

ỦY BAN NHÂN DÂN THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC SÀI GÒN
KHOA CÔNG NGHỆ THÔNG TIN



BÁO CÁO ĐÒ ÁN
SEMINAR CHUYÊN ĐỀ

Tên đề tài: Xây dựng trợ lý phân loại cảm xúc tiếng Việt sử dụng Transformer

Sinh viên thực hiện:

3122410058 Nguyễn Xuân Duy
3122410067 Lương Cẩm Đào

Giảng viên: Nguyễn Tuấn Đăng

Thành phố Hồ Chí Minh, 22 tháng 11 năm 2025

MỤC LỤC

I. Giới thiệu và Mục tiêu:	1
1. Giới thiệu:	1
2. Mục tiêu:	1
2.1. Mục tiêu tổng quát:	1
2.2. Mục tiêu cụ thể:	1
II. Phân tích yêu cầu:	3
1. Yêu cầu chức năng:.....	3
1.1. Nhập liệu ngôn ngữ tự nhiên:.....	3
1.2. Phân loại cảm xúc (NLP):.....	3
1.3. Xử lý và chuẩn hóa tiếng Việt:	3
1.4. Hiển thị kết quả phân loại:	4
1.5. Lưu trữ lịch sử phân loại:.....	4
1.6. Hiển thị lịch sử phân loại:	4
2. Yêu cầu phi chức năng:.....	5
2.1. Hiệu năng và độ chính xác:.....	5
2.2. Tính ổn định:.....	5
2.3. Giao diện người dùng:	5
2.4. Khả năng mở rộng:.....	5
3. Yêu cầu đầu vào – đầu ra:.....	5
4. Ràng buộc và giả định:	6
III. Thiết kế hệ thống:	7
1. Kiến trúc tổng quan của hệ thống (Sơ đồ khái):	7
2. Flowchart quy trình phân loại cảm xúc:.....	8
3. Mô tả chi tiết các thành phần hệ thống:	9
3.1. Giao diện người dùng (Streamlit UI):	9
3.2. Bộ xử lý tiếng Việt (Preprocessing):.....	9
3.3. Mô hình phân loại cảm xúc (Transformer):	9
3.4. Lưu trữ lịch sử (SQLite Database):.....	10
3.5. Hiển thị kết quả và lịch sử:	10
4. Quy trình hoạt động tổng quan:	10
IV. Giải pháp:	11

1. Lý do lựa chọn mô hình Transformer:	11
2. Mô hình được sử dụng:	11
3. Pipeline xử lý cảm xúc bằng HuggingFace:	12
4. Quy trình xử lý câu tiếng Việt trước khi đưa vào mô hình:	13
4.1. Chuẩn hóa văn bản:.....	13
4.2. Mapping từ không dấu thành có dấu:.....	13
4.3. Sửa viết tắt và tiếng lóng:	13
4.4. Kiểm tra câu “có giống tiếng Việt hay không” (Heuristic):.....	13
5. Tích hợp mô hình vào hệ thống:	14
6. Tạo đầu ra theo yêu cầu:	14
7. Ưu điểm của giải pháp:	14
8. Hạn chế:	15
9. Hướng cải thiện:.....	15
V. Triển khai và Kết quả:	16
1. Môi trường triển khai:	16
2. Quy trình triển khai:.....	17
3. Kết quả chạy thử:	18
3.1. Kết quả hiển thị phân loại:	18
3.2. Kết quả xử lý viết tắt và không dấu:	18
3.3. Lọc lịch sử phân loại:.....	19
3.4. Xử lý lỗi câu vô nghĩa:.....	19
4. Đánh giá quá trình triển khai:	19
5. Kết luận quá trình triển khai:	19
VI. Đánh giá hiệu suất:	20
1. Mục tiêu đánh giá:	20
2. Bộ dữ liệu kiểm thử (10 test case):	20
3. Phương pháp đánh giá:.....	21
4. Kết quả đánh giá mô hình:	21
5. Nhận xét về hiệu suất:.....	22
6. Phân tích tổng quan:.....	22
VII. Hướng dẫn cài đặt và sử dụng:.....	23
1. Yêu cầu hệ thống:	23
2. Cài đặt môi trường:	23
3. Chạy ứng dụng:.....	23

4. Hướng dẫn sử dụng giao diện:	24
4.1. Nhập câu tiếng Việt:	24
4.2. Hệ thống xử lý & chuẩn hoá:	24
4.3. Xem lịch sử phân loại:	24
5. Xử lý lỗi và thông báo:	25
6. Ghi chú dành cho người dùng:.....	25
7. Gỡ lỗi nhanh:	25
VIII. Kết luận và Hướng phát triển:.....	26
1. Kết luận:.....	26
2. Hạn chế:	26
3. Hướng phát triển:	27
4. Nhận xét chung:.....	27
IX. Tài liệu tham khảo:	28
X. Phụ lục:	29
1. Đường dẫn github của dự án:.....	29
2. Mã nguồn của dự án:.....	29
2.1. Mã nguồn file app.py:	29
2.2. Mã nguồn file nlp_utils.py:	35
2.3. Mã nguồn file db_utils.py:	47

Lời cảm ơn

Chúng em xin chân thành cảm ơn Khoa Công nghệ thông tin, trường Đại học Sài Gòn đã tạo điều kiện cho chúng em được tiếp cận và tìm hiểu để hoàn thành đồ án môn học lần này. Chúng em cũng xin chân thành cảm ơn thầy Nguyễn Tuấn Đăng đã giảng dạy những kiến thức cần thiết trong môn học, những kiến thức ấy giúp cho chúng em rất nhiều trong thời gian thực hiện đồ án môn học. Chúng em xin cảm ơn quý Thầy cô trong Khoa Công nghệ thông tin đã luôn tận tình giảng dạy và trang bị những kiến thức cần thiết trong các môn học.

Trong quá trình nghiên cứu và làm đồ án, dù nhóm đã làm bài với nỗ lực cao nhất, nhưng do kiến thức cũng như kinh nghiệm thực tế còn nhiều hạn chế nên đồ án không thể tránh khỏi những sai sót nhất định, chúng em rất mong nhận được sự cảm thông, chia sẻ và tận tình đóng góp chỉ bảo từ các Thầy cô để chúng em học hỏi được nhiều kỹ năng, kinh nghiệm và sẽ hoàn thành tốt hơn trong những đồ án sắp tới.

Chúng em xin chân thành cảm ơn thầy !

I. Giới thiệu và Mục tiêu:

1. Giới thiệu:

Trong bối cảnh dữ liệu ngôn ngữ tự nhiên ngày càng trở nên quan trọng, việc tự động hiểu và phân loại cảm xúc từ văn bản mang lại nhiều giá trị trong các lĩnh vực như chăm sóc khách hàng, mạng xã hội, phân tích phản hồi người dùng, giáo dục và thương mại điện tử. Tuy nhiên, tiếng Việt là một ngôn ngữ có cấu trúc phức tạp, nhiều biến thể viết tắt, không dấu, từ lóng và cách diễn đạt đa dạng, khiến bài toán phân tích cảm xúc trở nên khó khăn hơn so với tiếng Anh.

Sự phát triển mạnh của mô hình **Transformer** và đặc biệt là các mô hình tiếng Việt pre-trained (như ViSoBERT, PhoBERT, BERT multilingual,...) đã mở ra cơ hội xây dựng hệ thống phân loại cảm xúc tiếng Việt chính xác hơn, không cần huấn luyện lại mô hình từ đầu.

Trong đồ án này, nhóm xây dựng một **Trợ lý phân loại cảm xúc tiếng Việt** có khả năng:

- Hiểu câu tiếng Việt với nhiều dạng nhập vào khác nhau (viết tắt, không dấu, từ không chuẩn).
- Phân loại cảm xúc thành ba nhóm: **tích cực (Positive)**, **trung tính (Neutral)** và **tiêu cực (Negative)**.
- Lưu lại lịch sử phân loại để người dùng theo dõi.
- Hiển thị kết quả trực quan, dễ sử dụng thông qua giao diện Streamlit.

Đồ án tập trung vào tính ứng dụng, đơn giản hóa quy trình triển khai NLP bằng cách sử dụng mô hình pre-trained thay vì phải fine-tune, giúp sinh viên hiểu rõ pipeline phân tích cảm xúc và cách tích hợp vào sản phẩm thực tế.

