ĐỒ ÁN MÔN HỌC

SEMINAR CHUYÊN ĐỀ

XÂY DỰNG TRỢ LÝ PHÂN LOẠI CẢM XÚC TIẾNG VIỆT SỬ DỤNG TRANSFORMER

I. THÔNG TIN CHUNG

Mục	Nội dung
Tên đồ án	Trợ lý phân loại cảm xúc tiếng Việt (Vietnamese Sentiment Assistant) sử dụng Transformer
Mục đích	Phát triển ứng dụng phân loại cảm xúc (tích cực, trung tính, tiêu cực) từ văn bản tiếng Việt, sử dụng Transformer
Số lượng thành viên	1 – 2 sinh viên
Thời gian thực hiện	06/12/2025
Ngôn ngữ lập trình	Python
Thư viện chính	Hugging Face Transformers (`phobert-base-v2` hoặc `distilbert-base-multilingual-cased`), Underthesea (tùy chọn)
Công cụ giao diện	Không giới hạn (Streamlit, Tkinter, Flask…)
Yêu cầu bắt buộc	Ứng dụng chạy độc lập, phân loại cảm xúc tiếng Việt, lưu kết quả cục bộ

II. MỤC TIÊU ĐỒ ÁN

1. Xây dựng ứng dụng **phân loại cảm xúc** đơn giản, nhận câu tiếng Việt và trả về nhãn cảm xúc (tích cực, trung tính, tiêu cực).
2. Tích hợp **Transformer pre-trained** (PhoBERT hoặc DistilBERT) qua pipeline `sentiment-analysis` để phân loại, không cần fine-tuning.
3. Lưu trữ lịch sử phân loại cục bộ bằng SQLite.
4. Đảm bảo **độ chính xác phân loại ≥ 65%** trên 10 test case tiếng Việt.
5. Trình bày kết quả qua **báo cáo đồ án**.

III. YÊU CẦU KỸ THUẬT

1. Chức năng bắt buộc

Chức năng	Mô tả chi tiết
Nhập liệu ngôn ngữ tự nhiên	Người dùng nhập câu tiếng Việt tự do (ví dụ: "Hôm nay tôi rất vui" hoặc "Món ăn này dở quá")
Phân loại cảm xúc (NLP)	Sử dụng Transformer pre-trained để phân loại: • `POSITIVE` (tích cực) • `NEUTRAL` (trung tính) • `NEGATIVE` (tiêu cực)
Lưu trữ cục bộ	Lưu lịch sử phân loại (câu, nhãn cảm xúc, thời gian)
Hiển thị kết quả	Hiển thị nhãn cảm xúc và danh sách lịch sử phân loại

2. Yêu cầu về xử lý tiếng Việt

- **Đầu vào**: Câu tiếng Việt, có thể viết tắt, thiếu dấu.
- **Đầu ra**: Dictionary chứa 2 trường:

```
{
  "text": "Hôm nay tôi rất vui",
  "sentiment": "POSITIVE"
}
```

- **Yêu cầu xử lý**:
  - Phân loại đúng 3 nhãn cảm xúc: `POSITIVE`, `NEUTRAL`, `NEGATIVE`.
  - Hiểu các biến thể tiếng Việt (viết tắt, thiếu dấu).
  - Độ chính xác phân loại: ≥ 65% trên 10 test case.

3. Giao diện người dùng

- **Yêu cầu tối thiểu**:
  - Ô nhập văn bản tự do.
  - Nút "Phân loại cảm xúc" (gửi câu qua pipeline Transformer).
  - Hiển thị nhãn cảm xúc (VD: "Tích cực").
  - Danh sách lịch sử phân loại (bảng hoặc list).
  - Thông báo pop-up nếu nhập lỗi (VD: "Câu quá ngắn!").

