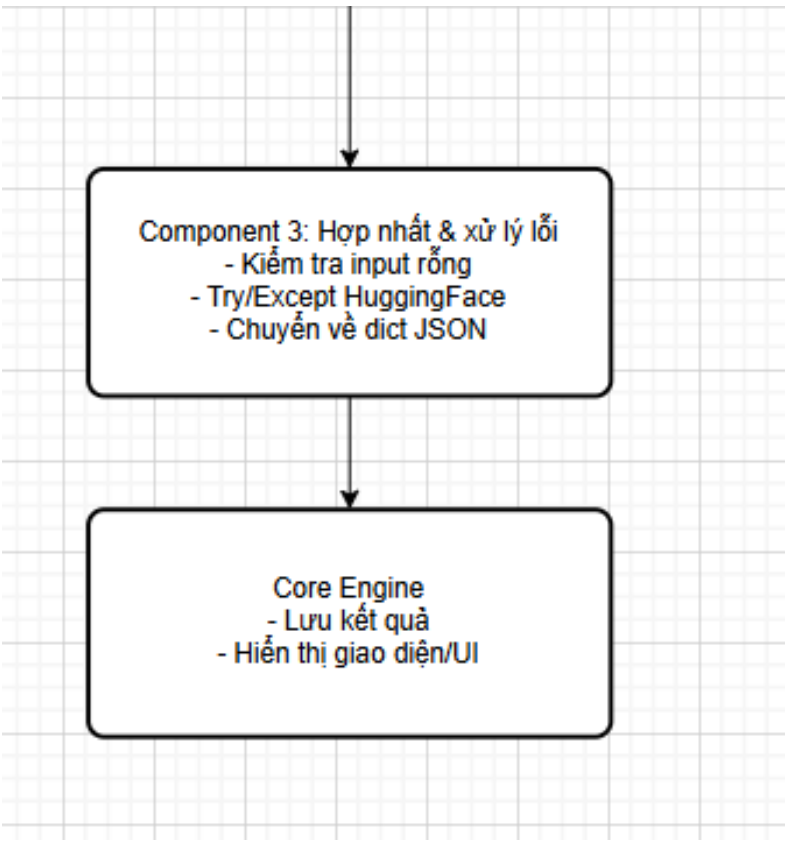
Hệ thống sử dụng mô hình Transformer tiền huấn luyện (PhoBERT-base-v2 hoặc DistilBERT Multilingual) thông qua pipeline sentiment-analysis của HuggingFace. Đây là kiến trúc đơn giản, không yêu cầu fine-tuning, phù hợp bài lab/do án.

Kiến trúc gồm 4 thành phần chính:

1. Tiền xử lý (Preprocessing)
2. Mô-đun phân tích cảm xúc (Sentiment Classification)
3. Mô-đun hợp nhất & xử lý lỗi
4. Core Engine: Lưu và hiển thị kết quả