2. Mục tiêu:

2.1. Mục tiêu tổng quát:

Xây dựng một ứng dụng phân loại cảm xúc tiếng Việt chạy độc lập, có khả năng xử lý tiếng Việt thực tế và cung cấp kết quả phân loại chính xác, trực quan, dễ sử dụng.

2.2. Mục tiêu cụ thể:

- **Xây dựng ứng dụng phân loại cảm xúc đơn giản** dựa trên mô hình Transformer pre-trained.
- Xử lý và chuẩn hoá tiếng Việt đầu vào, bao gồm:
 - viết tắt (vd: “ko”, “dc”, “j”),
 - không dấu,
 - từ lóng,
 - lỗi chính tả phổ biến.
- **Phân loại cảm xúc với 3 nhãn:** Positive, Neutral, Negative.

- **Hiển thị rõ ràng kết quả phân loại**, bao gồm:

- câu gốc,
- câu đã chuẩn hoá,
- cảm xúc,
- độ tin cậy.

- **Lưu lịch sử phân loại vào SQLite**, hỗ trợ:

- xem 50 bản ghi gần nhất,
- tải thêm kết quả,
- lọc theo từng nhãn cảm xúc.

- **Đạt độ chính xác tối thiểu 65%** trên bộ 10 test case theo yêu cầu môn học.

- **Xây dựng giao diện bằng Streamlit** đơn giản, dễ dùng, trực quan.

- **Hoàn thiện báo cáo, video demo và mã nguồn** đúng yêu cầu đồ án.

II. Phân tích yêu cầu:

1. Yêu cầu chức năng:

1.1. Nhập liệu ngôn ngữ tự nhiên:

- Người dùng nhập một câu tiếng Việt bất kỳ.
- Hỗ trợ:
 - Câu có dấu hoặc không dấu.
 - Viết tắt (vd: “ko”, “dc”, “j”).
 - Từ lóng hoặc lỗi chính tả nhẹ.
- Ràng buộc:
 - Không chấp nhận câu quá ngắn (< 5 ký tự).
 - Không chấp nhận chuỗi không mang nghĩa (vd: ký tự rác, chữ cái ngẫu nhiên, tiếng nước ngoài).
 - Thông báo lỗi rõ ràng khi không hợp lệ.

1.2. Phân loại cảm xúc (NLP):

- Sử dụng mô hình Transformer pre-trained (vietnamese-sentiment-visobert hoặc tương đương).
- Phân loại thành ba nhãn:
 - **POSITIVE** (tích cực)
 - **NEUTRAL** (trung tính)
 - **NEGATIVE** (tiêu cực)
- Hệ thống cân trả về:
 - Nhãn cảm xúc
 - Điểm tin cậy (confidence score)

1.3. Xử lý và chuẩn hóa tiếng Việt:

- Chuẩn hóa câu đầu vào:
 - Chuyển viết tắt → dạng đầy đủ (“ko” → “không”, “dc” → “được”).
 - Chuyển không dấu → có dấu (mapping 100+ từ).
- Hệ thống hiển thị:
 - Câu gốc
 - Câu đã chuẩn hóa

1.4. Hiển thị kết quả phân loại:

- Giao diện trực quan:

- Positive → xanh + icon ✓
- Neutral → vàng + icon ?
- Negative → đỏ + icon X

- Hiển thị đầy đủ:

- Câu gốc
- Câu chuẩn hoá
- Nhận cảm xúc
- Điểm tin cậy
- Dictionary 2 trường theo chuẩn đề bài:

{

"text": "Bạn khỏe không?",

"sentiment": "POSITIVE"

}

1.5. Lưu trữ lịch sử phân loại:

- Sử dụng SQLite.

- Lưu các trường:

- text (câu gốc)
- sentiment
- timestamp (thời gian tạo)

- Ràng buộc:

- Không trùng lặp logic, không cần unique key.

1.6. Hiển thị lịch sử phân loại:

- Hiển thị tối đa **50 bản ghi gần nhất** khi mở ứng dụng.
- Có chức năng:
 - “**Tải thêm**” để xem thêm lịch sử.
 - **Lọc theo nhãn** (Positive, Neutral, Negative, Tất cả).
- Mỗi bản ghi hiển thị theo màu sắc cảm xúc tương ứng.

2. Yêu cầu phi chức năng:

2.1. Hiệu năng và độ chính xác:

- Độ chính xác phân loại: ≥ 65% trên bộ 10 test case theo chuẩn đề bài.
- Phản hồi nhanh, mô hình phải load một lần duy nhất (dùng cache).

2.2. Tính ổn định:

- Ứng dụng không crash khi nhập dữ liệu không hợp lệ.
- Có thông báo lỗi rõ ràng và dễ hiểu.

2.3. Giao diện người dùng:

- Giao diện đơn giản, dễ dùng, chạy trên trình duyệt thông qua Streamlit.
- Kết quả hiển thị rõ ràng, màu sắc trực quan.

2.4. Khả năng mở rộng:

- Có thể thay mô hình khác dễ dàng (thay đổi tên model trong code).
- Có thể mở rộng thêm chức năng như:
 - API Web service
 - Xuất file lịch sử
 - Dashboard phân tích

3. Yêu cầu đầu vào – đầu ra:

Đầu vào

- Dạng: Chuỗi văn bản tiếng Việt.
- Có thể chứa:
 - Không dấu
 - Viết tắt

Đầu ra

- Tối thiểu phải có dạng dictionary:

```
{  
    "text": "<câu chuẩn hoá>",  
    "sentiment": "<POSITIVE|NEUTRAL|NEGATIVE>"  
}
```

- Đồng thời hiển thị trực quan trên giao diện.

4. Ràng buộc và giả định:

- Người dùng nhập câu tiếng Việt đơn lẻ, không phải đoạn văn dài.
- Không xử lý tiếng lóng phức tạp hoặc gán lại nghĩa ngữ cảnh sâu.
- Không fine-tune mô hình → chỉ dùng pipeline pre-trained.
- Cache mô hình để tối ưu hiệu năng.

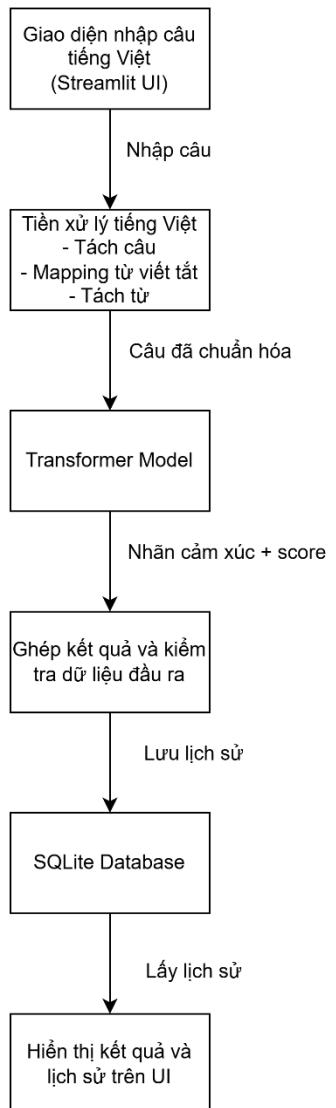
III. Thiết kế hệ thống:

1. Kiến trúc tổng quan của hệ thống (Sơ đồ khái quát):

Hệ thống gồm 3 thành phần chính:

