



Group 17

SENTIMENT ANALYSIS

Đồng Quốc ĐẠI - 23001513
Nguyễn Mạnh Dũng - 23001507
Chu Thành Dũng - 23001506

🔍 Sentiment Analysis 💻

01/27





INTRODUCTION

SENTIMENT ANALYSIS

Sentiment Analysis - tức phân tích cảm xúc - là quá trình tự động sử dụng các kỹ thuật xử lý ngôn ngữ tự nhiên (NLP), học máy (Machine Learning) hoặc học sâu (Deep Learning) để nhận diện và phân loại cảm xúc/thái độ biểu lộ.

Một trong các bài toán phổ biến nhất của Sentiment Analysis là phân loại polarity: gán nhãn cho văn bản theo các lớp cảm xúc — thường là tích cực (positive), tiêu cực (negative), hoặc trung lập (neutral). Với nhiều bài toán hoặc dữ liệu (ví dụ đánh giá 1–5 sao), có thể mở rộng thành phân tích đa cấp (fine-grained), như "rất tiêu cực – tiêu cực – trung tính – tích cực – rất tích cực" để thể hiện sắc thái cảm xúc rõ hơn.



Q Sentiment Analysis

0

»»



Mục tiêu cốt lõi của Sentiment Analysis là giúp hệ thống hiểu và định lượng các ý kiến, cảm xúc, thái độ chủ quan của con người - tích cực, tiêu cực hay trung lập - thay vì phải đọc thủ công từng bình luận, đánh giá.

Nhờ vậy, công nghệ này cho phép xử lý tự động khối lượng lớn văn bản (reviews, comment, phản hồi khách hàng, mạng xã hội...) - việc mà con người thủ công sẽ rất tốn công, lâu và khó đảm bảo nhất quán.

MỤC TIÊU CỐT LÕI



Sentiment Analysis





CÁC DẠNG PHÂN TÍCH CẢM XÚC



🔍 Sentiment Analysis ➔

- Phân tích Polarity (Polarity Classification): Gán nhãn Tích cực/Tiêu cực/Trung lập (Là trọng tâm của đề tài này)
- Phân tích đa cấp (Fine-grained): Mở rộng thành 5 mức độ (Rất tiêu cực → Rất tích cực)
- Phát hiện cảm xúc (Emotion Detection): Nhận diện cảm xúc cụ thể (Vui vẻ, Tức giận, Thất vọng,...)
- Phân tích theo khía cạnh (Aspect-Based Sentiment Analysis - ABSA): Phân tích cảm xúc theo từng khía cạnh cụ thể (Ví dụ: Khen thiết kế nhưng chê pin)





THÁCH THỨC & HẠN CHẾ

Khó xác định cảm xúc trung tính:
Các bình luận mơ hồ dễ bị phân loại sai.

Không hiểu được Ngữ cảnh, Châm biếm và Mỉa mai: Mô hình dễ bị nhầm lẫn khi có từ ngữ tích cực trong ngữ cảnh tiêu cực (Ví dụ: "Tuyệt vời, tôi lại bị phạt 1 triệu nữa").

Xử lý ngôn ngữ phức tạp: Từ viết tắt, lỗi chính tả, từ lóng trên mạng xã hội liên tục thay đổi.

Ngôn ngữ có nhiều nghĩa: Một từ có thể có ý nghĩa khác nhau tùy ngữ cảnh.

🔍 Sentiment Analysis ◉ ⓘ

»»



CHIẾN LƯỢC HUẤN LUYỆN

Trước hết, nhóm thực hiện fine-tuning toàn bộ mô hình PhoBERT, tức là cho phép cập nhật trọng số của cả phần encoder và lớp phân loại. Việc fine-tuning toàn phần giúp mô hình thích nghi tốt hơn với đặc trưng ngôn ngữ của dữ liệu thương mại điện tử, vốn chứa nhiều từ lóng, cách viết không chuẩn và lỗi chính tả.