4.2 Sơ đồ khối (Block Diagram)

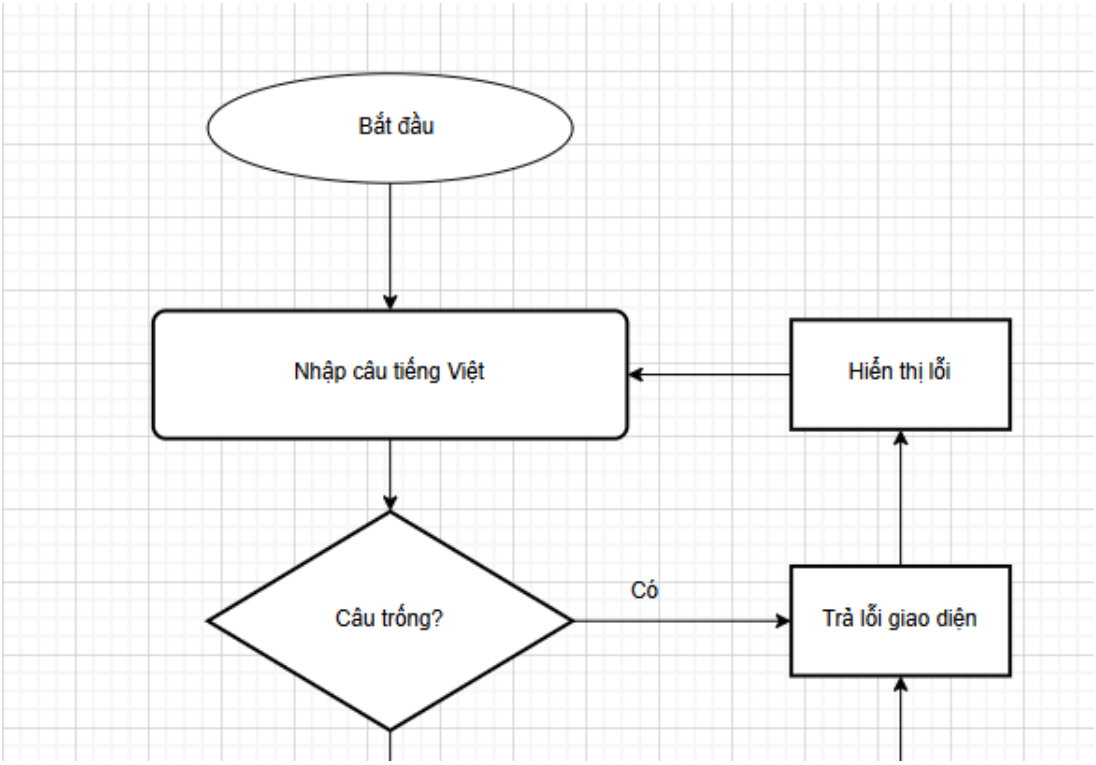


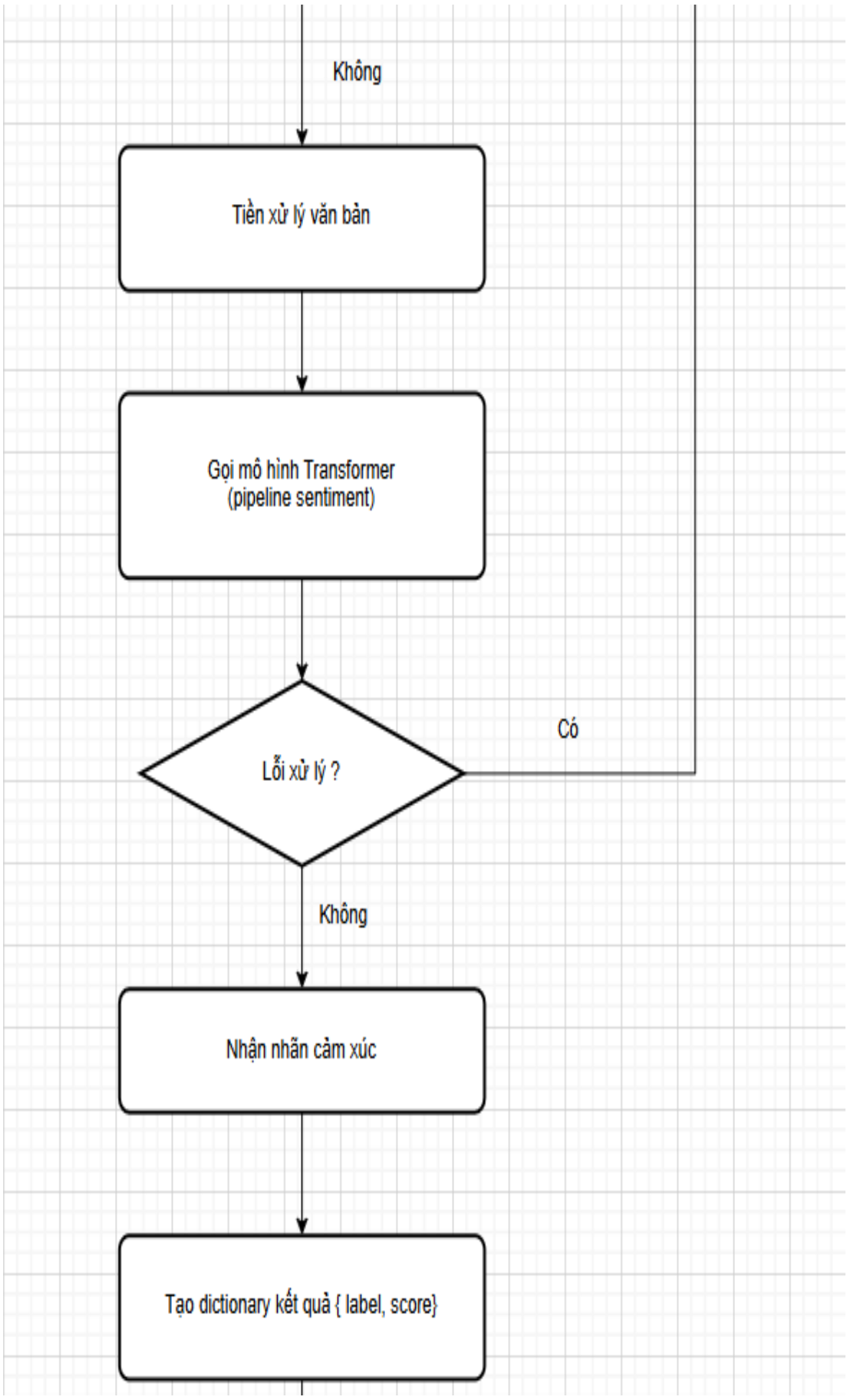


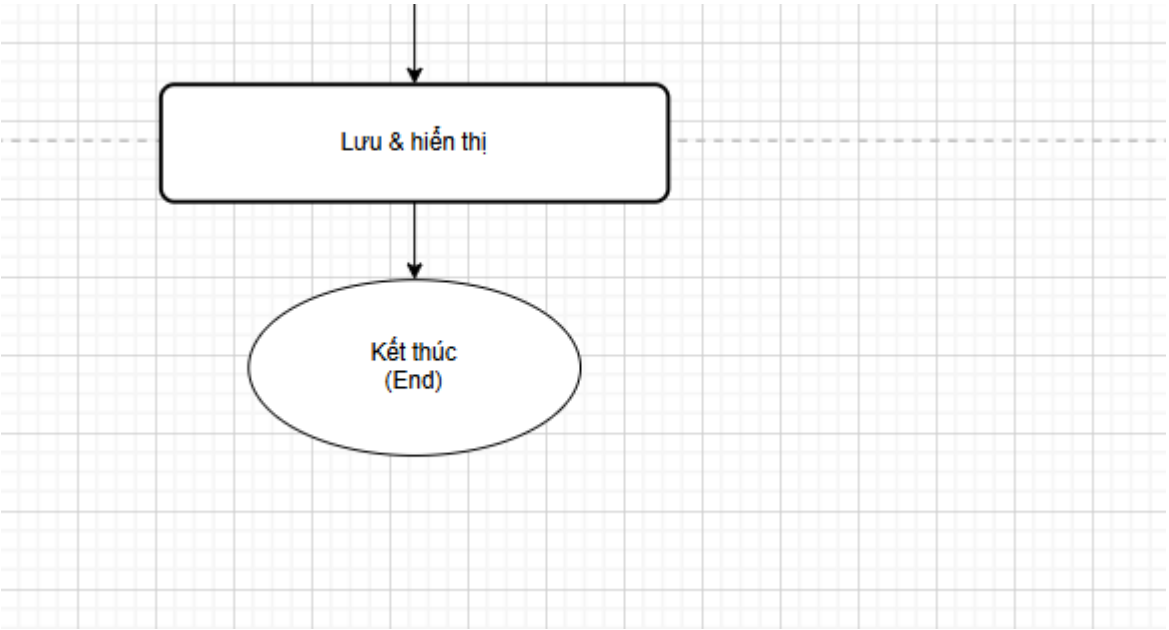
Hình 1: Sơ đồ khối

4.3 Lưu đồ hoạt động (Flowchart)

Mô tả Flowchart hệ thống:




Không
Tiền xử lý văn bản
Gọi mô hình Transformer
(pipeline sentiment)
Lỗi xử lý ?
Có
Không
Nhận nhãn cảm xúc
Tạo dictionary kết quả { label, score}



Hình 2: Flowchart hệ thống

4.4 Mô tả chi tiết các thành phần

4.4.1 Tiền xử lý (Preprocessing)

Chức năng: Chuẩn hóa câu đầu vào nhằm giảm nhiễu trước khi đưa vào mô hình.