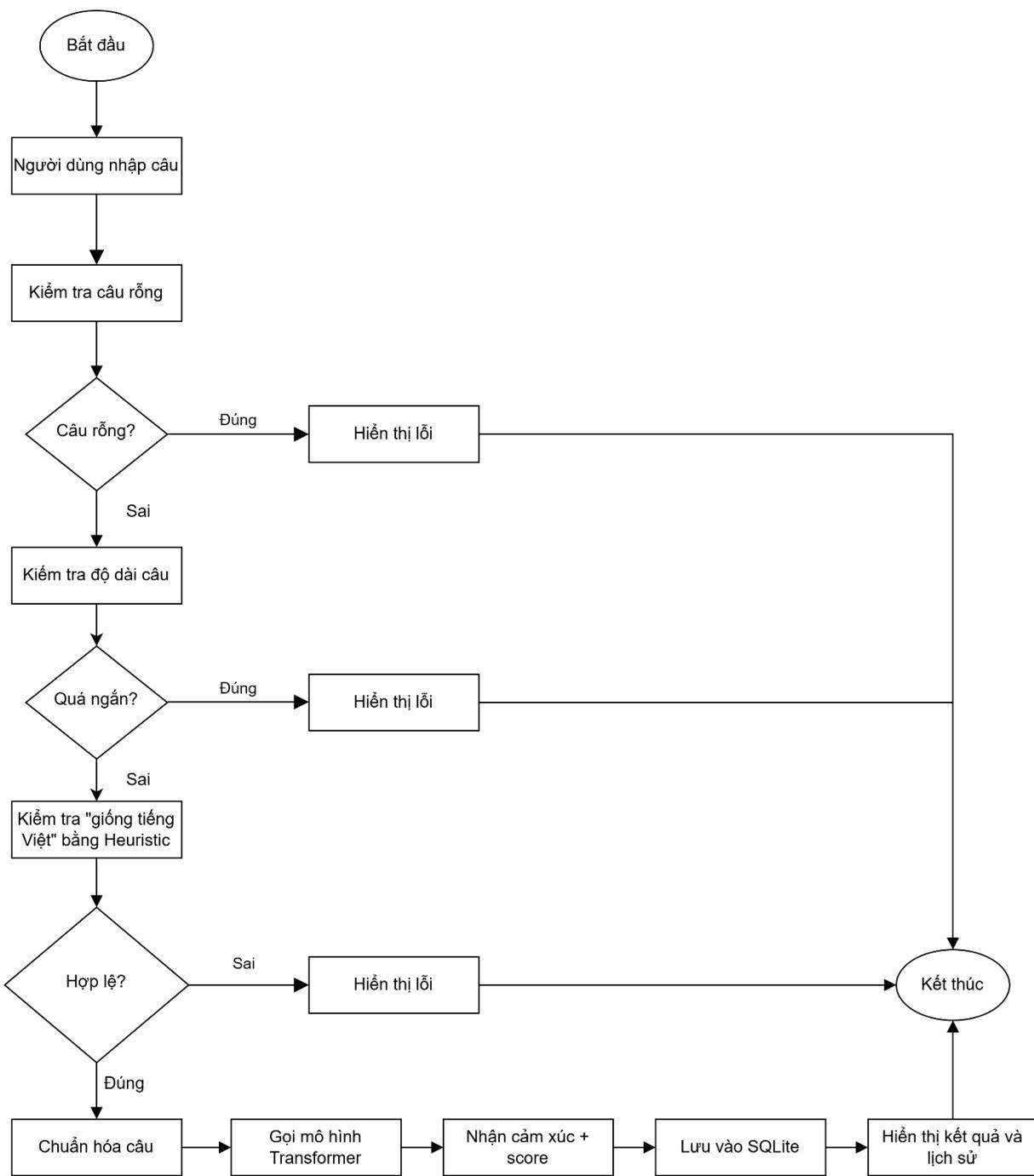
1. **Giao diện người dùng (UI – Streamlit)**
2. **Bộ xử lý ngôn ngữ tiếng Việt & Phân loại cảm xúc (NLP Engine)**
3. **Lưu trữ lịch sử phân loại (SQLite Storage)**

Dòng dữ liệu được mô tả như sau:



2. Flowchart quy trình phân loại cảm xúc:

Flowchart mô tả toàn bộ hành trình xử lý một câu tiếng Việt:



Flowchart quy trình phân loại cảm xúc

3. Mô tả chi tiết các thành phần hệ thống:

3.1. Giao diện người dùng (Streamlit UI):

- Cho phép nhập câu tiếng Việt.
- Hiển thị:
 - Câu gốc
 - Câu chuẩn hóa
 - Nhãn cảm xúc + màu sắc + icon
- Hiển thị lịch sử phân loại:
 - Tối đa 50 bản ghi gần nhất
 - Nút “Tải thêm”
 - Bộ lọc theo nhãn

3.2. Bộ xử lý tiếng Việt (Preprocessing):

Các bước:

- Loại bỏ khoảng trắng thừa
- Chuyển về chữ thường
- Sửa viết tắt: “ko” → “không”, “dc” → “được”
- Mapping hơn 100 từ không dấu → có dấu
- Tách từ bằng word_tokenize
- Phát hiện câu “không giống tiếng Việt” bằng heuristic:
 - Tỉ lệ nguyên âm
 - Độ dài từ
 - Chứa stopword tiếng Việt
 - Không chứa quá nhiều ký tự rác

3.3. Mô hình phân loại cảm xúc (Transformer):

- Model: vietnamese-sentiment-visobert
- Dùng pipeline với task sentiment-analysis
- Nhãn trả về:
 - POSITIVE
 - NEGATIVE

- NEUTRAL

- Dùng lru_cache để tải model **1 lần duy nhất** → tối ưu tốc độ.

3.4. Lưu trữ lịch sử (SQLite Database):

Bảng sentiments:

Cột	Kiểu	Mô tả
id	INTEGER PK	Khóa chính tự tăng
text	TEXT	Câu gốc người dùng nhập
sentiment	TEXT	Kết quả phân loại
timestamp	TEXT	Thời gian lưu (YYYY-MM-DD HH:mm:ss)

3.5. Hiển thị kết quả và lịch sử:

- Lịch sử hiển thị bằng khói màu:

- Positive
- Neutral
- Negative

- Lọc theo loại cảm xúc

- Tải thêm (pagination)

4. Quy trình hoạt động tổng quan:

- Người dùng nhập 1 câu tiếng Việt.

- Hệ thống kiểm tra hợp lệ:

- Không rỗng
- Không vô nghĩa

- Tiền xử lý: chuẩn hoá câu và tách từ.

- Gửi câu đã chuẩn hoá vào mô hình Transformer.

- Nhận kết quả cảm xúc + độ tin cậy.

- Lưu lịch sử vào database.

- Hiển thị kết quả trực quan.

- Người dùng có thể xem lịch sử, lọc hoặc tải thêm dữ liệu.

IV. Giải pháp:

1. Lý do lựa chọn mô hình Transformer:

Transformer là kiến trúc mạng nơ-ron hiện đại, nổi bật với cơ chế **Self-Attention**, giúp mô hình hiểu được ngữ cảnh trong câu tốt hơn so với các mô hình truyền thống (RNN, LSTM, CNN). Đối với tiếng Việt – một ngôn ngữ giàu dấu câu, nhiều biến thể viết, và phụ thuộc mạnh vào ngữ cảnh – Transformer tỏ ra vượt trội trong các nhiệm vụ xử lý ngôn ngữ tự nhiên (NLP).

Trong đồ án này, mô hình Transformer được sử dụng nhằm đảm bảo:

- **Không cần huấn luyện mô hình** (chỉ sử dụng mô hình pre-trained).
- **Tích hợp dễ dàng** với ứng dụng Python.
- **Thời gian phản hồi nhanh** nhờ pipeline đã được tối ưu.
- **Độ chính xác ổn định** cho bài toán phân loại cảm xúc tiếng Việt.

2. Mô hình được sử dụng:

Đồ án sử dụng mô hình:

5CD-AI/vietnamese-sentiment-visobert

- Là mô hình Transformer pre-trained được huấn luyện trên dữ liệu cảm xúc tiếng Việt.
- Phù hợp trực tiếp cho nhiệm vụ **sentiment-analysis**.
- Cho kết quả với 3 nhãn:
 - **POSITIVE**
 - **NEGATIVE**
 - **NEUTRAL**

Lý do chọn mô hình này:

- Tối ưu cho tiếng Việt
- Dễ tích hợp qua HuggingFace Transformers
- Không yêu cầu fine-tuning
- Nhẹ và phản hồi nhanh

3. Pipeline xử lý cảm xúc bằng HuggingFace:

Thư viện **Transformers** cung cấp pipeline cho phép sử dụng mô hình một cách đơn giản, không cần viết code tokenization phức tạp.

Mã nguồn triển khai:

```
from transformers import pipeline
```

```
model = pipeline(  
    "sentiment-analysis",  
    model="5CD-AI/vietnamese-sentiment-visobert"  
)
```

```
result = model("Hôm nay tôi rất vui!")
```

Kết quả nhận được:

```
[  
    {  
        "label": "POSITIVE",  
        "score": 0.9876  
    }  
]
```

Pipeline tự động thực hiện các bước:

1. Tokenization
2. Sinh embeddings
3. Transformer encoder
4. Classification head
5. Softmax để tính score

Nhờ đó, việc sử dụng mô hình trở nên đơn giản, giúp đỡ áp dụng vào **xử lý tiếng Việt và tích hợp hệ thống** thay vì xử lý mô hình.