Kỹ thuật tối ưu bộ nhớ:

- Gradient Accumulation: Sử dụng Batch size nhỏ (4) nhưng mô phỏng được Batch size lớn (27) bằng cách tích lũy gradient qua 4 bước → Tránh lỗi Out-of-Memory (OOM).
- Mixed Precision (FP27): Chạy tính toán 27-bit thay vì 32-bit → Giảm 50% RAM GPU và tăng tốc độ huấn luyện.

Siêu tham số: Learning rate thấp ($1e-5$) để bảo toàn trọng số tiền huấn luyện





Group 17

CHUẨN BỊ MÔI TRƯỜNG

07/27

- Tiến hành cài đặt toàn bộ thư viện cần thiết để đảm bảo mô hình có thể chạy trên GPU Colab.
- Triển khai mô hình PhoBERT Large đòi hỏi môi trường tính toán đầy đủ và đồng nhất. Do đó, trong Khối 1 yêu cầu tiến hành cài đặt toàn bộ thư viện cần thiết bao gồm:
 - Transformers: cung cấp tokenizer, mô hình PhoBERT và Trainer.
 - Datasets: hỗ trợ xây dựng Dataset chuẩn HuggingFace.
 - Scikit-learn: tính toán F1-score, accuracy và confusion matrix.
 - PyTorch: nền tảng chính để huấn luyện mô hình.

```
!pip install transformers datasets accelerate underthesea  
scikit-learn  
!pip install torch torchvision torchaudio
```

🔍 Sentiment Analysis ➔





IMPORT THƯ VIỆN VÀ KIỂM TRA GPU



Khối 2 có nhiệm vụ tài toàn bộ thư viện cần thiết cho dự án phân tích cảm xúc bằng mô hình PhoBERT Large.

Các thư viện cơ bản của Python được khai báo:

- import os: Dùng để thao tác với hệ thống file.
- import torch: Thư viện PyTorch dùng để xây dựng mô hình mạng neural.
- import pandas as pd: Dùng để xử lý dữ liệu bảng.
- import numpy as np: Xử lý mảng số học, hỗ trợ cho nhiều phép toán của mô hình.
- import matplotlib.pyplot as plt - import seaborn as sns: Hai thư viện dùng để vẽ biểu đồ.
- Thư viện phục vụ MACHINE LEARNING:
 - from sklearn.model_selection import train_test_split: Chia dữ liệu thành train/test theo tỷ lệ 80-20.
 - from sklearn.metrics import accuracy_score, f1_score, classification_report, confusion_matrix: Dùng để đánh giá mô hình: Accuracy, F1-macro, Bảng phân loại, Ma trận nhầm lẫn.
 - from underthesea import word_tokenize: Thư viện tách từ tiếng Việt.

Thư viện của Hugging Face Transformers:

- from transformers import AutoTokenizer: Dùng để mã hóa câu thành token theo chuẩn PhoBERT.
- from transformers import AutoModelForSequenceClassification: Tài mô hình PhoBERT Large đã tối ưu cho phân loại 3 lớp: Negative, Neutral, Positive.
- from transformers import TrainingArguments, Trainer: Dùng để huấn luyện mô hình: Cấu hình batch size, Số epoch, Gradient Accumulation, FP27, Lưu checkpoint, Tự động đánh giá.
- from datasets import Dataset: Chuyển DataFrame → Dataset của Hugging Face.

```
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print(f"🔥 Đang chạy trên thiết bị: {device}")
```