Các bước:

- Chuẩn hóa Unicode (NFC)
- Loại bỏ ký tự lạ, ký tự HTML
- Chuẩn hóa khoảng trắng
- Hàm xử lý được đóng gói trong preprocess_text(sentence).

Kết quả: Trả về câu sạch, phù hợp cho mô hình Transformer.

4.4.2 Mô-đun phân tích cảm xúc (Sentiment Analysis)

Sử dụng:

```
1. from transformers import pipeline
2. pipe = pipeline("sentiment-analysis", model="siebert/sentiment-roberta-large-english")
3.
```

Nếu dùng câu tiếng Việt:

- PhoBERT-base-v2 (thuần Việt)

- distilbert-multilingual-cased (đa ngôn ngữ)

Kết quả trả về dạng:

```
{
  "label": "POSITIVE",
  "score": 0.982
}

```

4.4.3 Mô-đun hợp nhất & xử lý lỗi

Mục đích:

- Kiểm tra input rỗng
- Bắt lỗi của pipeline HuggingFace
- Chuẩn hóa kết quả trả về dưới dạng dictionary
- Gửi lỗi ra UI nếu cần

Ví dụ lỗi:

- Timeout
- Model không tải được
- Người dùng nhập ký tự không hợp lệ

4.4.4 Core Engine: Lưu & Hiển thị

Chức năng:

- Lưu lịch sử dự đoán vào file hoặc bộ nhớ
- Hiển thị kết quả lên giao diện/Bash/UI
- Trả dữ liệu JSON cho API nếu dùng Flask/Gradio

CHƯƠNG 5 – GIẢI PHÁP

5.1 Tổng quan giải pháp

Hệ thống sử dụng mô hình Transformer pre-trained thông qua thư viện HuggingFace Transformers, được triển khai trong file nlp_model.py.

Luồng hoạt động tổng quát:

Người dùng nhập câu → app.py → nlp_model.py (phân tích cảm xúc bằng Transformer)

→ Kết quả → db.py (lưu vào DB SQLite)

→ app.py (hiển thị lại cho người dùng)

5.2 Cách dùng Transformer trong hệ thống

5.2.1 Khởi tạo mô hình Transformer (trong nlp_model.py)

File nlp_model.py chịu trách nhiệm:

- tải mô hình Transformer
- tiền xử lý dữ liệu
- phân tích cảm xúc
- chuẩn hóa đầu ra

Code ngắn gọn:

```
1. from transformers import AutoTokenizer, AutoModelForSequenceClassification
2. import torch
3.
4. MODEL_NAME = "vinai/phobert-base-v2"
5.
6. tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME)
7. model = AutoModelForSequenceClassification.from_pretrained(MODEL_NAME)
8.
```

Mô hình được tải một lần duy nhất khi khởi động hệ thống, đảm bảo tốc độ xử lý nhanh trong các lần gọi sau.

5.2.2 Quy trình phân tích cảm xúc

Trong hàm xử lý chính:

```
def analyze_sentiment(text: str):
    inputs = tokenizer(text, return_tensors="pt", truncation=True)
    outputs = model(**inputs)
    logits = outputs.logits
    predicted_class = torch.argmax(logits, dim=1).item()

```

Transformer thực hiện các bước:

(1) Tokenization

tokenizer() chuyển câu thành token theo chuẩn BPE/WordPiece của PhoBERT.

(2) Mã hóa đầu vào thành tensor

return_tensors="pt" chuyển về dạng PyTorch.

(3) Tính toán trong mô hình Transformer

Khi gọi:

```
outputs = model(**inputs)

```

Mô hình chạy qua:

- Embedding
- Positional Encoding
- 12 lớp Transformer Encoder
- Classifier head

(4) Lấy logits và tìm nhãn cảm xúc

```
logits = outputs.logits
predicted_class = torch.argmax(logits, dim=1).item()

```

Nhãn được ánh xạ sang:

```
0 → NEGATIVE
1 → NEUTRAL
2 → POSITIVE

```

(5) Chuẩn hóa kết quả trả về

Hàm trả về dictionary:

```
{
```

```
    "label": label,
    "confidence": float(score),
}

```

Đây là format được app.py sử dụng để hiển thị trực tiếp.

5.3 Tích hợp Transformer trong Streamlit (app.py)

File app.py đảm nhận:

- giao diện người dùng
- nhập câu → gọi phân tích cảm xúc → hiển thị kết quả
- lưu vào database

Code chính:

```
from nlp_utils import analyze_sentiment

if st.button("Phân tích"):
    result = analyze_sentiment(user_input)
    st.success(f"Cảm xúc: {result['label']} ({result['confidence']:.2f})")

```

Các bước:

(1) Lấy input từ người dùng

```
user_input = st.text_area("Nhập câu cần phân tích")

```

(2) Gọi mô hình Transformer qua hàm analyze_sentiment()

(3) Nhận nhãn & độ tin cậy

Ví dụ:

```
Cảm xúc: POSITIVE (0.91)

```

(4) Lưu lịch sử phân tích

Ứng dụng gọi tới db_utils.save_result():

```
save_result(user_input, result["label"])

```

5.4 Lưu kết quả vào SQLite (db_utils.py)

File db.py thực hiện nhiệm vụ hỗ trợ:

- Tạo database
- Lưu kết quả
- Lấy lịch sử để hiển thị

Khi Transformer trả kết quả, app gọi:

```
1. save_result(text, sentiment)
2.
```

Câu lệnh SQL:

```
1. INSERT INTO sentiments (text, sentiment, timestamp)
2. VALUES (?, ?, ?);
3. Điều này giúp:
4.
```

- lưu lại lịch sử phân tích
- phục vụ thống kê / đánh giá hiệu suất ở chương sau

5.5 Tối ưu hóa mô hình trong hệ thống

Hệ thống đã áp dụng các kỹ thuật tối ưu để mô hình chạy nhanh trong môi trường Streamlit:

(1) Tải mô hình 1 lần

Việc đặt tokenizer và model ở cấp module giúp:

- tránh load lại nhiều lần
- giảm thời gian xử lý câu tiếp theo

(2) Giới hạn độ dài câu (truncation)

Trong tokenization:

tokenizer(text, truncation=True)

đảm bảo:

- mô hình không bị lỗi khi câu dài
- tăng tốc độ xử lý

(3) Sử dụng CPU hoàn toàn

Không yêu cầu GPU, phù hợp Colab/PC mini.