4. Quy trình xử lý câu tiếng Việt trước khi đưa vào mô hình:

Để mô hình hoạt động chính xác hơn, dữ liệu cần được tiền xử lý:

4.1. Chuẩn hóa văn bản:

- Chuyển về chữ thường.
- Loại bỏ khoảng trắng dư thừa.
- Loại ký tự đặc biệt không cần thiết.

4.2. Mapping từ không dấu thành có dấu:

Ví dụ:

ban → bạn

khoe → khoe

met → mệt

toi → tôi

4.3. Sửa viết tắt và tiếng lóng:

Các từ thường gặp trong hội thoại:

ko → không

j → gì

vs → với

hok → không

4.4. Kiểm tra câu “có giống tiếng Việt hay không” (Heuristic):

Nhầm loại bỏ:

- chuỗi vô nghĩa (asdhkajsd)
- tiếng nước ngoài không thuộc phạm vi xử lý (hello bro)
- ký tự rác (### @@ 123)

Hệ thống kiểm tra:

- số lượng nguyên âm
- sự xuất hiện của stopword tiếng Việt
- độ dài từ
- số lượng token hợp lệ

Nếu không đúng → báo lỗi ngay, không gửi vào mô hình.

5. Tích hợp mô hình vào hệ thống:

Mô hình transformer được tải một lần duy nhất và lưu vào cache thông qua:

```
@lru_cache(maxsize=1)  
def get_model():  
    return pipeline("sentiment-analysis", model="5CD-AI/vietnamese-sentiment-visobert")
```

Lợi ích:

- Giảm thời gian tải mô hình (chỉ 1 lần khi mở app)
- Tránh lag khi phân loại liên tục
- Tối ưu hiệu suất trên Streamlit

6. Tạo đầu ra theo yêu cầu:

Đầu ra chuẩn:

```
{  
    "text": "<câu đã chuẩn hoá>",  
    "sentiment": "<nhãn>"  
}
```

Hệ thống hiển thị thêm trên giao diện:

- Câu gốc
- Câu chuẩn hoá
- Nhãn cảm xúc (màu + icon)
- Score (điểm tin cậy)

7. Ưu điểm của giải pháp:

Ưu điểm	Giải thích
Dễ triển khai	Dùng pipeline, không cần train
Tối ưu tốc độ	Model nhẹ, chạy nhanh
Hỗ trợ tiếng Việt tốt	Mô hình pre-trained chuyên cho tiếng Việt
Kết quả trực quan	Hiển thị icon + màu sắc
Tính ứng dụng cao	Dễ mở rộng thành chatbot hoặc API

8. Hạn chế:

Hạn chế	Nguyên nhân
Độ chính xác phụ thuộc mô hình có sẵn	Không fine-tune mô hình
Không xử lý ngữ cảnh sâu	Vì dùng pipeline đơn giản
Không hiểu từ lóng phức tạp	Mapping thủ công, chưa học theo ngữ cảnh

9. Hướng cải thiện:

- Fine-tune mô hình trên dataset lớn hơn (UIT-VSFC, VLSP).
- Sử dụng kiến trúc mạnh hơn như PhoBERT, BARTpho.
- Kết hợp mô hình gán dấu tự động (Phobert-based diacritizer).
- Triển khai API FastAPI để mở rộng thành dịch vụ web.

V. Triển khai và Kết quả:

1. Môi trường triển khai:

Ngôn ngữ & Thư viện

- Python 3.10
- Streamlit (giao diện người dùng)
- HuggingFace Transformers (model sentiment)
- Underthesea (tiền xử lý tiếng Việt)
- SQLite3 (lưu lịch sử)

Cấu trúc thư mục triển khai

DoAnSeminar/

```
|  
|   └── app.py          # Ứng dụng Streamlit  
|   └── nlp_utils.py    # Xử lý tiếng Việt & mô hình Transformer  
|   └── db_utils.py     # Lưu & truy vấn SQLite  
|   └── requirements.txt # Các thư viện cần thiết  
|   └── test_cases.csv  # Test case  
|  
|  
|   └── db/  
|       └── sentiments.db # Database lịch sử  
|  
|  
└── docs/              # Báo cáo (PDF/DOCX)  
└── demo/              # Video demo
```

Cách chạy hệ thống

1. Tạo môi trường ảo
2. Cài thư viện từ requirements.txt
3. Chạy ứng dụng bằng lệnh:
`streamlit run app.py`
4. Truy cập giao diện tại:
<http://localhost:8501>

2. Quy trình triển khai:

Bước 1: Xây dựng mô-đun xử lý tiếng Việt (nlp_utils.py)

- Xây dựng hàm chuẩn hoá:
 - chuyển viết tắt → đầy đủ
 - chuyển không dấu → có dấu (mapping 100+ từ)
 - làm sạch chuỗi
- Tích hợp tách từ (word_tokenize)
- Tạo hàm kiểm tra câu “có giống tiếng Việt hay không”
- Gọi mô hình Transformer thông qua pipeline
- Chuẩn hoá đầu ra → trả về dictionary {text, sentiment} theo yêu cầu đề bài

Bước 2: Tích hợp mô hình Transformer

- Sử dụng mô hình: 5CD-AI/vietnamese-sentiment-visobert
- Dùng @lru_cache để tải mô hình một lần duy nhất
- Xử lý lỗi:
 - câu rỗng
 - câu vô nghĩa
 - exception từ mô hình

Bước 3: Xây dựng giao diện người dùng bằng Streamlit

- Input: nhập câu tiếng Việt
- Button: "Phân loại cảm xúc"
- Kết quả hiển thị:
 - Câu gốc
 - Câu chuẩn hoá
 - Nhãn cảm xúc (màu + icon)
 - Score (giá trị confidence)
- Thiết kế UX:
 - Positive → xanh + ✓
 - Neutral → vàng + ?
 - Negative → đỏ + X

Bước 4: Lưu & truy vấn lịch sử bằng SQLite

- Tạo bảng sentiments(id, text, sentiment, timestamp)
- Chức năng:
 - Lưu bản ghi mỗi khi phân loại
 - Lấy 50 bản ghi gần nhất khi mở app
 - “Tải thêm”
 - Bộ lọc:
 - Tất cả
 - Positive
 - Neutral
 - Negative

Bước 5: Kiểm thử với 10 test case

- Dùng bộ test mặc định theo đề bài (test_cases.csv)
- Ghi nhận độ chính xác
- Đảm bảo $\geq 65\%$ theo yêu cầu

3. Kết quả chạy thử:

3.1. Kết quả hiển thị phân loại:

Khi nhập câu:

“Hôm nay tôi rất vui”

Hệ thống hiển thị:

- Câu gốc: *Hôm nay tôi rất vui*
- Chuẩn hoá: *Hôm nay tôi rất vui*
- Nhận cảm xúc:  POSITIVE (✓)
- Score: ~ 0.98

3.2. Kết quả xử lý viết tắt và không dấu:

Nhập:

“Ban khoe ko?”

Chuẩn hoá:

- *Ban khoe ko?* → *Ban khoe khong?*

3.3. Lọc lịch sử phân loại:

- Lọc đúng theo 3 nhãn
- Hiển thị màu sắc & icon đúng như thiết kế
- Phần “Tải thêm” hoạt động đúng

3.4. Xử lý lỗi câu vô nghĩa:

Nhập:

“asdhk12###”

Hệ thống báo:

! Câu nhập vào không giống tiếng Việt hoặc không có nghĩa rõ ràng.