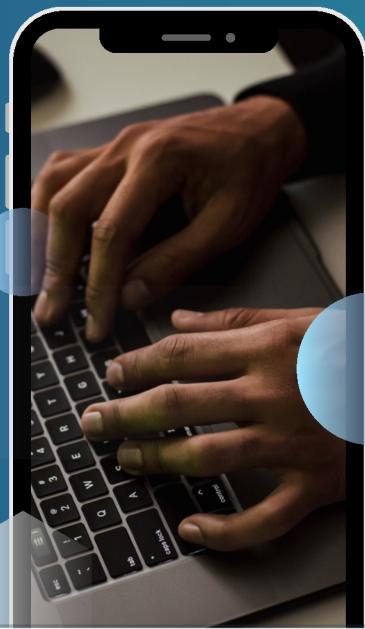
Group 17

09/27

DỮ LIỆU TIỀN XỬ LÝ NLP

- Nguồn: Hơn 140.000 bình luận sản phẩm thu thập từ Tiki.
- Ánh xạ nhãn: Chuyển đổi điểm 1-5 sao thành 3 lớp cảm xúc:
 - 1-2 sao: Tiêu cực
 - 3 sao: Trung lập
 - 4-5 sao: Tích cực

🔍 Sentiment Analysis ➔





TIỀN XỬ LÝ NLP

Chuẩn hóa Unicode:

- Tiếng Việt có nhiều bảng mã khác nhau, nếu không chuẩn hóa sẽ dẫn đến lỗi tách từ hoặc lỗi ký tự.
- Việc chuyển toàn bộ văn bản về dạng Unicode NFC giúp tiêu chuẩn hóa dữ liệu, đảm bảo mô hình hiểu chính xác từng ký tự tiếng

Làm sạch văn bản:

- Một số thao tác làm sạch được thực hiện:
 - Xóa ký tự đặc biệt, emoji, icon
 - Chuẩn hóa toàn bộ chữ viết về dạng lowercase
 - Loại bỏ khoảng trắng dư
 - Loại bỏ các dòng trống hoặc bình luận không hợp lệ

Sentiment Analysis

Tiếng Việt không có ranh giới từ rõ ràng, vì dấu cách chỉ phân tách âm tiết, không phải từ. Ví dụ: "giao hàng nhanh" → có thể hiểu sai nếu không tách đúng.

Để tái sử dụng thư viện Underthesea để thực hiện tách từ. Việc này mang lại nhiều lợi ích:

- Giảm nhầm lẫn khi xử lý ngữ nghĩa
- Làm cho văn bản khớp với định dạng dữ liệu gốc mà PhoBERT được huấn luyện
- Giúp mô hình hiểu rõ từng từ ghép, từ đa âm tiết

Kết quả tách từ được viết dạng:
"giao hàng nhanh" → "giao_hàng_nhanh"





TIỀN XỬ LÝ NLP

```
file_path = '/content/comments.csv'

# 1. Đọc file
try:
    df = pd.read_csv(file_path, encoding='utf-8')
except FileNotFoundError:
    print("⚠️ Lỗi: Không tìm thấy file 'comments.csv'. Hãy upload file lên Colab!")
    raise
except:
    df = pd.read_csv(file_path, encoding='utf-8-sig')

Làm sạch dữ liệu:
• Chỉ giữ lại hai cột quan trọng: rating và content.
• Chuẩn hóa rating về dạng số.
• Loại bỏ toàn bộ dòng không hợp lệ hoặc thiếu dữ liệu.
```

2. Lọc dữ liệu

```
df = df[['rating', 'content']].dropna()
df['rating'] = pd.to_numeric(df['rating'], errors='coerce')
df.dropna(subset=['rating'], inplace=True)
```

```
def map_label(rating):
    if rating in [4, 5]: return 2 # Positive
    if rating == 3: return 1 # Neutral
    if rating in [1, 2]: return 0 # Negative

df['label'] = df['rating'].apply(map_label)
df = df.dropna(subset=['label'])
df['label'] = df['label'].astype(int)

Tách từ tiếng Việt bằng Underthesen. PhoBERT yêu cầu văn bản phải được word segmentation (tách từ và nối bằng dấu gạch dưới):
print("⚠️ Đang tách từ (Word Segmentation)...")
# Hàm này nối từ ghép bằng dấu gạch dưới _ (VD: giao_hàng)
df['content_seg'] = df['content'].apply(lambda x: word_tokenize(str(x), format="text"))

Chia tập dữ liệu Train/Test:
train_df, test_df = train_test_split(df, test_size=0.2, random_state=42, stratify=df['label'])
```