IV. SẢN PHẨM NỘP (DELIVERABLES)

STT	Sản phẩm	Định dạng	Yêu cầu
1	**Ứng dụng chạy được**	`.exe` / Web / Python script	Chạy độc lập, không lỗi
2	**Mã nguồn**	Trình bày trong phụ lục của đồ án	Có `README.md`, cấu trúc rõ ràng
3	**Báo cáo đồ án**	PDF	Theo mẫu tiểu luận
4	**Video demo**	MP4 (1–2 phút)	Quay màn hình, có âm thanh
5	**Bộ test case**	Trình bày trong phụ lục của báo cáo đồ án	10 câu tiếng Việt + kết quả mong đợi

---

V. BÁO CÁO ĐỒ ÁN (CẤU TRÚC BẮT BUỘC)

1. Giới thiệu & Mục tiêu
2. Phân tích yêu cầu
3. Thiết kế hệ thống (Sơ đồ khối, Flowchart)
4. Giải pháp (Mô tả cách dùng Transformer)
5. Triển khai & Kết quả
6. Đánh giá hiệu suất (Bảng test 10 câu, độ chính xác)
7. Hướng dẫn cài đặt & sử dụng
8. Kết luận & Hướng phát triển

VI. RUBRICS CHẤM ĐIỂM (THANG ĐIỂM 10)

Tiêu chí	Mô tả chi tiết	Điểm
1. Ứng dụng chạy ổn định & Giao diện (3.0đ)	• Ứng dụng khởi động nhanh, không crash (1.25đ) • Giao diện rõ ràng, dễ dùng (1.0đ) • Hiển thị nhãn và lịch sử phân loại (0.75đ)	**3.0**
2. Tích hợp NLP hiệu quả (3.0đ)	• Phân loại cảm xúc đúng ≥ 65% trên 10 test case (2.0đ) • Xử lý biến thể tiếng Việt (0.75đ) • Phản hồi nhanh qua pipeline (0.25đ)	**3.0**
3. Xử lý ngôn ngữ tự nhiên tiếng Việt (2.0đ)	• Hiểu câu viết tắt, thiếu dấu (1.0đ) • Xử lý lỗi nhập liệu (0.5đ) • Phản hồi tự nhiên qua giao diện (0.5đ)	**2.0**
4. Lưu trữ lịch sử phân loại (1.5đ)	• Lưu trữ SQLite chính xác (1.0đ) • Hiển thị lịch sử phân loại (0.5đ)	**1.5**
5. Báo cáo, mã nguồn, demo (0.5đ)	• Báo cáo khoa học, đầy đủ (0.25đ) • Mã nguồn sạch, có README; video demo rõ ràng (0.25đ)	**0.5**

VII. HƯỚNG DẪN TRIỂN KHAI

1. Giải pháp NLP

Kiến trúc sử dụng **Transformer pre-trained** (`phobert-base-v2` hoặc `distilbert-base-multilingual-cased`) qua pipeline `sentiment-analysis` của Hugging Face. Không cần fine-tuning, tập trung vào tích hợp đơn giản.

- **Kiến trúc tổng quát** (Sơ đồ khối – sinh viên vẽ flowchart trong báo cáo):

```
[Đầu vào: Câu tiếng Việt]
    ↓ (Preprocessing)
[Component 1: Tiền xử lý] → Câu đã chuẩn hóa
    ↓ (Sentiment Analysis)
[Component 2: Phân loại cảm xúc] → Nhãn (POSITIVE, NEUTRAL, NEGATIVE)
    ↓ (Validation)
[Component 3: Hợp nhất & xử lý lỗi] → Đầu ra dictionary hoặc lỗi
    ↓
[Core Engine: Lưu & hiển thị]
```

Hướng dẫn chi tiết và tối ưu hóa vấn đề kỹ thuật

Bước	Mục đích	Hướng dẫn thực hiện
1. **Tiền xử lý**	Chuẩn hóa câu tiếng Việt để phù hợp với Transformer.	• (Optional) Dùng `underthesea.word_tokenize()` để tách từ (VD: "Rat vui hom nay" → "Rất vui hôm nay"). • Chuyển chữ thường, thay "rat" → "rất" bằng từ điển nhỏ (10–20 từ phổ biến: rat/rất, dở/do, v.v.). • Giới hạn câu ≤50 ký tự để giảm thời gian xử lý.
2. **Phân loại cảm xúc**	Xác định nhãn cảm xúc (POSITIVE, NEUTRAL, NEGATIVE).	• Sử dụng pipeline `sentiment-analysis` với model `phobert-base-v2` (ưu tiên tiếng Việt) hoặc `distilbert-base-multilingual-cased` (hỗ trợ đa ngôn ngữ). • Gửi câu chuẩn hóa qua pipeline, lấy nhãn có xác suất cao nhất. • Nếu xác suất <0.5, trả về NEUTRAL mặc định.
3. **Hợp nhất & xử lý lỗi**	Ghép kết quả thành dictionary, kiểm tra hợp lệ.	• Tạo dictionary: `{text, sentiment}`. • Kiểm tra: Câu nhập ≥5 ký tự; nếu rỗng hoặc lỗi pipeline, hiển thị pop-up "Câu không hợp lệ, thử lại". • Lưu vào SQLite và hiển thị trên giao diện.