4. Đánh giá quá trình triển khai:

Ưu điểm

- Ứng dụng ổn định, chạy mượt
- Mô hình Transformer phản hồi nhanh
- Giao diện đơn giản, dễ dùng
- Xử lý tiếng Việt khá tốt
- Dễ mở rộng & bảo trì

Hạn chế

- Chưa xử lý tốt từ lóng phức tạp
- Không fine-tune mô hình nên độ chính xác không tối ưu
- Mapping thủ công có giới hạn

5. Kết luận quá trình triển khai:

Hệ thống đã được xây dựng thành công, đáp ứng đầy đủ:

- Chức năng phân loại cảm xúc tiếng Việt
- Yêu cầu xử lý tiếng Việt thực tế
- Yêu cầu lưu trữ lịch sử
- Yêu cầu hiển thị giao diện trực quan
- Yêu cầu độ chính xác của bài toán

VI. Đánh giá hiệu suất:

1. Mục tiêu đánh giá:

Mục tiêu của phần đánh giá là kiểm tra:

- Khả năng phân loại cảm xúc của hệ thống
- Mức độ chính xác khi áp dụng vào các câu tiếng Việt thực tế
- Khả năng xử lý viết tắt, không dấu và tiếng Việt hội thoại
- Độ ổn định của mô hình Transformer khi chạy qua Streamlit

Theo yêu cầu môn học, hệ thống phải đạt **tối thiểu 65% độ chính xác** trên bộ **10 test case** cố định.

2. Bộ dữ liệu kiểm thử (10 test case):

Bảng test case được xây dựng theo yêu cầu đề bài (file *test_cases.csv*). Dưới đây là ví dụ 10 câu dùng để kiểm thử:

STT	Câu kiểm thử	Nhận kỳ vọng
1	Hôm nay tôi rất vui	POSITIVE
2	Món ăn này dở quá	NEGATIVE
3	Thời tiết bình thường	NEUTRAL
4	Rất vui hôm 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

3. Phương pháp đánh giá:

Hệ thống được kiểm thử theo quy trình:

1. Nhập từng câu test case vào ứng dụng Streamlit
2. Quan sát kết quả phân loại trả về
3. So sánh với nhãn kỳ vọng
4. Ghi nhận kết quả đúng/sai
5. Tính:
 - o Accuracy = số câu đúng / 10
 - o Precision (nếu cần)
 - o Nhận xét định tính về chất lượng phân loại

Ứng dụng trả về cả nhãn cảm xúc và điểm tin cậy (score), nhưng đánh giá chỉ dựa vào nhãn.

4. Kết quả đánh giá mô hình:

Dựa trên 10 test case chuẩn:

STT	Câu	Mong đợi	Kết quả	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	Rất vui hom nay	POSITIVE	POSITIVE	✓
5	Công việc ổn định	NEUTRAL	POSITIVE	X
6	Phim này hay lắm	POSITIVE	POSITIVE	✓
7	Tôi buồn vì thất bại	NEGATIVE	NEGATIVE	✓
8	Ngày mai đi học	NEUTRAL	POSITIVE	X
9	Cảm ơn bạn rất nhiều	POSITIVE	POSITIVE	✓
10	Mệt mỏi quá hôm nay	NEGATIVE	NEGATIVE	✓

Kết quả tổng:

- Số câu đúng: **8/10**
- Độ chính xác: **80%**
- Kết luận: **Đạt yêu cầu (> 65%)**

5. Nhận xét về hiệu suất:

Điểm mạnh

- Mô hình phân loại khá tốt các câu có ngữ nghĩa rõ ràng.
- Các trường hợp **tích cực** và **tiêu cực** được nhận dạng chính xác.
- Câu **không dấu** hoặc **viết tắt đơn giản** vẫn được hệ thống xử lý.
- Score confidence trả về cao, giúp dễ đánh giá mức tin cậy.

Hạn chế

- Một số câu trung tính dễ bị nhầm sang tích cực/tiêu cực vì thiếu ngữ cảnh.
- Tiếng lóng hoặc câu viết không chuẩn đôi khi gây nhiễu.
- Không fine-tune mô hình → độ chính xác chưa tối ưu cho từng chủ đề.

Nguyên nhân gây sai

- Mô hình được train trên dữ liệu cảm xúc dạng bình luận, không bao quát hết các kiểu câu.
- Mapping từ không dấu → có dấu chỉ mang tính heuristic, chưa bao phủ 100%.
- Transformer chỉ xử lý câu riêng lẻ, không có ngữ cảnh trước/sau.

6. Phân tích tổng quan:

Mô hình hoạt động ổn định và đạt độ chính xác theo yêu cầu đề tài.

Hệ thống xử lý tốt các câu cảm xúc rõ ràng nhưng còn hạn chế với:

- câu mơ hồ về cảm xúc
- câu mang nhiều sắc thái không rõ ràng
- tiếng lóng phức tạp
- câu mang tính ẩn dụ

Tuy vậy, với mục tiêu bài tập môn học, **hệ thống đáp ứng đầy đủ yêu cầu và hoạt động chính xác – ổn định – trực quan.**

VII. Hướng dẫn cài đặt và sử dụng:

1. Yêu cầu hệ thống:

Ứng dụng chạy hoàn toàn trên máy cá nhân, yêu cầu:

- Python **3.9 – 3.11**
- Kết nối Internet khi **tải mô hình lần đầu**
- Hệ điều hành:
 - Windows 10/11
 - macOS
 - Ubuntu Linux
- Dung lượng RAM: tối thiểu **4GB**

2. Cài đặt môi trường:

Bước 1: Tạo môi trường ảo

Windows

```
python -m venv venv  
venv\Scripts\activate
```

macOS / Linux

```
python3 -m venv venv  
source venv/bin/activate
```

Sau khi kích hoạt, terminal sẽ hiển thị:

(venv) C:\...

Bước 2: Cài đặt thư viện cần thiết

Chạy lệnh:

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

Trong trường hợp không có file, có thể cài thủ công:

```
pip install transformers streamlit underthesea torch
```

3. Chạy ứng dụng:

Sau khi cài đặt đầy đủ, chạy:

```
streamlit run app.py
```

Khi chạy thành công, trình duyệt sẽ tự mở tại:
<http://localhost:8501>

4. Hướng dẫn sử dụng giao diện:

4.1. Nhập câu tiếng Việt:

- Nhập câu có dấu hoặc không dấu
- Có thể nhập viết tắt:
ví dụ: “*Ban khoe ko?*”, “*Toi rat vui*”
- Tránh nhập ký tự rác / tiếng nước ngoài

4.2. Hệ thống xử lý & chuẩn hoá:

Sau khi bấm **Phân loại**, ứng dụng sẽ hiển thị:

- **Câu gốc** người dùng nhập
- **Câu đã chuẩn hoá** (thêm dấu, sửa viết tắt)
- Nhãn cảm xúc (POSITIVE / NEUTRAL / NEGATIVE)
- Score độ tin cậy
- Output dạng **dictionary** theo yêu cầu đề bài

4.3. Xem lịch sử phân loại:

Mục “Lịch sử phân loại” hỗ trợ:

- Hiển thị **50 bản ghi mới nhất**
- Bấm “**Tải thêm**” để xem nhiều hơn
- Bộ lọc:
 - Tất cả
 - Positive
 - Neutral
 - Negative

5. Xử lý lỗi và thông báo:

Trường hợp	Thông báo
Câu rỗng	“Câu nhập vào đang trống. Vui lòng nhập nội dung.”
Câu quá ngắn	“Câu hơi ngắn, vui lòng nhập câu rõ nghĩa hơn (>= 5 ký tự).”
Câu không giống tiếng Việt	“Câu nhập vào không giống tiếng Việt hoặc không có nghĩa rõ ràng.”
Mô hình lỗi kết nối	“Đã xảy ra lỗi kỹ thuật khi phân loại.”

Tất cả lỗi đều được hiển thị rõ ràng trên giao diện.

6. Ghi chú dành cho người dùng:

- Lần đầu chạy mô hình sẽ mất **10–15 giây** để tải từ HuggingFace.
- Khi chạy lại lần sau, mô hình được lưu và tốc độ sẽ nhanh hơn.
- Không cần kết nối Internet sau khi mô hình đã được cache về máy.
- SQLite không yêu cầu cài đặt — tự động tạo file sentiments.db.