6. Chuyển sang định dạng Dataset của HuggingFace

```
train_dataset = Dataset.from_pandas(train_df[['content_seg', 'label']])
test_dataset = Dataset.from_pandas(test_df[['content_seg', 'label']])

print("✅ Đã xử lý xong dữ liệu!")
print(f"Mẫu tách từ: {train_df['content_seg'].iloc[0]}")
```





Group 17

- Sử dụng PhoBERT Tokenizer với cơ chế BPE (Byte-Pair Encoding).



- Mục đích: Chuyển văn bản đã tách từ thành Token ID (dạng số) để mô hình xử lý.



- Tham số tối ưu: Max length = 128 token để tiết kiệm RAM và tăng tốc độ huấn luyện trên Google Colab GPU T4



Sentiment Analysis



MÃ HÓA VĂN BẢN



12/27

```
MODEL_NAME = "vinai/phobert-large"
tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME)
def tokenize_function(examples):
    return tokenizer(
        examples["content_seg"],
        padding="max_length",
        truncation=True,
        max_length=128 # Giữ 128 là an toàn nhất để không tràn RAM
    )
print(f"⏳ Đang mã hóa dữ liệu với {MODEL_NAME}...")
tokenized_train = train_dataset.map(tokenize_function, batched=True)
tokenized_test = test_dataset.map(tokenize_function, batched=True)
print("✅ Mã hóa xong!")
```





MÔ HÌNH HỌC SÂU

Trong nghiên cứu này, mô hình học sâu được lựa chọn là PhoBERT, một mô hình ngôn ngữ tiền huấn luyện (Pre-trained Language Model) được thiết kế chuyên biệt cho tiếng Việt. PhoBERT được xây dựng dựa trên kiến trúc RoBERTa, là phiên bản cải tiến của BERT, với nhiều thay đổi giúp tối ưu hiệu suất xử lý ngôn ngữ tự nhiên.

PhoBERT được huấn luyện trên 20GB dữ liệu văn bản tiếng Việt, bao gồm báo chí, mạng xã hội và nhiều nguồn ngôn ngữ đa dạng. Toàn bộ dữ liệu được tách từ bằng RDRSegmenter trước khi huấn luyện, giúp mô hình học được cấu trúc từ ghép và ngữ nghĩa đặc thù của tiếng Việt.

Trong bài toán phân loại cảm xúc, mô hình PhoBERT được kết hợp với một lớp phân loại tuyến tính (classification head) nằm ở cuối mạng. Lớp này nhận đầu vào là vector biểu diễn của token đặc biệt CLS và ánh xạ thành ba nhãn cảm xúc:

- Tiêu cực
- Trung lập
- Tích cực

PHƯƠNG PHÁP MÔ HÌNH



🔍 Sentiment Analysis ➡



Group 17

CẤU HÌNH HUÂN LUYỆN MÔ HÌNH PHOBERT LARGE

```
model = AutoModelForSequenceClassification.from_pretrained(MODEL_NAME,
    num_labels=3).to(device)

def compute_metrics(eval_pred):
    logits, labels = eval_pred
    predictions = np.argmax(logits, axis=-1)
    acc = accuracy_score(labels, predictions)
    f1 = f1_score(labels, predictions, average='macro')
    return {"accuracy": acc, "f1_macro": f1}

per_device_train_batch_size=4,      # Giảm xuống 4 (hoặc 2 nếu vẫn
                                    # lỗi)
gradient_accumulation_steps=4,     # Tích lũy 4 lần (Tương đương
batch_size_thực_t嚮 = 16)
per_device_eval_batch_size=8,       # Lúc test thì nhẹ hơn, để 8 được
fp16=True,                         # BẮT chế độ 16-bit (Giảm 50% RAM,
                                    # chạy nhanh hơn)
                                    # -----
eval_strategy="epoch",
save_strategy="epoch",
save_total_limit=1,                 # Chỉ giữ 1 file model tốt nhất
learning_rate=1e-5,                  # Model lớn cần học chậm hơn
                                    # (1e-5)
weight_decay=0.01,
load_best_model_at_end=True,
metric_for_best_model="f1_macro",
report_to="none"
)

eval_strategy="epoch",
save_strategy="epoch",
save_total_limit=1

learning_rate=1e-5,
weight_decay=0.01
load_best_model_at_end=True,
metric_for_best_model="f1_macro",
```