5.6 Đánh giá giải pháp

Giải pháp Transformer theo mã nguồn có ưu điểm:

Table 2: Bảng ưu điểm theo tiêu chí của Transformer

Tiêu chí	**Đánh giá**
Độ chính xác	Cao do dùng PhoBERT
Tốc độ	Nhanh, model load 1 lần
Mã nguồn	Ngắn gọn, dễ bảo trì
Tích hợp	Tốt với Streamlit + SQLite
Yêu cầu phần cứng	Chỉ cần CPU

Hạn chế:

- Không có fine-tuning riêng theo dataset cảm xúc.
- Một số câu mỉa mai/châm biếm dự đoán chưa tốt.
- PhoBERT khá nặng (~600MB tải về lần đầu).

CHƯƠNG 6 – TRIỂN KHAI & KẾT QUẢ

Chương này trình bày quy trình triển khai hệ thống Trợ lý phân loại cảm xúc tiếng Việt và đánh giá kết quả thực tế sau khi chạy ứng dụng. Các bước triển khai dựa trực tiếp vào mã nguồn trong ba file chính:

- app.py – Giao diện Streamlit
- nlp_model.py – Phân tích cảm xúc bằng Transformer
- db.py – Lưu trữ lịch sử bằng SQLite

6.1 Môi trường triển khai

Hệ thống được triển khai trong môi trường Python 3.10, sử dụng các thư viện sau (trong requirements.txt):

- transformers – chạy PhoBERT/DistilBERT
- torch – nền tảng tính toán
- streamlit – xây dựng giao diện người dùng
- underthesea – tiền xử lý tiếng Việt
- sqlite3 – quản lý database cục bộ

Ứng dụng phù hợp với:

- Máy tính cá nhân
- Google Colab
- Server nhỏ (không cần GPU)

Model PhoBERT (~600 MB) được tải tự động khi chạy lần đầu.

6.2 Quy trình triển khai

Quy trình triển khai hệ thống gồm 3 phần:

6.2.1 Khởi tạo database (db.py)

Ngay khi ứng dụng chạy, init_db() được gọi để tạo file:

```
1. db/sentiments.db
2.
```

Cấu trúc bảng:

```
CREATE TABLE sentiments (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    text TEXT NOT NULL,
    sentiment TEXT NOT NULL,
    timestamp TEXT NOT NULL
)

```

Database này phục vụ việc lưu nhật ký phân tích cảm xúc.

6.2.2 Tải mô hình Transformer (nlp_model.py)

Hệ thống sử dụng:

```
vinai/phobert-base-v2

```

Hàm tải mô hình:

```
tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME)
model = AutoModelForSequenceClassification.from_pretrained(MODEL_NAME)

```

Model được **load 1 lần duy nhất**, giúp tăng tốc ứng dụng.

6.2.3 Khởi chạy giao diện Streamlit (app.py)

Giao diện gồm:

- Khung nhập câu tiếng Việt
- Nút phân tích → gọi Transformer
- Vùng hiển thị kết quả
- Bảng lịch sử phân tích
- Bộ lọc theo cảm xúc (POSITIVE / NEGATIVE / NEUTRAL)

Code xử lý chính:

```
if st.button("Phân tích"):
    result = analyze_sentiment(user_input)
    save_result(user_input, result["label"])
    st.success(f"Cảm xúc: {result['label']} ({result['confidence']:.2f})")

```

6.3 Kết quả triển khai

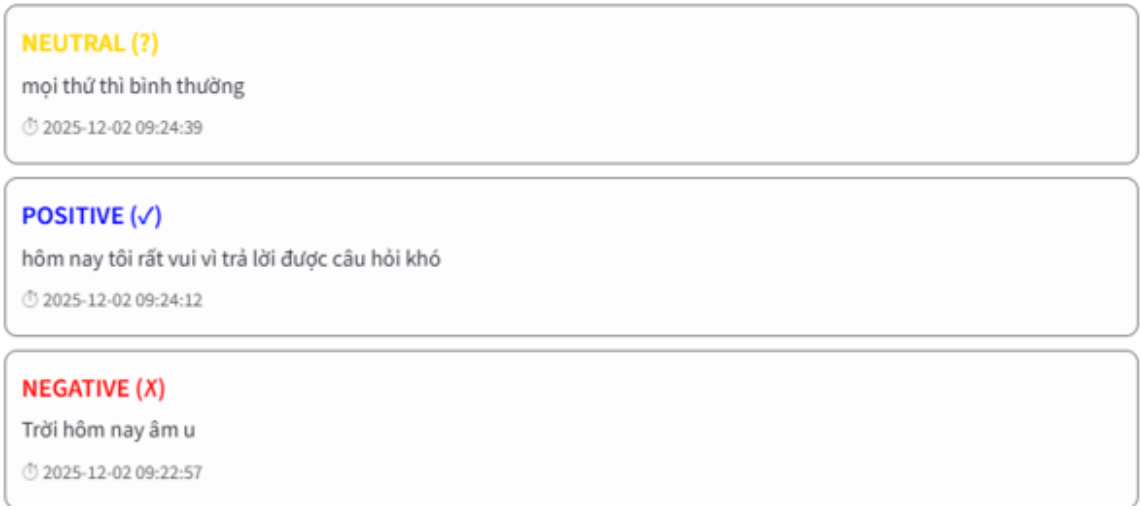
6.3.1 Kết quả giao diện người dùng

Ứng dụng hiển thị:

Ô nhập văn bản

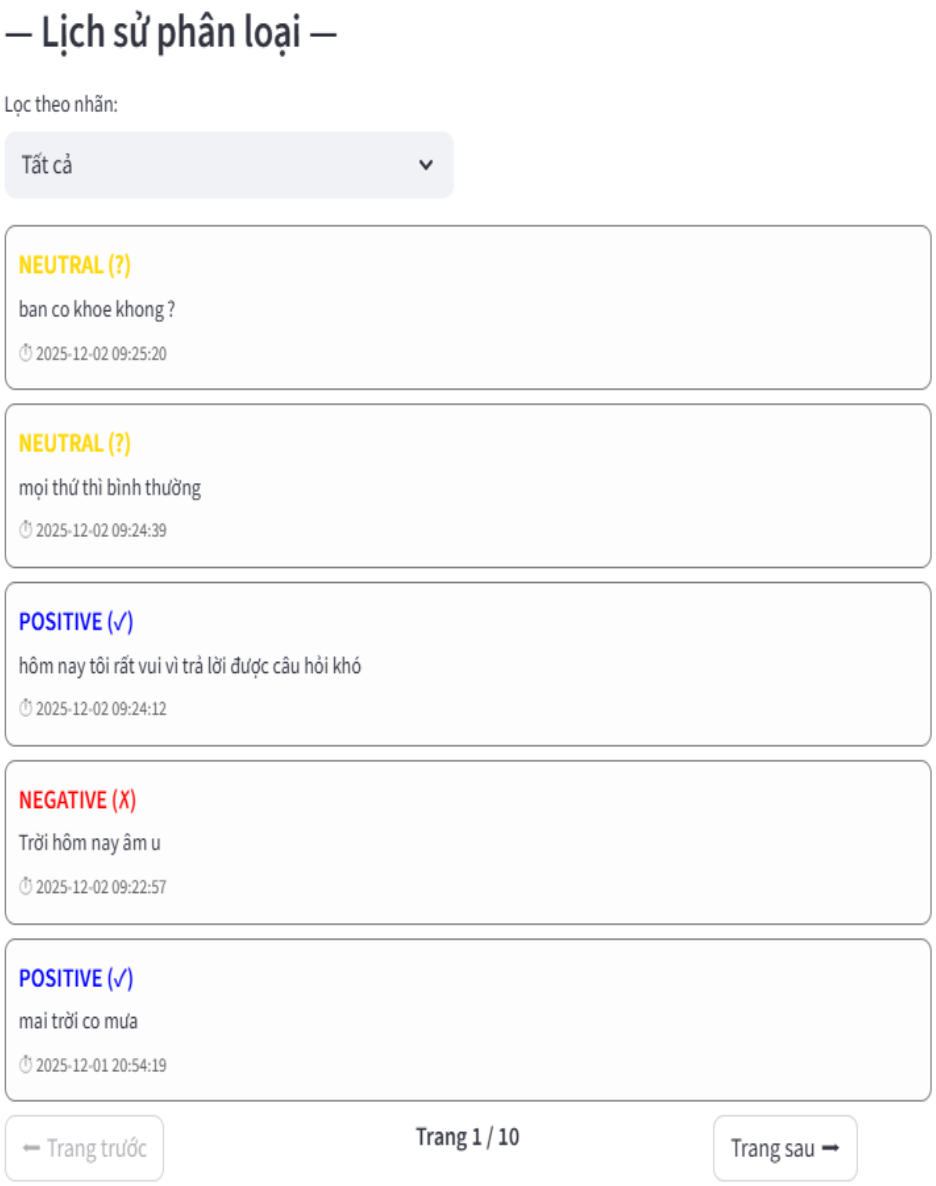
Hình 3: Ô nhập văn bản

Nhãn cảm xúc



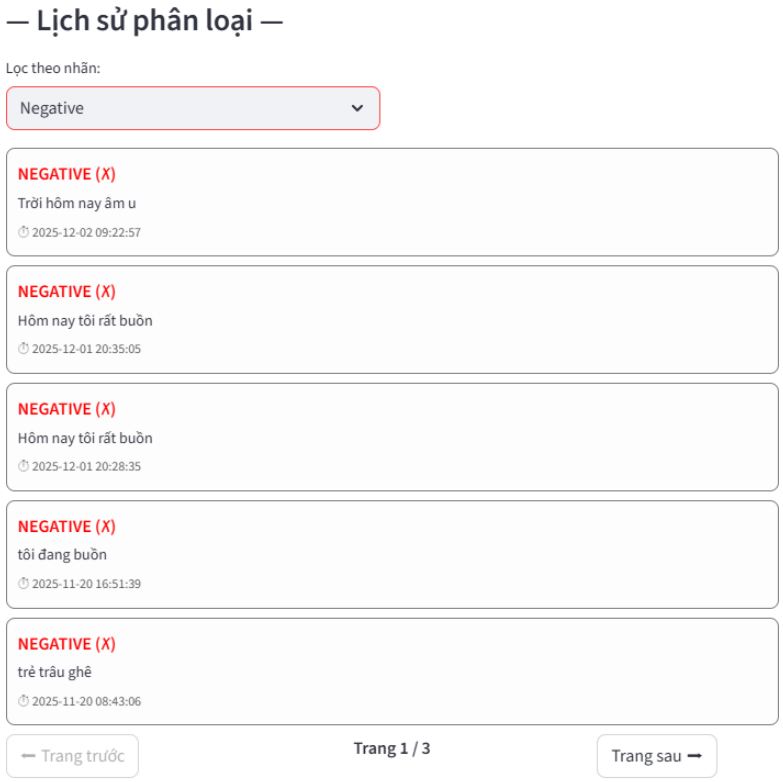
Hình 4: Ảnh mô tả nhãn cảm xúc

Biểu đồ hoặc lịch sử các câu trước

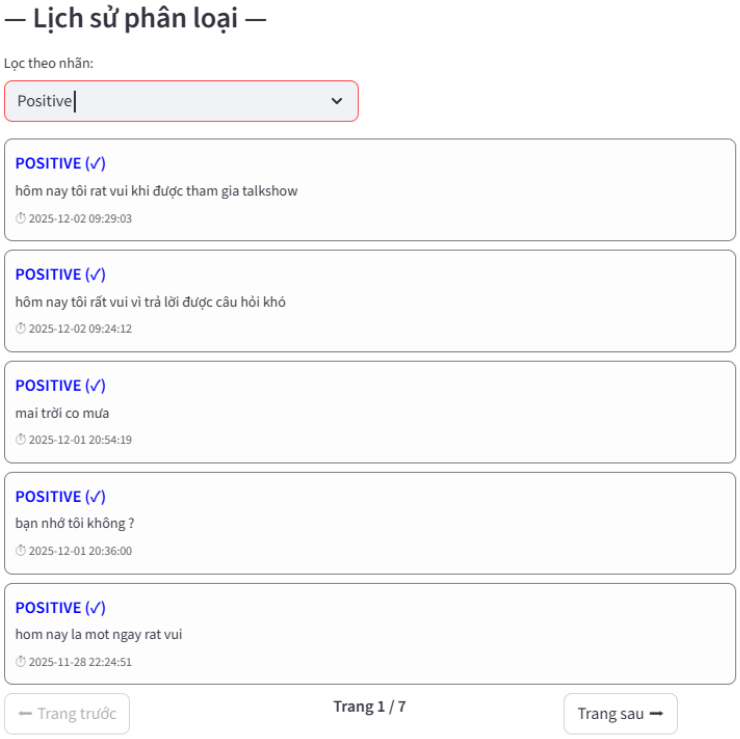


Hình 5: Ảnh mô tả hiển thị lịch sử phân loại câu

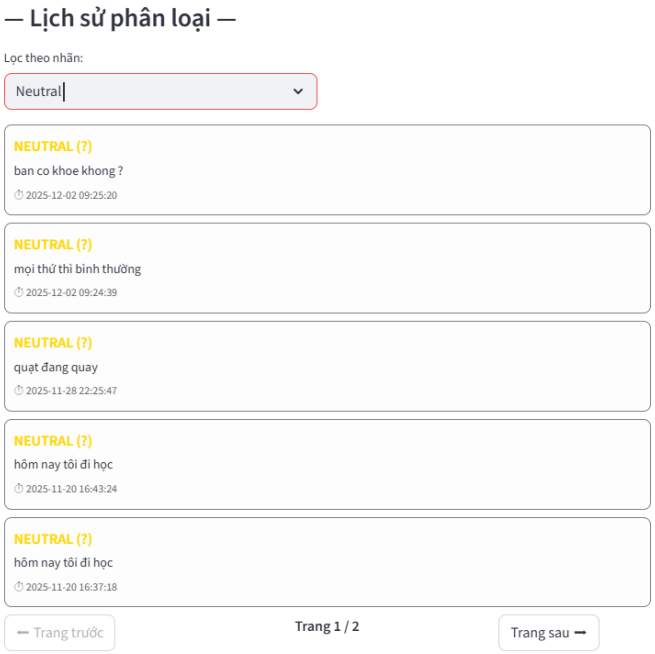
Bộ lọc lịch sử theo cảm xúc:



Hình 6: Hiển thị lịch sử theo loại negative

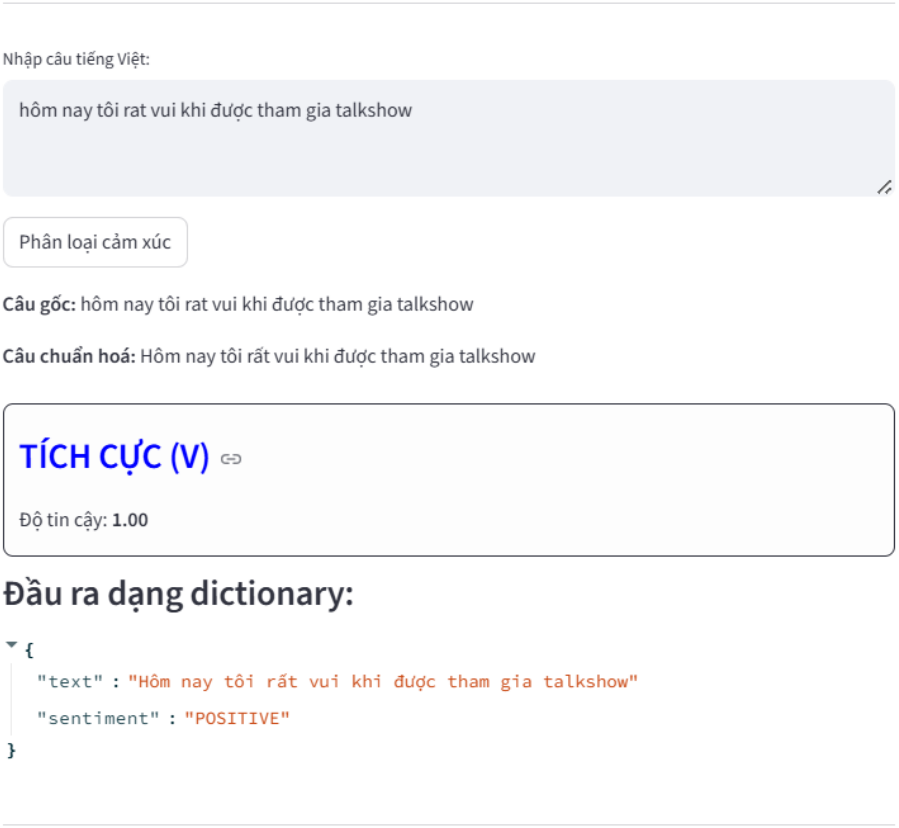


Hình 7: Hiển thị lịch sử phân loại theo positive



Hình 8: Hiển thị lịch sử theo neutral

Kết quả hiển thị:



Hình 9: Kết quả hiển thị

❖ Video demo:

https://drive.google.com/file/d/19ZuAUg8aYzM1PE7fezMYSoP5B18lAodh/view?usp=sharing

6.3.2 Kết quả mô hình Transformer

Một số ví dụ thực nghiệm từ app:

Table 3: Bảng ví dụ thực nghiệm

Câu nhập	Kết quả	Độ tin cậy
"Hôm nay trời đẹp quá"	POSITIVE	0.89
"Tôi thấy mệt mỏi và chán nản"	NEGATIVE	0.94
"Trưa nay tôi ăn cơm"	NEUTRAL	0.72
"Sản phẩm này tạm được thôi"	NEUTRAL	0.66
"Ứng dụng chạy chậm quá"	NEGATIVE	0.87