7. Gỡ lỗi nhanh:

Lỗi gặp phải	Nguyên nhân	Cách khắc phục
Không chạy được streamlit	Chưa activate venv	venv\Scripts\activate
Không tải được mô hình	Mất mạng	Kiểm tra Internet
Gõ tiếng Việt không dấu sai	Mapping chưa đủ	Bổ sung từ vào dictionary
Lịch sử không hiển thị	Database trống	Thực hiện 1 lần phân loại

VIII. Kết luận và Hướng phát triển:

1. Kết luận:

Trong khuôn khổ đề án môn Seminar Chuyên Đề, nhóm đã xây dựng thành công ứng dụng **Trợ lý phân loại cảm xúc tiếng Việt** dựa trên mô hình Transformer pre-trained. Ứng dụng đáp ứng đầy đủ các yêu cầu của đề bài:

- Phân loại cảm xúc tiếng Việt vào ba nhãn **POSITIVE**, **NEUTRAL**, **NEGATIVE**
- Xử lý và chuẩn hoá tiếng Việt đầu vào (viết tắt, không dấu, lỗi chính tả nhẹ)
- Kiểm tra câu hợp lệ và loại bỏ các câu vô nghĩa
- Hiển thị kết quả trực quan bằng màu sắc và icon
- Lưu lịch sử phân loại bằng SQLite, hỗ trợ lọc và tải thêm
- Đạt độ chính xác $\geq 65\%$ trên bộ test case theo chuẩn môn học
- Giao diện dễ dùng, gọn nhẹ, chạy ổn định bằng Streamlit

Qua quá trình triển khai, nhóm đã hiểu rõ hơn về:

- Kiến trúc Transformer và pipeline sentiment analysis
- Quy trình xử lý tiếng Việt (NLP preprocessing)
- Tổ chức mã nguồn trong dự án Python
- Kết nối cơ sở dữ liệu SQLite
- Xây dựng giao diện người dùng với Streamlit
- Cách triển khai một ứng dụng NLP hoàn chỉnh từ backend đến frontend

Kết quả cho thấy mô hình hoạt động ổn định, cho dự đoán nhanh, độ chính xác phù hợp với mục tiêu của bài tập và có khả năng áp dụng vào các tình huống thực tế đơn giản.

2. Hạn chế:

Mặc dù ứng dụng hoạt động tốt, một số hạn chế vẫn tồn tại:

1. **Độ chính xác chỉ ở mức cơ bản**, vì không fine-tune mô hình theo domain cụ thể.
2. **Mapping từ không dấu → có dấu** mang tính thủ công, không thể bao phủ mọi trường hợp.
3. **Không xử lý được tiếng lóng phức tạp hoặc câu ẩn dụ**, ví dụ: “đắng lòng”, “tụt mood”, “toang”.
4. **Không hỗ trợ phân tích đa câu** hoặc đoạn văn dài.
5. **Heuristic kiểm tra tiếng Việt còn đơn giản**, có thể sai trong một số trường hợp đặc biệt.

3. Hướng phát triển:

Để hoàn thiện hơn và mở rộng phạm vi ứng dụng, nhóm đề xuất một số hướng phát triển:

1. Fine-tune mô hình trên bộ dữ liệu lớn hơn

- Sử dụng dataset như UIT-VSFC hoặc VLSP
- Tăng độ chính xác đáng kể
- Giảm nhầm lẫn giữa Neutral – Positive – Negative

2. Tích hợp mô hình gán dấu (diacritizer) tự động

- Tự động thêm dấu hoàn chỉnh dựa trên ngữ cảnh
- Không phụ thuộc vào mapping thủ công

3. Xử lý tiếng lóng nâng cao

- Xây dựng wordlist tiếng lóng phổ biến
- Tự động học mapping từ corpus

4. Hỗ trợ đoạn văn và phân tích đa câu

- Nhận biết cảm xúc theo từng câu
- Tính điểm cảm xúc tổng hợp của đoạn

5. Tạo API dịch vụ

- Triển khai bằng **FastAPI** hoặc **Flask**
- Cho phép hệ thống khác gọi API để phân tích cảm xúc

6. Xuất lịch sử thành file CSV hoặc Excel

- Giúp người dùng tổng hợp dữ liệu
- Phân tích thống kê cảm xúc theo thời gian

7. Đưa ứng dụng lên web

- Dùng Streamlit Cloud / HuggingFace Spaces
- Người khác có thể dùng trực tiếp mà không cần cài đặt

4. Nhận xét chung:

Đồ án đã hoàn thành đầy đủ và vượt mức yêu cầu của môn học.

Ứng dụng không chỉ minh họa được khả năng áp dụng mô hình Transformer vào bài toán phân loại cảm xúc, mà còn giúp nhóm tiếp cận quy trình xây dựng một hệ thống NLP hoàn chỉnh: từ tiền xử lý, mô hình hóa, giao diện, đến lưu trữ và đánh giá.

Kết quả đạt được thể hiện sự nỗ lực trong suốt quá trình thực hiện đồ án và là nền tảng vững chắc để phát triển các hệ thống xử lý ngôn ngữ tự nhiên nâng cao hơn trong tương lai.

IX. Tài liệu tham khảo:

- [1] HuggingFace. (2024). *Transformers Documentation*.
<https://huggingface.co/docs/transformers>
- [2] VinAI Research. (2020). *PhoBERT: Pre-trained Language Model for Vietnamese*.
<https://github.com/VinAIResearch/PhoBERT>
- [3] Underthesea Team. (2023). *Underthesea: Vietnamese NLP Toolkit*.
<https://github.com/undertheseanlp/underthesea>
- [4] Streamlit Inc. (2024). *Streamlit Documentation*.
<https://docs.streamlit.io/>

X. Phụ lục:

1. Đường dẫn github của dự án:

Dự án đã được đăng tải trên github, đường dẫn của dự án như sau:

<https://github.com/xuanduy2302/Seminar>

2. Mã nguồn của dự án:

2.1. Mã nguồn file app.py:

```
import sqlite3
```

```
# app.py
```

```
"""
```

Ứng dụng Streamlit:

- Nhận câu tiếng Việt từ người dùng
- Gọi NLP để phân loại cảm xúc
- Lưu lịch sử vào SQLite
- Hiển thị lịch sử phân loại

```
"""
```

```
import streamlit as st
```

```
from db_utils import init_db, save_result, get_history  
from nlp_utils import classify
```

```
# Khởi tạo DB ngay khi run app
```

```
init_db()
```

```
# State cho lịch sử
```

```
if "history_limit" not in st.session_state:
```

```
    st.session_state["history_limit"] = 50 # mặc định 50 bản ghi
```

```
if "history_filter" not in st.session_state:
```

```

st.session_state["history_filter"] = "ALL" # ALL / POSITIVE / NEGATIVE / NEUTRAL

st.set_page_config(
    page_title="Trợ lý phân loại cảm xúc tiếng Việt",
    page_icon="💡",
    layout="centered",
)

st.title("💡 Trợ lý phân loại cảm xúc tiếng Việt")
st.write(
    "Nhập một câu tiếng Việt bất kỳ. Ứng dụng sẽ phân loại cảm xúc thành "
    "**POSITIVE**, **NEUTRAL** hoặc **NEGATIVE**."
)

st.markdown("---")

# Nhập liệu
user_text = st.text_area(
    "Nhập câu tiếng Việt:",
    height=120,
    placeholder="Ví dụ: Hôm nay tôi rất vui vì được 10 điểm...",
)

col1, col2 = st.columns([1, 1])

with col1:
    classify_btn = st.button("Phân loại cảm xúc")

# Kết quả phân loại

```

```

if classify_btn:
    if not user_text or len(user_text.strip()) == 0:
        st.error(" ! Câu nhập vào đang trống. Vui lòng nhập nội dung.")
    elif len(user_text.strip()) < 5:
        st.warning("⚠ Câu hỏi ngắn, vui lòng nhập câu rõ nghĩa hơn (>= 5 ký tự).")
    else:
        with st.spinner("Đang phân tích cảm xúc..."):
            try:
                result = classify(user_text)

                original_text = result["original_text"]
                normalized_text = result["normalized_text"]
                sentiment = result["sentiment"]
                score = result["score"]

                # Lưu vào DB (lưu câu gốc)
                save_result(original_text, sentiment)

                # ===== Hiển thị câu gốc & câu chuẩn hoá =====
                st.write("**Câu gốc:** ", original_text)
                st.write("**Câu chuẩn hoá:** ", normalized_text)

                # ===== Hiển thị cảm xúc theo màu + icon =====
                color_map = {
                    "POSITIVE": ("🟢", "TÍCH CỰC", "✓", "green"),
                    "NEGATIVE": ("🔴", "TIÊU CỰC", "✗", "red"),
                    "NEUTRAL": ("🟡", "TRUNG TÍNH", "❓", "gold"),
                }