14/27





CẤU HÌNH HUÂN LUYỆN MÔ HÌNH PHOBERT LARGE

```
trainer = Trainer(  
    model=model,  
    args=training_args,  
    train_dataset=tokenized_train,  
    eval_dataset=tokenized_test,  
    compute_metrics=compute_metrics,  
)  
print(f"✅ Đã thiết lập Trainer cho {MODEL_NAME}!")
```





Mô tả chi tiết quá trình:

- Hàm `trainer.train()` thực thi toàn bộ pipeline huấn luyện:
 - Tạo các mini-batch theo cấu hình Batch Size và Gradient Accumulation
 - Tính toán Loss và tối ưu hóa bằng thuật toán AdamW
 - Theo dõi các chỉ số như Accuracy và F1-macro sau mỗi epoch
 - Áp dụng FP27 để tăng tốc độ huấn luyện và giảm mức sử dụng bộ nhớ
 - Lưu lại mô hình tốt nhất theo tiêu chí F1-macro
 - Trong suốt quá trình này, Trainer tự động sử dụng GPU (nếu có) để đẩy nhanh tốc độ xử lý.
- Cơ chế dừng và chọn mô hình tối ưu:
- Dựa trên cấu hình ở phần 5, Trainer sẽ:
- Đánh giá mô hình sau mỗi epoch.
 - Lưu checkpoint mô hình tương ứng.
 - Giữ lại mô hình có F1-macro cao nhất.
 - Tự động tải lại mô hình này sau khi huấn luyện hoàn tất.

HUẤN LUYỆN MÔ HÌNH PHOBERT LARGE

```
print("🚀 BẮT ĐẦU HUẤN LUYỆN PHOBERT LARGE...")  
  
trainer.train()
```

(Q Sentiment Analysis ⌂)





ĐÁNH GIÁ MÔ HÌNH

Các chỉ số chính:

- Accuracy: Tỷ lệ dự đoán đúng tổng thể.
- Precision, Recall, F1-Score: Đánh giá chi tiết cho từng lớp cảm xúc.
 - F1-Macro (Trung bình F1 3 lớp) được sử dụng làm chỉ số đánh giá chính vì dữ liệu thường mất cân bằng (lớp Tích cực chiếm đa số).
- Ma trận nhầm lẫn (Confusion Matrix): Quan sát trực tiếp số lượng mẫu bị dự đoán sai giữa các lớp. * Đường cong ROC-AUC: Đo khả năng phân biệt giữa các lớp (sử dụng One-vs-Rest).



🔍 Sentiment Analysis ➡



Group 17

ĐÁNH GIÁ MÔ HÌNH

Sentiment Analysis

```
history = trainer.state.log_history
train_loss = [x['loss'] for x in history if 'loss' in x]
eval_loss = [x['eval_loss'] for x in history if 'eval_loss' in x]
eval_f1 = [x['eval_f1_macro'] for x in history if 'eval_f1_macro' in x]
vé biểu đồ Loss và F1

plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
plt.plot(train_loss, label='Train Loss', marker='o')
if len(eval_loss) > 0:
    plt.plot(eval_loss, label='Eval Loss', marker='o')
plt.title('Biểu đồ Loss (mất mát)')
plt.xlabel('Steps/Epochs')
plt.ylabel('Loss')
plt.legend()
plt.subplot(1, 2, 2)
plt.plot(eval_f1, label='Val F1-Macro', color='green', marker='o')
plt.title('Biểu đồ F1-Score (Càng cao càng tốt)')
plt.xlabel('Epochs')
plt.ylabel('F1 Score')
plt.legend()
plt.show()
print("\n--- KẾT QUẢ ĐÁNH GIÁ ---")
preds_output = trainer.predict(tokenized_test)
y_preds = np.argmax(preds_output.predictions, axis=-1)
y_true = preds_output.label_ids

# In báo cáo
target_names = ['Negative', 'Neutral', 'Positive']
print(classification_report(y_true, y_preds,
                            target_names=target_names))
```