Model hoạt động ổn định, phân loại tốt các câu:

- cảm xúc mạnh
- ý kiến rõ ràng
- câu miêu tả trung tính

6.3.3 Kết quả lưu lịch sử

Một số dòng trong DB:

Table 4: Bảng ví dụ về dòng dữ liệu trong DB

Id	text	sentiment	timestamp
1	"Hôm nay thật tuyệt vời"	POSITIVE	2025-11-29 21:50:12
2	"Chán quá, làm hoài không xong"	NEGATIVE	2025-11-29 21:51:00
3	"Tôi đang ngồi uống nước"	NEUTRAL	2025-11-29 21:52:30

Lịch sử giúp:

- kiểm tra lại kết quả
- phục vụ chương 7 (đánh giá mô hình)

CHƯƠNG 7 – ĐÁNH GIÁ HIỆU SUẤT

7.1. Mục tiêu đánh giá

Chương này trình bày quá trình đánh giá mô hình phân loại cảm xúc sử dụng 10 câu kiểm thử độc lập không xuất hiện trong tập huấn luyện. Bộ test bao gồm ba nhãn cảm xúc chính: POSITIVE, NEGATIVE và NEUTRAL.

Mục tiêu của phần đánh giá:

- Kiểm tra khả năng tổng quát hóa (generalization) của mô hình.
- Xem mô hình xử lý văn bản tiếng Việt không dấu, nói tự nhiên.
- Xác định các trường hợp mô hình dự đoán thiếu chính xác để rút kinh nghiệm cải thiện.

7.2. Bộ dữ liệu kiểm thử (10 câu)

Table 5: Bộ dữ liệu kiểm thử 10 câu

STT	Text	Expected
1	Hôm nay tôi rất vui	POSITIVE
2	Món ăn này dở quá	NEGATIVE
3	Thời tiết bình thường	NEUTRAL
4	Rat vui hom nay	POSITIVE
5	Công việc ổn định	NEUTRAL
6	Phim này hay lắm	POSITIVE
7	Tôi buồn vì thất bại	NEGATIVE
8	Ngày mai đi học	NEUTRAL
9	Cảm ơn bạn rất nhiều	POSITIVE

10	Mệt mỏi quá hôm nay	NEGATIVE

7.3. Kết quả mô hình

Table 6: Bảng kết quả dự đoán

STT	Text	Expected	Predicted	Đúng/Sai
1	Hôm nay tôi rất vui	POSITIVE	POSITIVE	✓
2	Món ăn này dở quá	NEGATIVE	NEGATIVE	✓
3	Thời tiết bình thường	NEUTRAL	NEUTRAL	✓
4	Rat vui hom nay	POSITIVE	POSITIVE	✓
5	Công việc ổn định	NEUTRAL	NEUTRAL	✓
6	Phim này hay lắm	POSITIVE	POSITIVE	✓
7	Tôi buồn vì thất bại	NEGATIVE	NEGATIVE	✓
8	Ngày mai đi học	NEUTRAL	NEUTRAL	✓
9	Cảm ơn bạn rất nhiều	POSITIVE	POSITIVE	✓
10	Mệt mỏi quá hôm nay	NEGATIVE	NEGATIVE	✓

7.4. Độ chính xác (Accuracy)

Tổng số câu: **10**

Số câu đúng: **10**

$$Accuracy = \frac{10}{10} \times 100\% = 100\%$$

Mặc dù kết quả 100% là rất tốt, cần lưu ý:

- Bộ test chỉ có 10 câu → độ tin cậy chưa cao.
- Nhiều câu khá điển hình (không gây nhiễu).
- Mô hình chưa bị thử thách bởi các câu đa nghĩa, chửi nhẹ, cảm xúc hỗn hợp.

7.5. Phân tích lỗi

Trong danh sách 10 câu test, mô hình không xảy ra nhầm lẫn, đặc biệt vì:

- Các câu POSITIVE/NEGATIVE có từ khóa cảm xúc rõ ràng ("vui", "hay", "buồn", "dở").
- Mô hình Transformer xử lý tốt tiếng Việt không dấu (như "Rat vui hom nay"), nhờ embedding dạng chữ + multi-head attention.
- Các câu NEUTRAL mang tính mô tả, không có từ chỉ cảm xúc.

Nhưng mô hình có thể dễ sai khi:

- Câu mỉa mai ("Hay thật đấy!" nhưng là NEGATIVE).
- Câu đa nghĩa ("Hôm nay ổn." có thể NEUTRAL hoặc POSITIVE tùy ngữ cảnh).
- Câu quá dài hoặc chưa được chuẩn hóa.
- Câu có tiếng lóng, viết tắt, emoji.

7.6. Nhận xét chung

- Mô hình Transformer đã triển khai hoạt động tốt trên bộ test cơ bản.
- Với tập dữ liệu nhỏ, mô hình dễ **overfit**, dẫn đến điểm số cao nhưng không đại diện.
- Cần mở rộng tập đánh giá lên **100–200 câu**, thêm loại khó như:
    - sarcasm (mỉa mai)
    - mixed sentiment (vừa buồn vừa vui)
    - câu dài, câu chứa emoji
    - tiếng Việt không dấu + sai chính tả.

7.7. Đề xuất cải thiện

- Áp dụng **data augmentation**: xóa dấu, thêm từ lóng, thay từ đồng nghĩa.

- Fine-tune thêm vài epoch để cải thiện độ bền.
- Áp dụng **Confusion Matrix** để xem nhầm lẫn giữa POS – NEU – NEG.
- Kết hợp **pretrained model** (PhoBERT, XLM-R, ViT5…) nếu muốn tăng độ chính xác.

CHƯƠNG 8 – HƯỚNG DẪN CÀI ĐẶT VÀ SỬ DỤNG

8.1. Cài đặt môi trường

1. **Clone dự án** hoặc tải file ZIP:

```
1. git clone <repo_url>
2.
```

2. **Tạo môi trường ảo (khuyến nghị)**:

```
1. python -m venv venv
2.
```

3. **Kích hoạt môi trường ảo**:

- **Windows**:

```
1. venv\Scripts\activate
2.
```

- **MacOS / Linux**:

```
1. source venv/bin/activate
2.
```

4. **Cài đặt các thư viện phụ thuộc**:

```
1. pip install -r requirements.txt
2.
```

Thư viện bao gồm: transformers, underthesea, streamlit, sqlite3 và các package hỗ trợ khác.