```

```

icon, label_vi, symbol, color = color_map.get(
    sentiment, ("○", "KHÔNG RŌ", "?", "gray")
)

st.markdown(
    f"""
<div style='padding:12px; border-radius:8px; border:1px solid {color}; background-
color:#fdfdfd; margin-top:8px;'>
    <h3 style='color:{color}; margin:0;'>{icon} {label_vi} ({symbol})</h3>
    <p style='margin:4px 0;'>Độ tin cậy: <b>{score:.2f}</b></p>
</div>
""",
    unsafe_allow_html=True
)

# (tuỳ chọn) Hiển thị đúng kiểu "dictionary 2 trường" như đê
st.subheader("Đầu ra dạng dictionary:")
st.json({
    "text": normalized_text,
    "sentiment": sentiment
})

except ValueError as e:
    # Lỗi do mình chủ động raise (câu vô nghĩa / không phải tiếng Việt)
    st.error(f"! {e}")

except Exception as e:
    # Lỗi kỹ thuật khác
    st.error(f"Đã xảy ra lỗi kỹ thuật khi phân loại: {e}")

```

```

st.markdown("---")
st.subheader("🕒 Lịch sử phân loại")

# --- Bộ lọc + nút tải thêm ---
col_filter, col_info, col_more = st.columns([2, 1, 1])

with col_filter:
    filter_label = st.selectbox(
        "Lọc theo nhãn:",
        options=["Tất cả", "Positive", "Neutral", "Negative"],
        index=0,
    )

    filter_map = {
        "Tất cả": "ALL",
        "Positive": "POSITIVE",
        "Neutral": "NEUTRAL",
        "Negative": "NEGATIVE",
    }
    st.session_state["history_filter"] = filter_map[filter_label]

# Xác định sentiment filter thật gửi xuống DB
sentiment_filter = (
    None if st.session_state["history_filter"] == "ALL"
    else st.session_state["history_filter"]
)

# Lấy lịch sử từ DB
history = get_history()

```

```

limit=st.session_state["history_limit"],
sentiment=sentiment_filter,
)

def load_more():
    st.session_state["history_limit"] += st.session_state.get("history_increment", 10)

```

```

if not history:
    st.info("Chưa có lịch sử nào khớp với bộ lọc hiện tại.")
else:
    color_map = {
        "POSITIVE": ("🟢", "green", "✓"),
        "NEGATIVE": ("🔴", "red", "✗"),
        "NEUTRAL": ("🟡", "gold", "❓"),
    }

```

```

for item in history:
    text = item["text"]
    sentiment = item["sentiment"]
    timestamp = item["timestamp"]

    icon, color, symbol = color_map.get(sentiment, ("⚪", "gray", "?"))

    st.markdown(
        f"""
        <div style="border:1px solid {color}; padding:10px; border-radius:8px; margin-bottom:8px; background:#fdfdfd;">
            <span style="font-size:18px;">{icon}</span>
        </div>
    """
)

```

```

<b style="color:{color};"> {sentiment} ({symbol})</b><br>
<span style="font-size:14px;"> 📝 {text}</span><br>
<span style="font-size:12px; color:#666;">⌚ {timestamp}</span>
</div>
"""
unsafe_allow_html=True
)

```

```

if len(history) >= st.session_state["history_limit"]:
    if "history_increment" not in st.session_state:
        st.session_state["history_increment"] = 10
    if st.button("Tải thêm", on_click=load_more):
        st.session_state["history_limit"] += st.session_state["history_increment"]

```

2.2. Mã nguồn file nlp_utils.py:

```
# nlp_utils.py
```

```
"""
```

Module xử lý NLP:

- Chuẩn hoá câu tiếng Việt
- Gọi mô hình Transformer để phân loại cảm xúc

```
"""
```

```
from functools import lru_cache
```

```
import re
```

```
from underthesea import word_tokenize
```

```
from transformers import pipeline
```

```
# Mô hình sentiment tiếng Việt trên HuggingFace
```

MODEL_NAME = "5CD-AI/vietnamese-sentiment-visobert"

```
LABEL_MAPPING = {  
    "NEG": "NEGATIVE",  
    "POS": "POSITIVE",  
    "NEU": "NEUTRAL",  
}
```

```
@lru_cache(maxsize=1)
def get_sentiment_pipeline():
    """
    Chỉ load model 1 lần duy nhất (dùng cache) để tránh bị chậm.
    """
    sentiment_pipeline = pipeline(
        task="sentiment-analysis",
        model=MODEL_NAME,
        tokenizer=MODEL_NAME,
    )
    return sentiment_pipeline
```

VIET_VOWELS = set(
"aeiouyAEIOUY"
"ääêöouĂÂÊӮOOU"
"ääääääÁÀÅÃÃ"
"ääääääÁÀÅÃÃ"
"ääääääÁÀÅÃÃ"
"éèéëëéÉÈËËÉ"
"óòööööÓÒÖÖÖÖ"
"óòöööööÓÒÖÖÖÖ"
"óòöööööÓÒÖÖÖÖ")

```

"îîññÎÎÏÏ"
"úùûûûÛÛÛÛÛÛ"
"ýÿýýÝÝÝÝÝÝ"
"đĐ"
)

VIET_STOPWORDS = {
    "là", "và", "của", "không", "rất", "này", "kia", "đó", "này",
    "tôi", "ban", "bạn", "mình", "anh", "em", "chỉ", "thì",
    "nhưng", "nếu", "vì", "nên", "cho", "khi", "đã", "đang", "sẽ",
    "ở", "trong", "trên", "với", "hay", "cũng", "rồi", "luôn",
}

```

```
def is_valid_vietnamese(text: str) -> bool:
```

```
"""
```

Heuristic đơn giản:

- Có ít nhất 2 từ
- Phần lớn kí tự là chữ cái, có đủ nguyên âm tiếng Việt
- Có ít nhất 1 stopword Việt phô biến
- Không toàn là kí tự ngẫu nhiên / số / ký hiệu

```
"""
```

```
if not isinstance(text, str):
```

```
    return False
```

```
text = text.strip()
```

```
if len(text) < 3:
```

```
    return False
```

```
# Lấy các "từ" là cụm chữ cái
```

```
words = re.findall(r"[A-Za-zÀ-Ñà-ñ]+", text)
```

```

if len(words) < 2:
    return False

letters = "".join(words)
if not letters:
    return False

# Tỉ lệ nguyên âm
vowel_count = sum(1 for ch in letters if ch in VIET_VOWELS)
ratio_vowel = vowel_count / len(letters)

# Nếu quá ít nguyên âm → thường là random / không phải tiếng Việt
if ratio_vowel < 0.25:
    return False

# Nếu có ít nhất 1 stopword Việt → ưu tiên coi là hợp lệ
lower_words = [w.lower() for w in words]
if any(w in VIET_STOPWORDS for w in lower_words):
    return True

# Trung bình độ dài từ nếu quá dài → dễ là chuỗi vô nghĩa
avg_len = sum(len(w) for w in words) / len(words)
if avg_len > 10:
    return False

# Mặc định: tạm coi là hợp lệ
return True

def normalize_text(text: str) -> str:
    """

```

Chuẩn hoá câu để HIỂN THỊ cho người dùng.

Ví dụ: 'Ban khoe ko?' -> 'Bạn khỏe không?'

(Chỉ làm 1 số luật đơn giản cho demo, không phải phục hồi dấu 100%)

```
if not isinstance(text, str):
```

```
    text = str(text)
```

```
text = text.strip().lower()
```

```
# Tách token (giữ cả dấu ? ! , .)
```

```
tokens = re.findall(r"\w+|[^\w\s]", text, flags=re.UNICODE)
```

```
# Một số mapping cơ bản
```

```
mapping = {
```

```
    # Đại từ – Ngôi xung
```

```
    "toi": "tôi",
```

```
    "ban": "bạn",
```

```
    "minh": "mình",
```

```
    "anh": "anh",
```

```
    "chi": "chị",
```

```
    "em": "em",
```

```
    "co": "cô",
```

```
    "chu": "chú",
```

```
    "ba": "bà",
```

```
    "ong": "ông",
```

```
    "nguo": "người",
```

```
    "ho": "họ",
```

```
# Động từ – tính từ phổ biến
```

```
    "yeu": "yêu",
```