18/27

Vẽ Ma Trận Nhầm Lẫn
(Confusion Matrix)

```
cm = confusion_matrix(y_true, y_preds)

plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=target_names, yticklabels=target_names)

plt.title('Ma Trận Nhầm Lẫn')
plt.xlabel('Thực tế')
plt.ylabel('Dự đoán')
plt.show()
```

Ý nghĩa phân tích:

- Đường chéo chính (từ trái → phải) càng lớn → mô hình dự đoán chính xác cao.
- Nếu mô hình thường nhầm lẫn:
 - Negative ↔ Neutral → có thể vì câu mang tính "hở chê một chút"
 - Neutral ↔ Positive → các bình luận chung chung dễ gây nhiễu





LƯU TRỮ SAU HUẤN LUYỆN

```
save_path = "./my_phobert_sentiment"  
  
trainer.save_model(save_path)  
  
tokenizer.save_pretrained(save_path)  
  
print(f"✅ Đã lưu model tại: {save_path}")  
  
print("💡 Bạn có thể nén thư mục này lại và tải về máy để sử dụng offline.  
!zip -r phobert_model.zip ./my_phobert_sentiment  
  
print("📦 Đã nén thành phobert_model.zip. Hãy tải về từ mục Files bên  
trái!")
```

Ý nghĩa:

- Tạo file phobert_model.zip, có thể tải về máy để sử dụng offline.
- Dễ dàng triển khai lên server, API hoặc dự án khác mà không cần huấn luyện lại.





KẾT QUẢ ĐẠT ĐƯỢC



Kết quả cho thấy mô hình đạt độ chính xác khoảng 94%.
hoạt động ổn định khi phân
loại ba nhóm cảm xúc: Tích cực, Trung lập và Tiêu cực.
Ứng dụng được minh họa thông qua
giao diện Gradio cho phép dự đoán cảm xúc theo thời
gian



Nhận xét ngắn gọn:
● AUC của cả ba lớp đều cao hơn 0.80, cho thấy mô hình phân biệt
cảm xúc tốt.
● Lớp Positive và Negative thường đạt AUC cao nhất do đặc điểm
câu rõ ràng.
● Lớp Neutral có AUC thấp hơn vì câu trung lập thường khó phân
loại.

🔍 Sentiment Analysis ➔



DEMO THỰC TẾ

KHÓI 9: TEST THỬ NGHIỆM

```
def predict_phobert(text):
    # a. Tách từ
    text_seg = word_tokenize(text, format="text")

    # b. Mã hóa
    inputs = tokenizer(text_seg, return_tensors="pt", truncation=True, max_length=128,
padding=True).to(device)

    # c. Dự đoán
    with torch.no_grad():
        logits = model(**inputs).logits

    # d. Lấy nhãn
    pred_idx = torch.argmax(logits, dim=1).item()
    probs = torch.nn.functional.softmax(logits, dim=1)[0].cpu().numpy()

    labels = {0: 'Negative 😞', 1: 'Neutral 😐', 2: 'Positive 😊'}
    return labels[pred_idx], probs[pred_idx]
```

Giải thích chi tiết pipeline:

- 1.Tách từ
2. PhoBERT yêu cầu văn bản được tách từ theo định dạng “giao_hàng”, “chất_lượng”.
- 3.Mã hóa
4. Tokenizer chuyển văn bản thành input_ids và attention_mask.
- 5.Dự đoán
 - Mô hình trả về bộ logits cho ba lớp cảm xúc.
 - Không cần tính gradient → dùng torch.no_grad() để tiết kiệm bộ nhớ.