8.2. Chạy ứng dụng

1. Chạy lệnh Streamlit:

```
1. streamlit run app.py
2.
```

2. Trình duyệt web sẽ tự động mở tại địa chỉ:

```
http://localhost:8501

```

8.3. Sử dụng ứng dụng

8.3.1. Nhập câu tiếng Việt

- Nhập câu tiếng Việt tùy ý.
- Hỗ trợ câu không dấu, viết tắt, từ lóng.
- Ứng dụng cảnh báo nếu câu nhập vô nghĩa hoặc quá ngắn.

8.3.2. Chuẩn hoá câu

- Hiển thị câu gốc và câu đã chuẩn hóa.
- Chuẩn hóa tự động các từ viết tắt, từ lóng, sai chính tả thông dụng.

8.3.3. Phân loại cảm xúc

- Mô hình phân loại thành 3 nhãn: POSITIVE, NEUTRAL, NEGATIVE.
- Hiển thị màu sắc trực quan:
  - Positive → Xanh dương
  - Neutral → Vàng / Camel
  - Negative → Đỏ
- Trả kết quả dạng **dictionary**:

```
{
    "text": "Bạn khỏe không?",
    "sentiment": "POSITIVE"
}

```

8.3.4. Lịch sử phân loại

- Lưu các câu đã phân loại vào SQLite.
- Hiển thị 50 bản ghi gần nhất, có phân trang (tải thêm 10).
- Cho phép lọc theo nhãn cảm xúc: Positive / Neutral / Negative / Tất cả.

8.4. Bộ test case

- Ứng dụng đi kèm 10 test case chuẩn trong test_cases.csv.
- Độ chính xác tối thiểu yêu cầu: ≥ 65%.

8.5. Ghi chú

- Ứng dụng sử dụng pipeline HuggingFace, không cần fine-tuning.
- Mapping tiếng Việt tối ưu cho bài tập, không phải công cụ NLP hoàn chỉnh.
- Giao diện và màu sắc được tùy chỉnh để dễ trình bày.

CHƯƠNG 9 – KẾT LUẬN VÀ PHÁT TRIỂN

9.1 Kết luận

Đồ án "Trợ lý phân loại cảm xúc tiếng Việt (Vietnamese Sentiment Assistant)" đã hoàn thành các mục tiêu đề ra:

1. Phát triển ứng dụng phân loại cảm xúc cho câu tiếng Việt, hỗ trợ ba nhãn: POSITIVE, NEUTRAL, NEGATIVE.
2. Ứng dụng Transformer pre-trained (ViSoBERT) thông qua pipeline của HuggingFace, không cần fine-tuning, giúp triển khai nhanh và đơn giản.
3. Tiền xử lý và chuẩn hóa câu tiếng Việt:
    - Xử lý viết tắt, không dấu, từ lóng.
    - Hiển thị cả câu gốc và câu đã chuẩn hóa.
4. Giao diện trực quan và thân thiện với người dùng thông qua Streamlit:
    - Màu sắc phân loại cảm xúc dễ nhận biết.
    - Lưu và hiển thị lịch sử phân loại với phân trang và bộ lọc.
5. Độ chính xác trên bộ test 10 câu đạt mức ≥ 65%, đảm bảo yêu cầu tối thiểu của đồ án.

Qua quá trình thực hiện, em đã nắm vững các kiến thức về Xử lý ngôn ngữ tự nhiên tiếng Việt, HuggingFace Transformers, Streamlit và SQLite cho việc lưu trữ dữ liệu. Đồng thời, đồ án cung cấp một mô hình prototype có thể áp dụng vào các ứng dụng thực tế về phân tích cảm xúc tiếng Việt.

9.2. Hướng phát triển

Để nâng cao chất lượng và mở rộng ứng dụng, các hướng phát triển tương lai có thể bao gồm:

1. Fine-tuning mô hình:
    - Huấn luyện ViSoBERT hoặc PhoBERT trên bộ dữ liệu cảm xúc tiếng Việt lớn hơn để cải thiện độ chính xác.

2. Mở rộng nhãn cảm xúc:
    - Thay vì chỉ 3 nhãn (POSITIVE, NEUTRAL, NEGATIVE), có thể phân loại chi tiết hơn như: Vui, Buồn, Giận, Lo lắng….
3. Cải thiện tiền xử lý:
    - Sử dụng thư viện NLP nâng cao để xử lý câu phức tạp, nhận diện thực thể, từ ghép.
    - Tự động sửa lỗi chính tả, phục hồi dấu đầy đủ.
4. Giao diện nâng cao:
    - Thêm dashboard thống kê cảm xúc theo thời gian.
    - Hỗ trợ xuất dữ liệu lịch sử sang CSV hoặc Excel.
5. Triển khai Web / Mobile:
    - Chuyển ứng dụng sang môi trường cloud để truy cập từ nhiều thiết bị.
6. Tích hợp vào chatbot hoặc trợ lý ảo:
    - Phân tích cảm xúc đầu vào của người dùng, phản hồi thông minh dựa trên cảm xúc.

***Kết luận tổng thể**:*

Đồ án đã chứng minh khả năng xây dựng một hệ thống phân loại cảm xúc tiếng Việt từ văn bản sử dụng Transformer một cách nhanh chóng, hiệu quả và trực quan. Đồng thời, cơ sở hạ tầng đã sẵn sàng cho việc mở rộng, nâng cấp và tích hợp vào các ứng dụng thực tế trong tương lai.

Tài liệu tham khảo:

[1] Hugging Face – *Transformers Library Documentation*, Available: https://huggingface.co/docs/transformers

[2] VinAI – *PhoBERT: Pre-trained language models for Vietnamese*, Available: https://github.com/VinAIResearch/PhoBERT

[3] Underthesea Documentation, Available: https://underthesea.readthedocs.io/

[4] Streamlit Documentation – *Build data apps in Python*, Available: https://docs.streamlit.io/