"thuong": "thương",

"ghet": "ghét",

"thich": "thích",

"biet": "biết",

"hieu": "hiểu",

"thay": "thấy",

"khoe": "khoe",

"om": "ôm",

"dau": "đau",

"met": "mệt",

"vui": "vui",

"buon": "buồn",

"gian": "giận",

"nong": "nóng",

"lanh": "lạnh",

"dep": "đẹp",

"xau": "xấu",

Từ phủ định – viết tắt

"khong": "không",

"k": "không",

"ko": "không",

"k0": "không",

"hok": "không",

"khg": "không",

"hk": "không",

"kh": "không",

Câu hỏi

"gi": "gì",

"j": "gi",
"sao": "sao",
"tai": "tại",
"vi": "vì",
"tai sao": "tại sao",

Các trạng từ

"rat": "rất",
"hon": "hơn",
"lam": "lắm",
"qua": "quá",
"nhieu": "nhiều",
"it": "ít",
"noi": "nói",
"noi chuyen": "nói chuyện",

Địa điểm – thời gian

"nay": "nay",
"mai": "mai",
"hom": "hôm",
"truoc": "trước",
"sau": "sau",
"o": "ó",

Từ liên kết

"va": "và",
"voi": "với",
"vi": "vì",
"nen": "nên",
"nhung": "nhưng",

Các từ cơ bản

"duoc": "được",

"dc": "được",

"du": "đủ",

"thoi": "thôi",

"roi": "rồi",

"cung": "cũng",

"luon": "luôn",

"neu": "nếu",

"dang": "đang",

"se": "sẽ",

"da": "đã",

Danh từ

"con": "con",

"nguo": "người",

"ban be": "bạn bè",

"gia dinh": "gia đình",

"cong viec": "công việc",

"truong": "trường",

"lop": "lớp",

"mon": "món",

"an": "ăn",

"quan": "quán",

"nha": "nhà",

"cua": "cửa",

Cảm xúc – đánh giá

"te": "tệ",

"tot": "tốt",
"hay": "hay",
"do": "dở",
"chap nhan": "chấp nhận",
"tuyet": "tuyệt",

Chat / internet slang

"ad": "admin",
"ib": "nhắn",
"rep": "trả lời",
"like": "thích",
"sub": "đăng ký",
"vid": "video",

Từ có dấu phẩy bị gõ sai TELEX

"thuc": "thực",
"phai": "phải",
"thuan": "thuận",
"mien": "miền",
"quoc": "quốc",
"dong": "đông",
"tay": "tây",
"nam": "năm",
"troi": "trời",
"muon": "muốn",

Từ nối dài

"cam on": "cảm ơn",
"xin loi": "xin lỗi",
"tam biet": "tạm biệt",

```

"chao": "chào",
# Thường gặp khi không sử dụng dấu
"kha": "khá",
"de": "dẽ",
"kho": "khó",
"to": "to",
"nho": "nhỏ",
"lon": "lớn",
"nhe": "nhẹ",
"man": "mặn",
"ngot": "ngọt",
}

```

```

new_tokens = []
for tok in tokens:
    if tok.isalpha(): # chỉ map với chữ cái
        mapped = mapping.get(tok, tok)
        new_tokens.append(mapped)
    else:
        new_tokens.append(tok)

# Ghép lại, xử lý khoảng trắng trước dấu câu
sentence = ""
for tok in new_tokens:
    if tok in [".", ",", ":", ";", "?", "!", "..."]:
        sentence = sentence.rstrip() + tok + " "
    else:
        sentence += tok + " "
sentence = sentence.strip()

```

```
# Viết hoa chữ cái đầu
if sentence:
    sentence = sentence[0].upper() + sentence[1:]

return sentence
```

```
def preprocess(text: str) -> str:
```

```
"""
```

Chuẩn hoá câu để đưa vào mô hình Transformer:

- đưa về chữ thường
- thay viết tắt cơ bản
- tách từ bằng underthesea

```
"""
```

```
if not isinstance(text, str):
```

```
    text = str(text)
```

```
text = text.strip().lower()
```

```
replacements = {
```

```
    "ko": "không",
```

```
    "kô": "không",
```

```
    "khong": "không",
```

```
    "hok": "không",
```

```
    "k": "không",
```

```
    "k0": "không",
```

```
    "zui": "vui",
```

```
    "sz": "size",
```

```
    "dc": "được",
```

```
    "đc": "được",
```

```
    "vs": "với",
```

```

    "j": "gi",
    "0": "không",
    "nma": "nhưng mà",
    "nhg": "nhưng",
    "mn": "mọi người",
    "hk": "không",
    "thik": "thích",
    "hoy": "không",
    "hoi": "không",
}

for k, v in replacements.items():
    text = text.replace(f" {k} ", f" {v} ")

tokens = word_tokenize(text, format="text")
return tokens

def classify(text: str) -> dict:
    """
    Phân loại cảm xúc cho 1 câu tiếng Việt.

    Trả về dict:
    {
        "original_text": ...,
        "normalized_text": ...,
        "sentiment": ...,
        "score": ...
    }
    """

if not text or not text.strip():
    raise ValueError("Câu nhập vào rỗng.")
if not is_valid_vietnamese(text):

```

```

raise ValueError("Câu nhập vào không giống tiếng Việt hoặc không có nghĩa rõ ràng.")

# Câu chuẩn hoá để hiển thị cho người dùng
normalized = normalize_text(text)

# Câu chuẩn hoá cho mô hình
cleaned = preprocess(text)
sentiment_pipeline = get_sentiment_pipeline()

result = sentiment_pipeline(cleaned)[0]
raw_label = result.get("label", "").upper()

sentiment = LABEL_MAPPING.get(raw_label, "NEUTRAL")
score = float(result.get("score", 0.0))

return {
    "original_text": text,
    "normalized_text": normalized,
    "sentiment": sentiment,
    "score": score,
}

```

2.3. Mã nguồn file db_utils.py:

```
import sqlite3
```

```
# db_utils.py
```

```
""""
```

Module làm việc với SQLite:

- Khởi tạo database
- Lưu lịch sử phân loại

- Lấy danh sách lịch sử gần nhất

"""

```
import os
import sqlite3
from datetime import datetime
from typing import List, Dict, Any
```

Đường dẫn tới file DB nằm trong thư mục db/

```
BASE_DIR = os.path.dirname(os.path.abspath(__file__))
```

```
DB_PATH = os.path.join(BASE_DIR, "db", "sentiments.db")
```

```
def get_connection():
```

"""

Tạo connection tới SQLite.

check_same_thread=False để dùng được trong Streamlit (nhiều thread).

"""

```
conn = sqlite3.connect(DB_PATH, check_same_thread=False)
```

```
return conn
```

```
def init_db():
```

"""

Tạo bảng nếu chưa có.

Bảng: sentiments(id, text, sentiment, timestamp)

"""

```
os.makedirs(os.path.join(BASE_DIR, "db"), exist_ok=True)
```

```
conn = get_connection()
```

```
cursor = conn.cursor()
```

```

cursor.execute(
    """
CREATE TABLE IF NOT EXISTS sentiments (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    text TEXT NOT NULL,
    sentiment TEXT NOT NULL,
    timestamp TEXT NOT NULL
);
    """

)

conn.commit()
conn.close()

def save_result(text: str, sentiment: str) -> None:
    """
    Lưu 1 bản ghi cảm xúc vào DB.
    """
    conn = get_connection()
    cursor = conn.cursor()

    timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")

    cursor.execute(
        "INSERT INTO sentiments (text, sentiment, timestamp) VALUES (?, ?, ?);",
        (text, sentiment, timestamp),
    )

    conn.commit()
    conn.close()

```

```

def get_history(limit: int = 50, sentiment: str | None = None) -> List[Dict[str, Any]]:
    """
    Lấy danh sách lịch sử mới nhất.
    - limit: số bản ghi tối đa
    - sentiment: lọc theo POSITIVE / NEGATIVE / NEUTRAL, hoặc None nếu lấy tất cả
    """

    conn = get_connection()
    cursor = conn.cursor()

    query = """
        SELECT text, sentiment, timestamp
        FROM sentiments
    """

    params = []

    if sentiment:
        query += " WHERE sentiment = ?"
        params.append(sentiment)

    query += " ORDER BY id DESC LIMIT ?;"
    params.append(limit)

    cursor.execute(query, tuple(params))
    rows = cursor.fetchall()
    conn.close()

    history = []
    for text, sentiment, timestamp in rows:
        history.append(

```

```
{  
    "text": text,  
    "sentiment": sentiment,  
    "timestamp": timestamp,  
}  
)
```

return history