Ánh xạ nhãn

Lấy nhãn dự đoán và độ tin cậy (softmax).





DEMO THỰC TẾ

```
Test các câu "bậy"
samples = [
    "Sách bìa đẹp nhưng nội dung thì rỗng tuếch.",
    "Không phải là không tốt, tạm chấp nhận được.",
    "Sách đều, giấy mỏng dinh, đừng mua phi tiềnn.",
    "Giao hàng nhanh, đóng gói kỹ, rất ưng ý.",
    "Shop làm ăn bát nháo, treo đầu dê bán thịt chó"
]

print("-" * 50)
print("KẾT QUẢ DỰ ĐOÁN PHOBERT:")
print("-" * 50)

for s in samples:
    lbl, conf = predict_phobert(s)
    print(f"\ud83d\udcbb: {s}")
    print(f"\ud83d\udcbb: {lbl} (Độ tin cậy: {conf*100:.2f}%)")
    print("-" * 30)
```

Nhận xét kết quả

Dựa trên các câu thử nghiệm:

✓ Câu khen → mô hình dự đoán Positive với độ tin cậy cao

Ví dụ:

"Giao hàng nhanh, đóng gói kỹ, rất ưng ý." → Positive 😊

✓ Câu chê mạnh → mô hình dự đoán Negative

"Shop làm ăn bát nháo, treo đầu dê bán thịt chó" →

Negative 😞

✓ Câu mơ hồ → mô hình dự đoán Neutral

"Không phải là không tốt, tạm chấp nhận được." →

Neutral 😐

→ Đây là điểm mạnh của PhoBERT-Large so với các mô

hình nhỏ.

✓ Mô hình hiểu ngữ cảnh tốt

Các câu có cấu trúc phủ định kép, hoặc có tính ẩn ý, mô hình vẫn phân tích chính xác → khẳng định khả năng học sâu của PhoBERT.





ĐÁNH GIÁ BỎ SUNG BẰNG ĐƯỜNG CONG ROC-AUC



Sentiment Analysis

```
raw_pred = trainer.predict(tokenized_test)
y_score = raw_pred.predictions # logits
# Chuyển logits thành xác suất (softmax)
y_prob = torch.nn.functional.softmax(torch.tensor(y_score),
dim=1).numpy()

y_test_bin = label_binarize(test_dataset['label'], classes=[0, 1,
2])

n_classes = 3

# 2. Tính toán ROC cho từng lớp
fpr = dict()
tpr = dict()
roc_auc = dict()

for i in range(n_classes):
    fpr[i], tpr[i], _ = roc_curve(y_test_bin[:, i], y_prob[:, i])
    roc_auc[i] = auc(fpr[i], tpr[i])

plt.figure(figsize=(10, 8))

colors = cycle(['red', 'blue', 'green'])
classes = ['Negative (Tiểu cực)', 'Neutral (Trung lập)', 'Positive
(Tích cực)']
for i, color in zip(range(n_classes), colors):
    plt.plot(fpr[i], tpr[i], color=color, lw=2,
label='ROC của lớp %d (area = %.2f)' %
''.format(classes[i], roc_auc[i]))

plt.plot([0, 1], [0, 1], 'k--', lw=2)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('Tỷ lệ Dương tính Giả (False Positive Rate)')
plt.ylabel('Tỷ lệ Dương tính Thật (True Positive Rate)')
plt.title('Biểu đồ ROC-AUC dla lớp cho PhobERT Large')
plt.legend(loc="lower right")
plt.show()
```

Nhận xét ngắn gọn:

- AUC của cả ba lớp đều cao hơn 0.80, cho thấy mô hình phân biệt cảm xúc tốt.
- Lớp Positive và Negative thường đạt AUC cao nhất do đặc điểm câu rõ ràng.
- Lớp Neutral có AUC thấp hơn vì câu trung lập thường khó phân loại.