2. Lưu trữ & Hiển thị

- **Lưu trữ**: Dùng `sqlite3`, tạo bảng `sentiments` (cột: id, text, sentiment, timestamp). Lưu timestamp dưới dạng ISO string (`YYYY-MM-DD HH:MM:SS`).
  - **Vấn đề kỹ thuật**: SQL injection nếu không sanitize input.
  - **Giải pháp**: Dùng parameterized queries (`cursor.execute("INSERT INTO sentiments VALUES (?, ?, ?)", (id, text, sentiment))`).
- **Hiển thị**: Hiển thị danh sách lịch sử phân loại (giới hạn 50 bản ghi mới nhất, dùng `SELECT * FROM sentiments ORDER BY timestamp DESC LIMIT 50`).
  - **Vấn đề kỹ thuật**: Danh sách dài làm chậm giao diện.
  - **Giải pháp**: Giới hạn hiển thị, thêm nút "Tải thêm" nếu cần.

3. Giao diện

- **Khuyến khích**: Streamlit vì dễ tích hợp, hỗ trợ asynchronous (`st.spinner` khi gọi pipeline).
- **Vấn đề kỹ thuật**: Treo UI khi gọi pipeline trên Tkinter.
- **Giải pháp**: Nếu dùng Tkinter, gọi pipeline trong thread riêng (`import threading`). Cache pipeline để tái sử dụng.
- **Mẹo**: Test giao diện với câu đơn giản (VD: "Tôi vui") trước khi tích hợp đầy đủ.

4. Đánh giá & Test Case

- **Vấn đề kỹ thuật**: Độ chính xác thấp nếu model không hiểu tiếng Việt.
- **Giải pháp**: Dùng `phobert-base-v2` (tối ưu tiếng Việt); nếu không đạt ≥65%, thử `distilbert-base-multilingual-cased`. Test riêng pipeline với 5 câu đầu, điều chỉnh từ điển chuẩn hóa nếu cần.
- **Báo cáo**: Thêm bảng so sánh kết quả thực tế vs mong đợi, phân tích lỗi (VD: "Câu viết tắt gây nhầm → cải thiện bằng từ điển").

VIII. BỘ TEST CASE (10 CÂU)

STT	Đầu vào (Câu tiếng Việt)	Đầu ra mong đợi (Dictionary)
1	Hôm nay tôi rất vui	{"text": "Hôm nay tôi rất vui", "sentiment": "POSITIVE"}
2	Món ăn này dở quá	{"text": "Món ăn này dở quá", "sentiment": "NEGATIVE"}
3	Thời tiết bình thường	{"text": "Thời tiết bình thường", "sentiment": "NEUTRAL"}
4	Rat vui hom nay	{"text": "Rất vui hôm nay", "sentiment": "POSITIVE"}
5	Công việc ổn định	{"text": "Công việc ổn định", "sentiment": "NEUTRAL"}

STT	Đầu vào (Câu tiếng Việt)	Đầu ra mong đợi (Dictionary)
6	Phim này hay lắm	{"text": "Phim này hay lắm", "sentiment": "POSITIVE"}
7	Tôi buồn vì thất bại	{"text": "Tôi buồn vì thất bại", "sentiment": "NEGATIVE"}
8	Ngày mai đi học	{"text": "Ngày mai đi học", "sentiment": "NEUTRAL"}
9	Cảm ơn bạn rất nhiều	{"text": "Cảm ơn bạn rất nhiều", "sentiment": "POSITIVE"}
10	Mệt mỏi quá hôm nay	{"text": "Mệt mỏi quá hôm nay", "sentiment": "NEGATIVE"}

IX. TÀI LIỆU THAM KHẢO

1. Hugging Face Transformers
2. VinAI PhoBERT
3. Underthesea Documentation
4. Streamlit Documentation

Lưu ý:

- Không sao chép mã nguồn từ bất kỳ nguồn nào.
- Được dùng pipeline `sentiment-analysis` để đơn giản hóa.