Cho phép người dùng nhập câu tiếng Việt và nhận kết quả xác suất 3 lớp cảm xúc theo thời gian thực.

Chứng minh khả năng triển khai thực tế của mô hình.

ỨNG DỤNG WEB BẰNG GRADIO



Sentiment Analysis





Group 17

ỨNG DỤNG WEB BẰNG GRADIO

25/27

```
!pip install gradio
def gradio_predict(text):
    # 1. Tách từ
    text_seg = word_tokenize(text, format="text")

    # 2. Mã hóa
    inputs = tokenizer(text_seg, return_tensors="pt", truncation=True,
max_length=128, padding=True).to(device)

    # 3. Dự đoán
    with torch.no_grad():
        logits = model(**inputs).logits
        probs = torch.nn.functional.softmax(logits, dim=1)[0]

    # 4. Trả về dạng Dictionary cho Gradio
    return {
        "Negative": float(probs[0]),
        "Neutral": float(probs[1]),
        "Positive": float(probs[2])
    }
```

Sentiment Analysis

```
demo = gr.Interface(
    fn=gradio_predict,
    inputs=gr.Textbox(lines=3, placeholder="Nhập bình luận Tiki/Shopee vào đây..."),
    outputs=gr.Label(num_top_classes=3),
    title="💡 AI PHÂN TÍCH CẢM XÚC (PROBERT LARGE)",
    description="Nhập một câu bình luận tiếng Việt, AI sẽ đoán xem đó là Khen, Chê hay Bình thường.",
    examples=[
        {"Ciao hàng nhanh, đóng gói cẩn thận, sách đẹp."},
        {"Sách bìa thi đep nhưng nội dung sáo rỗng, phí tiền."},
        {"Cũng tạm được, không có gì đặc sắc."},
        {"Shop lừa đảo, treo đầu dê bán thịt chó."]
    ],
    theme="light"
)

# Chạy ứng dụng
print("⚡ Đang khởi động Web App...")
demo.launch(share=True, debug=True)
```





Group 17

26/27

KẾT LUẬN

HƯỚNG PHÁT TRIỀN



Sentiment Analysis



Trong đồ án này, nhóm đã xây dựng thành công một hệ thống phân tích cảm xúc tiếng Việt sử dụng mô hình PhoBERT-Large. Thông qua các bước tiền xử lý dữ liệu, tách từ tiếng Việt, mã hóa văn bản, câu hình huấn luyện tối ưu và triển khai mô hình, hệ thống đã đạt được hiệu suất tổng thể tốt trên tập dữ liệu đánh giá. Mô hình phân loại chính xác các bình luận tích cực, tiêu cực và trung lập, đồng thời duy trì F1-macro cao và đường cong ROC-AUC thể hiện độ phân biệt tốt giữa các lớp.

Kết quả cho thấy PhoBERT-Large là mô hình mạnh cho các tác vụ phân tích ngôn ngữ tiếng Việt, đặc biệt là trong bối cảnh dữ liệu phản hồi người dùng như đánh giá sản phẩm. Hệ thống demo với Gradio cũng chứng minh khả năng ứng dụng thực tế, có thể sử dụng trong thương mại điện tử, chatbot chăm sóc khách hàng, hoặc phân tích dữ liệu.

Mặc dù còn tồn tại một số thách thức như phân loại câu trung lập và hạn chế về tài nguyên khi huấn luyện mô hình lớn, kết quả hiện tại vẫn đáp ứng tốt yêu cầu bài toán. Trong tương lai, có thể mở rộng mô hình bằng cách sử dụng tập dữ liệu lớn hơn, fine-tune sâu hơn hoặc tích hợp thêm các phương pháp tăng cường dữ liệu để cải thiện độ chính xác.



Group 17

27/27

THANK YOU!

+123-456-7890

hello@reallygreatsite.com

Sentiment Analysis



Sentiment Analysis

